IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent of:	Sekiguchi et al.	
U.S. Patent No.:	8,230,101	Attorney Docket No. 50095-0245IP1
Issue Date:	July 24, 2012	
Appl. Serial No.:	12/527,777	
Filing Date:	September 2, 2009	
Title:	SERVER DEVICE FOR MEDIA, METHOD FOR CONTROLLING SERVER FOR MEDIA, AND PROGRAM	

DECLARATION OF EREZ ZADOK, PH.D.

I hereby declare that all statements made of my own knowledge are true and that all statements made on information and belief are believed to be true. I further declare that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of the Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patents issued thereon.



Dated: June 4, 2025 By: _____

Erez Zadok, PhD

TABLE OF CONTENTS

I. Table of Claims....8
II. Introduction and Scope of Work....18
III. Qualifications....19
IV. Materials Considered....29
V. Summary of Conclusions....33
VI. Legal Principles....33
A. Person of Ordinary Skill in the Art....33
B. Obviousness....34
C. Claim Construction Standard....35
VII. Person of Ordinary Skill in the Art....37
A. A POSITA's Ordinary Level of Skill....37
VIII. Technology Overview....38
A. General Computer Operations....38
B. Networking....44
1. Networking Overview....44
2. Network Failures and Mitigation....49
3. Network Protocols....53
C. Data Storage Devices, Files, and File Systems....58
1. Hierarchical File Systems....63
D. Networked and Distributed Storage Systems....67
1. Multi-Tier and Caching Architectures....72
E. Multi-Media Streaming....75
1. Copy Protection and Digital Rights Management....80
2. UPnP Standard....83
F. Caching Technologies....84
1. Least-Recently Used (LRU)....92
2. Least-Frequently Used (LFU)....93
IX. Overview of the '101 Patent....95
A. Overview of the '101 Patent....95
B. Summary of Prosecution History of the '101 Patent....101
C. Priority Date of the Challenged Claims....104
X. Summary of the Prior Art....104
1. Overview of Lamkin (EX1004)....104
2. Overview of Franke (EX1005)....108

3. Overview of Fiechter (EX1006) 112
4. Overview of Ito (EX1012) 114

XI. GROUND 1A: Lamkin Renders Claims 1-12 Obvious 115
A. Claim 1 115
1. Claim [1pre]: A server device for media, the server device for media comprising 115
2. Claim [1a-i]: an internal storage device for storing digital contents 117
3. Claim [1a-ii]: wherein the server device for media responds to a data transmission request from a network player by stream-delivering corresponding data in corresponding digital contents from the internal storage device to the network player during connection to a network 121
4. Claim [1b-i]: a transfer control unit adapted to transfer and store part of held digital contents in the internal storage device to a network storage device 126
5. Claim [1b-ii]: wherein the network storage device is connected to the network and is capable of storing data 132
6. Claim [1b-iii]: wherein said transfer control unit does not transfer, from the internal storage device to the network storage device, the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device. 134
7. Claim [1c-i]: a list information transmission unit adapted to respond to a list presentation request for the held digital contents of the server device for media from the network player by transmitting list information to the network player 140
8. Claim [1c-ii]: wherein the list information lists the digital contents left in the internal storage device and the digital contents transferred from the internal storage device to the network storage device and stored in the network storage device 143
9. Claim [1c-iii]: wherein the list information maintains a tree structure of the digital contents in the internal storage device before transferring the digital contents to the network storage device. 145
10. Claim [1d]: a search unit adapted to respond to a data transmission request for the held digital contents from the network player by searching for a location where the held digital contents are currently stored 149
11. Claim [1e]: a digital contents data transmission processing unit adapted to allow the corresponding data in held digital contents to be stream-

delivered from the network storage device to the network player, if the result of search shows the network storage device 154
12. Claim [1f]: wherein the server device for media is a media player. 157
B. Claim 6.....159
1. Element [6b-iii]: wherein the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device is transferred after obtaining permission from a user.....159
C. Dependent Claims 2-5 and 8-11 163
1. Claim 2: The server device for media according to claim 1, wherein said digital contents data transmission processing unit causes the network storage device to transmit the corresponding data to the server device for media, and then transmits the corresponding data received from the network storage device from the server device for media to the network player.....163
2. Claim 3: The server device for media according to claim 1, wherein said digital contents data transmission processing unit transmits the corresponding data and information for identifying the network storage device to the network player, and causes the network storage device to directly transmit the corresponding data to the network player.....165
3. Claim 4: The server device for media according to claim 1, further comprising a return control unit adapted to cause the digital contents corresponding to a predetermined condition among the digital contents which have been transferred to the network storage device to be returned from the network storage device to the internal storage device 170
4. Claim 5: The server device for media according to claim 1, wherein said list information transmission unit makes the list information to be transmitted to the network player include information for identifying whether each digital content is currently stored in the internal storage device or the network storage device in the display list of the network player.....176
XII. GROUND 1B: Lamkin in view of Fiechter Renders Claims 1-12 Obvious 178
A. Element [1b-iii]: wherein said transfer control unit does not transfer, from the internal storage device to the network storage device, the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device.....182
B. Element [6b-iii]: wherein the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents

from the internal storage device to the network storage device is transferred after obtaining permission from a user.....184
C. Motivations for the Combination of Lamkin and Fiechter.....185
XIII. GROUND 1C: Lamkin in view of Ito Renders Claims 1-5 and 7-11 Obvious 189
A. Element [1b-iii]: wherein said transfer control unit does not transfer, from the internal storage device to the network storage device, the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device.....192
B. Motivations for the Combination of Lamkin and Ito 193
XIV. GROUND 2A: Franke Renders Claims 1-12 Obvious 197
A. Claim 1.....197
Claim.....197
1. [1pre]: A server device for media, the server device for media comprising 197
2. Claim [1a-i]: an internal storage device for storing digital contents 199
Claim.....202
3. [1a-ii]: wherein the server device for media responds to a data transmission request from a network player by stream-delivering corresponding data in corresponding digital contents from the internal storage device to the network player during connection to a network.....202
Claim.....205
4. [1b-i]: a transfer control unit adapted to transfer and store part of held digital contents in the internal storage device to a network storage device 205
5. Claim [1b-ii]: wherein the network storage device is connected to the network and is capable of storing data.....210
Claim.....212
6. [1b-iii]: wherein said transfer control unit does not transfer, from the internal storage device to the network storage device, the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device 212
7. Claim [1c-i]: a list information transmission unit adapted to respond to a list presentation request for the held digital contents of the server device for media from the network player by transmitting list information to the network player.....215

Claim.....219
8. [1c-ii]: wherein the list information lists the digital contents left in the internal storage device and the digital contents transferred from the internal storage device to the network storage device and stored in the network storage device 219
9. Claim [1c-iii]: wherein the list information maintains a tree structure of the digital contents in the internal storage device before transferring the digital contents to the network storage device 221
Claim.....225
10. [1d]: a search unit adapted to respond to a data transmission request for the held digital contents from the network player by searching for a location where the held digital contents are currently stored 225
Claim [1e]: a digital contents data transmission processing unit adapted to allow the corresponding data in held digital contents to be stream-delivered 228
11. from the network storage device to the network player, if the result of search shows the network storage device 228
12. Claim [1f]: wherein the server device for media is a media player 233
B. Claim 6.....234
1. Element [6b-iii]: wherein the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device is transferred after obtaining permission from a user 234
C. Dependent Claims 2-5 and 8-11 238
Claim 2:.....238
1. The server device for media according to claim 1, wherein wherein said digital contents data transmission processing unit causes the network storage device to transmit the corresponding data to the server device for media, and then transmits the corresponding data received from the network storage device from the server device for media to the network player 238
Claim 3:.....241
2. The server device for media according to claim 1, wherein said digital contents data transmission processing unit transmits the corresponding data and information for identifying the network storage device to the network player, and causes the network storage device to directly transmit the corresponding data to the network player 241
Claim 4:.....244
3. The server device for media according to claim 1, further comprising a return control unit adapted to cause the digital contents

corresponding to a predetermined condition among the digital contents which have been transferred to the network storage device to be returned from the network storage device to the internal storage device 244
Claim 5:.....247
4. The server device for media according to claim 1, wherein said list information transmission unit makes the list information to be transmitted to the network player include information for identifying whether each digital content is currently stored in the internal storage device or the network storage device in the display list of the network player 247
XV. GROUND 2B: Franke in view of Fiechter Renders Claims 1-12 Obvious 250
Element [1b-iii]:.....253
A. wherein said transfer control unit does not transfer, from the internal storage device to the network storage device, the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device.....253
Element [6b-iii]:.....254
B. wherein the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device is transferred after obtaining permission from a user 254
C. Motivations for the Combination Franke and Fiechter 256
XVI. GROUND 2C: Franke in view of Ito Renders Claims 1-5 and 7-11 Obvious 261
A. Element [1b-iii]: wherein said transfer control unit does not transfer, from the internal storage device to the network storage device, the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device.....263
B. Motivation for the Combination Franke and Ito.....265
XVII. ADDITIONAL REMARKS/CONCLUSION 267

I. TABLE OF CLAIMS

Claim 1	Claim 7	Claim 6	Claim 12
[1pre] A server device for media, the server device for media comprising:	**[7pre]** A method for controlling a server device for media which is equipped with an internal storage device for storing digital contents, the method comprising the steps of:	**[6pre]** A server device for media, the server device for media comprising:	**[12pre]** A method for controlling a server device for media which is equipped with an internal storage device for storing digital contents, the method comprising the steps of:
[1a-i] an internal storage device for storing digital contents,	**[7a]** responding to a data transmission request from a network player by stream-delivering corresponding data in corresponding digital contents from the internal storage device to the network player	**[6a-i]** an internal storage device for storing digital contents,	**[12a]** responding to a data transmission request from a network player by stream-delivering corresponding data in corresponding digital contents from the internal storage device to the network player
[1a-ii] wherein the server device for media responds to a data transmission request from a network player by stream-delivering corresponding		**[6a-ii]** wherein the server device for media responds to a data transmission request from a network player by stream-delivering corresponding	

Claim 1	Claim 7	Claim 6	Claim 12
data in corresponding digital contents from the internal storage device to the network player during connection to a network;	during connection to a network;	data in corresponding digital contents from the internal storage device to the network player during connection to a network;	during connection to a network;
[1b-i] a transfer control unit adapted to transfer and store part of held digital contents in the internal storage device to a network storage device,	**[7b-i]** transferring and storing part of held digital contents in the internal storage device to a network storage device,	**[6b-i]** a transfer control unit adapted to transfer and store part of held digital contents in the internal storage device to a network storage device,	**[12b-i]** transferring and storing part of held digital contents in the internal storage device to a network storage device,
[1b-ii] wherein the network storage device is connected to the network and is capable of storing data,	**[7b-ii]** wherein the network storage device is connected to the network and is capable of storing data,	**[6b-ii]** wherein the network storage device is connected to the network and is capable of storing data,	**[12b-ii]** wherein the network storage device is connected to the network and is

Claim 1	Claim 7	Claim 6	Claim 12
			capable of storing data,
[1b-iii] and wherein said transfer control unit does not transfer, from the internal storage device to the network storage device, the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device;	**[7b-iii]** and wherein the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents are not transferred from the internal storage device to the network storage device;	**[6b-iii]** and wherein the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device is transferred after obtaining permission from a user;	**[12b-iii]** and wherein the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device is transferred after obtaining permission from a user;
[1c-i] a list information	**[7c-i]** responding to a list	**[6c-i]** a list information	**[12c-i]** responding to a list

Claim 1	Claim 7	Claim 6	Claim 12
transmission unit adapted to respond to a list presentation request for the held digital contents of the server device for media from the network player by transmitting list information to the network player,	presentation request for the held digital contents of the server device for media from the network player by transmitting list information to the network player,	transmission unit adapted to respond to a list presentation request for the held digital contents of the server device for media from the network player by transmitting list information to the network player,	presentation request for the held digital contents of the server device for media from the network player by transmitting list information to the network player,
[1c-ii] wherein the list information lists the digital contents left in the internal storage device and the digital contents transferred from the internal storage device to the network storage device and	**[7c-ii]** wherein the list information lists the digital contents left in the internal storage device and the digital contents transferred from the internal storage device to the network storage device and	**[6c-ii]** wherein the list information lists the digital contents left in the internal storage device and the digital contents transferred from the internal storage device to the network storage device and	**[12c-ii]** wherein the list information lists the digital contents left in the internal storage device and the digital contents transferred from the internal storage device to the network

Claim 1	Claim 7	Claim 6	Claim 12
stored in the network storage device,	stored in the network storage device,	stored in the network storage device,	storage device and stored in the network storage device,
[1c-iii] and wherein the list information maintains a tree structure of the digital contents in the internal storage device before transferring the digital contents to the network storage device;	**[7c-iii]** and wherein the list information maintains a tree structure of the digital contents in the internal storage device before transferring the digital contents to the network storage device;	**[6c-iii]** and wherein the list information maintains a tree structure of the digital contents in the internal storage device before transferring the digital contents to the network storage device;	**[12c-iii]** and wherein the list information maintains a tree structure of the digital contents in the internal storage device before transferring the digital contents to the network storage device;
[1d] a search unit adapted to respond to a data transmission request for the held digital contents from the	**[7d]** responding to a data transmission request for the held digital contents from the network player by	**[6d]** a search unit adapted to respond to a data transmission request for the held digital contents from the	**[12d]** responding to a data transmission request for the held digital contents from the network player by

Claim 1	Claim 7	Claim 6	Claim 12
network player by searching for a location where the held digital contents are currently stored; and	searching for a location where the held digital contents are currently stored; and	network player by searching for a location where the held digital contents are currently stored; and	searching for a location where the held digital contents are currently stored; and
[1e] a digital contents data transmission processing unit adapted to allow the corresponding data in held digital contents to be stream-delivered from the network storage device to the network player, if the result of search shows the network storage device,	**[7e]** allowing the corresponding data in held digital contents to be stream-delivered from the network storage device to the network player, if the result of search shows the network storage device,	**[6e]** a digital contents data transmission processing unit adapted to allow the corresponding data in held digital contents to be stream-delivered from the network storage device to the network player, if the result of search shows the network storage device,	**[12e]** allowing the corresponding data in held digital contents to be stream-delivered from the network storage device to the network player, if the result of search shows the network storage device,
[1f] wherein the server device for	**[7f]** wherein the service device for	**[6f]** wherein the server device for	**[12f]** wherein the server device for

Claim 1	Claim 7	Claim 6	Claim 12
media is a media player.	media is a media player.	media is a media player.	media is a media player.
[2] The server device for media according to claim 1, wherein said digital contents data transmission processing unit causes the network storage device to transmit the corresponding data to the server device for media, and then transmits the corresponding data received from the network storage device from the server device for media to the network player.	**[8]** The server device for media according to claim 6, wherein said digital contents data transmission processing unit causes the network storage device to transmit the corresponding data to the server device for media, and then transmits the corresponding data received from the network storage device from the server device for media to the network player.		

Claim 1	Claim 7	Claim 6	Claim 12
[3] The server device for media according to claim 1, wherein said digital contents data transmission processing unit transmits the corresponding data and information for identifying the network storage device to the network player, and causes the network storage device to directly transmit the corresponding data to the network player.	**[9]** The server device for media according to claim 6, wherein said digital contents data transmission processing unit transmits the corresponding data and information for identifying the network storage device to the network player, and causes the network storage device to directly transmit the corresponding data to the network player.		
[4] The server device for media according to claim	**[10]** The server device for media according to claim		

Claim 1	Claim 7	Claim 6	Claim 12
1, further comprising a return control unit adapted to cause the digital contents corresponding to a predetermined condition among the digital contents which have been transferred to the network storage device to be returned from the network storage device to the internal storage device.	6, further comprising a return control unit adapted to cause the digital contents corresponding to a predetermined condition among the digital contents which have been transferred to the network storage device to be returned from the network storage device to the internal storage device.		
[5] The server device for media according to claim 1, wherein said list information	**[11]** The server device for media according to claim 6, wherein said list information		

Claim 1	Claim 7	Claim 6	Claim 12
transmission unit makes the list information to be transmitted to the network player include information for identifying whether each digital content is currently stored in the internal storage device or the network storage device in the display list of the network player.	transmission unit makes the list information to be transmitted to the network player include information for identifying whether each digital content is currently stored in the internal storage device or the network storage device in the display list of the network player.		

I, Erez Zadok, Ph.D., declare that:

II. INTRODUCTION AND SCOPE OF WORK

1. I have been retained by Fish & Richardson P.C., on behalf of Apple ("Petitioner"), as an independent expert consultant in this *inter partes* review ("IPR") proceeding before the United States Patent and Trademark Office.

2. I have been asked by Petitioner's counsel ("Counsel") to consider whether certain references teach or suggest the features recited in Claims 1-12 of U.S. Patent No. 8,230,101 ("the '101 patent") (EX1001). My opinions and the bases for my opinions are set forth below. My opinions are based on my education and experience.

3. In writing this Declaration, I have considered the following: the prior art publications cited below and my own knowledge and experience, including my teaching and work experience in the below-described fields, and my experience of working with others in those fields during the relevant period.

4. I have no direct financial interest in either party or in the outcome of this proceeding. My consulting company is being compensated for my work on an hourly basis for all tasks involved at my standard consulting rate. This compensation is not dependent on the outcome of these proceedings or on any pending litigation, and the compensation is also not dependent on the content of my opinions.

III. QUALIFICATIONS

5. My *curriculum vitae*, which includes a more detailed summary of my background, experience and publications, is attached as APPENDIX-A.

6. I am a Professor in the Computer Science Department at Stony Brook University (part of the State University of New York ("SUNY") system). I direct the File-systems and Storage Lab (FSL) at Stony Brook's Computer Science Department. My research interests include file systems and storage systems, operating systems, transactional systems including database technologies, information technology and system administration, security/privacy and information assurance, networking, energy efficiency, performance and benchmarking, virtualization, cloud systems, compilers, applied machine learning, and software engineering.

7. I studied at a professional high school in Israel, focusing on electrical engineering ("EE"), and graduated in 1982. I spent one more year at the high school's college division, receiving a special Certified Technician's degree in EE. I then went on to serve in the Israeli Defense Forces for three years (1983–1986). I received my Bachelor of Science degree in computer science ("CS") in 1991, my Master's degree in CS in 1994, and my PhD in CS in 2001—all from Columbia University in New York.

8. When I began my undergraduate studies at Columbia University, I also started working as a student assistant in the various campus-wide computer labs, eventually becoming an assistant to the head labs manager, who was managing all public computer labs on campus. During that time, I also became more involved with research within the CS Department at Columbia University, conducting research on operating systems, file and storage systems, distributed and networked systems, security, and other topics. I also assisted the CS department's computer administrators in managing the department's computers, which included storage, IT, networking, and cyber-security related duties.

9. In 1991, I joined Columbia University's CS department as a full-time systems administrator, studying towards my MS degree part-time. My MS thesis topic is related to file system reliability, fault tolerance, replication, and failover in mobile networked storage systems using file virtualization. My main duties as a systems administrator involved installing, configuring, and managing many networked servers, proxies, and desktops running several operating systems, as well as network devices setup; this included many software and hardware upgrades, device upgrades, and BIOS firmware/chipset updates/upgrades. My duties also included ensuring reliable, secure, authenticated access to networked systems/storage and licensed software, as well as software updates, security and bug fixes. Examples of servers and their protocols included email (SMTP), file transfer

(FTP), domain names (DNS), network file systems (NFS), network news systems (NNTP), and Web (HTTP).

10. In 1994, I left my systems administrator position to pursue my doctoral studies at Columbia University. My PhD thesis topic was on versatile file system development using stackable (virtualized) file systems, with examples in the fields of security and encryption, efficiency, reliability, and failover. I continued to work part-time as a systems administrator at the CS department, and eventually I was asked to serve as manager to the entire information technology ("IT") staff. From 1991 to 2001, I was also a member of the faculty-level Facilities Committee that oversaw all IT operations at the CS department.

11. As part of my PhD studies at Columbia, I collaborated on projects to develop advanced AI-like techniques to detect previously unknown viruses (a.k.a. "zero-day malware"), using data mining and rule-based detection. This work led to several highly cited papers (over 1,600 citations for one of the papers alone) and two patents. I also became a Teaching Assistant ("TA") for a first-ever Computer Security course given at Columbia University's CS department with Dr. Matt Blaze as instructor.

12. From 1990 to 1998, I consulted for SOS Corporation and HydraWEB Technologies as a systems administrator and programmer, managing data storage use and backup/restore duties, databases, web servers, as well as information

assurance and cyber-security (*e.g.*, malware protection, software licensing). From 1994 to 2000, I led projects at HydraWEB Technologies, and then became the Director of Software Development—overseeing the development of several products and appliances such as stateful firewalls and HTTP load-balancers, utilizing network-virtualization and high-availability techniques. From 2009 to 2019, I have consulted for Packet General Networks, a startup specializing in secure, virtualized, network storage and applications' data security in the cloud.

13. In 2001, I joined the faculty of Stony Brook University, a position I have held since that time. In 2002, I joined the Operations Committee, which oversees the IT operations of the CS department at Stony Brook University. From 2006 to 2010, I was the Director of IT Operations of the CS department. My day-to-day duties included setting policies regarding computing, hiring and training new staff, assisting any staff with topics of my specialty, defining requirements for new software/hardware, and purchasing. From 2010 to 2015, I had served as the Co-Chair to the Operations Committee. From 2016 to 2019, I oversaw the IT Operations as the Chair of the Operations Committee. A significant component of these duties included defining and helping implement policies for data management, to ensure the security of users and their data, and data reliability and availability, while minimizing the inconvenience and performance impact to users. I personally helped setup and maintain an initial virtual-host infrastructure in the department. Since late

2019, I've been a member of the department's Executive Committee that also oversees all IT operations.

14. In 2017, I became the department's Graduate Academic Adviser, advising all Master students (over 400 annually on average) and many other graduate students on an assortment of academic matters. In 2024, I took over as the department's Graduate Program Director, overseeing the entire graduate CS program (700-800 students annually on average).

15. Since 2001, I have personally configured and managed my own research lab's network. This includes setting up and configuring multiple storage systems (*e.g.*, NFS, CIFS/SMB, NAS), virtual and physical environments, applications such as database (*e.g.*, MySQL, PostgreSQL), Web servers (*e.g.*, Apache), and mail servers; user access control (*e.g.*, NIS, LDAP), backups and restores, snapshot policies, and more. I have personally installed, configured, changed, replaced parts, and upgraded components in numerous devices including mobile devices, laptops, desktops, and servers, both physical and virtual.

16. Since 1995, I have taught courses on operating systems, storage and file systems, advanced systems programming in Unix/C, systems administration, data structures, data/software security, and more. My courses often use storage, file systems, distributed systems, and system/network security as key teaching principles and practical examples for assignments and projects. I have taught these concepts

and techniques to my students, both to my direct advisees as well as in my courses. For example, in my graduate Operating Systems course, I often cover Linux's kernel mechanisms to protect users, applications, and data files, virtual file systems, as well as distributed storage systems (*e.g.*, NFS). And in the System Administration undergraduate course, I covered many topics such as networking, storage, backups, and configuring complex applications such as mail, web, and database servers.

17. My research often investigates computer systems from many angles: security, efficiency, energy use, scalability, reliability, portability, survivability, usability, ease-of-use, versatility, flexibility, and more. My research gives special attention to balancing five often-conflicting aspects of computer systems: performance, reliability, energy use, security, and ease-of-use.

18. Since joining Stony Brook University in 2001, my group in the File-systems and Storage Lab ("FSL") has developed many file systems and operating system extensions; examples include a highly-secure cryptographic file system, a portable copy-on-write ("COW") versioning file system, a tracing file system useful to detect intrusions, a replaying file system useful for forensics, a snapshotting and sandboxing file system, a namespace unification file system (that uses stackable, virtualized, file-based COW), an anti-virus file system, an integrity-checking file system, a load balancing and replication/mirroring file system, network file system extensions for security and performance, distributed secure cloud-based storage

systems, transactional key-value stores and file systems, OS-level embedded databases, a compiler to convert user-level C code to in-kernel efficient yet safe code, GCC plugins, stackable file system templates, and a Web-based backup system. Many of these projects used one form of virtualization or another (storage, network, host, etc.). I continue to maintain and release newer versions of some of these file systems and software.

19. I have published over 120 refereed publications (in ACM, IEEE, USENIX, and more). To date, my publications have been cited more than 10,000 times (as per Google Scholar as of May 19, 2025). My papers cover a wide range of related technologies such as file systems, storage systems, transactional systems, security, clouds and virtualization, performance benchmarking and optimization, energy efficiency, system administration, web systems, and more. I also published a book titled "Linux NFS and Automounter Administration" (Sybex, 2001), covering systems administration topics related to network storage and data security.

20. Some of my research has led to public software releases that have been used worldwide. I have publicly maintained the Amd Berkeley Automounter in a package called "am-utils" since 1992; this software helps administrators manage the multitude of file system mounts on dozens of different Unix systems, especially helping to automate access to multiple NFS/NAS storage volumes. Since 1997, I have maintained and released several stackable (virtualized) file system software

projects for Linux, FreeBSD, and/or Sun Solaris, in a package called FiST. One of my stackable file system encryption projects, called Cryptfs, became the basis for IBM's public release of eCryptfs, now part of Linux. Packet General Networks, for whom I have provided consulting services between 2009 and 2019, licensed another encryption file system called Ncryptfs. Another popular file system released in 2003, called Unionfs, offers virtual namespace unification, transparent shadow copying (a.k.a. copy-on-write or COW), file system snapshotting (*e.g.*, useful for forensics and disaster recovery), and the ability to save disk space by sharing a read-only copy of data across several computers, among other features.

21. My research and teaching make extensive use of data security features. For example, each time I taught the graduate operating system course, the first homework assignment includes the creation of a new system call that performs new or added functionality, often for encrypting a file or verifying its integrity; many of my other assignments cover topics of user/process access control, anti-virus filtering, and more. Since 2001, over 1,000 graduate students were exposed to these principles directly through my teaching and research at Stony Brook University.

22. Moreover, in an undergraduate course titled "Advanced Systems Programming in Unix/C," I cover many topics of system security and vulnerabilities, such as the structure of UNIX processes, and memory segments such as the heap and stack. This course covers details of several hundred Linux system calls. Often, the

first assignment for this course is to develop a tool to encrypt/decrypt files using advanced ciphers, use digital signatures to certify the cipher keys used, and reliably recover files in case of failures. Since 2001, several hundred undergraduate students were exposed to these principles directly through my teaching and research at Stony Brook University.

23. In another undergraduate course, System Administration, I taught network configuration, security, and storage configuration and reliability. In a special topics course on Storage Systems, I covered many topics such as data deduplication, RAID, transactional storage, storage hardware including modern Flash based ones, virtual storage, backup/restore, snapshots and continuous data protection ("CDP"), NAS and SAN, and NFS.

24. Overall, in addition to the aforementioned experience, my technical experience relevant to the '101 Patent at the time of the alleged invention included the following: I configured and research network servers including NFS; configured clients to utilize NFS servers, including setting up diskless workstations that can boot off of NFS servers; configured Web servers, Web clients/browsers, and Web proxies (*e.g.*, Squid); configured and managed hierarchical-storage management (HSM) systems, as well as backup systems; and researched, used, managed, configured, taught, and developed hierarchical, tree-based file systems and multi-

tier storage systems; developed, released software, and published work on namespace unification techniques.

25. My research has been supported by many federal and state grants as well as industry awards, including an NSF CAREER award, two IBM Faculty awards, two NetApp Faculty awards, a Western Digital award, a Facebook award, several Dell-EMC awards, and several equipment gifts. I received the 2008 SUNY Chancellor's Excellence in Teaching award, and the 2022 SUNY Chancellor's Award for Excellence in Scholarship and Creative Activities (both awards can be given only once in a lifetime). In 2021, I was named an ACM Distinguished Member for "Outstanding Scientific Contributions to Computing."

26. My service record to the community includes serving as the co-chair for the USENIX Annual Technical Conference in 2020 (ATC'20); serving as the co-chair for USENIX File and Storage Technologies (FAST'15) in 2015 and on the FAST Conference Steering Committee from 2015 to 2023; serving on the ACM HotStorage Steering Committee since 2021; and serving as the co-chair in 2012 and on the Steering Committee of the ACM SYSTOR conference since 2012. I have served as an Associate Editor to the ACM Transactions on Storage ("TOS") journal from 2009 to 2022; in 2022, I was named the Editor-in-Chief for ACM's TOS journal.

27. I am a named inventor on four patents, two titled "Systems and Methods for Detection of New Malicious Executables" (U.S. Patent No. 7,487,544, issued February 3, 2009; and U.S. Patent No. 7,979,907, issued July 12, 2011); and two more titled "Multi-Tier Caching," (U.S. Patent No. 9,355,109, issued May 31, 2016; and U.S. Patent 9,959,279, issued May 1, 2018).

28. I have been disclosed as a testifying expert in 19 cases (including *inter partes* review ("IPR") proceedings) in the past four years. I have been deposed 13 times and testified in trial twice.

IV. MATERIALS CONSIDERED

29. I have reviewed the '101 patent (EX1001) and relevant excerpts of its prosecution history (EX1002). For purposes of this Declaration, I have been asked to use March 2, 2007 as the Critical Date.

30. As part of my independent analysis for this Declaration, I have considered the following: my professional judgment; my education, experience and knowledge regarding electrical engineering, computer science, video and audio and communications sciences and technologies such as data storage architectures; my experience in teaching these subjects; my experience in working with others involved in those fields; the common knowledge of persons of ordinary skill prior to the Critical Date; and the following publications and materials.

Exhibit	Description	Abbreviation
1001	U.S. Patent No. 8,230,101 to Sekiguchi	’101 Patent
1002	File History of U.S. Patent No. 8,230,101	’101 Patent File History
1004	U.S. Patent Publication No. 2006/0161635 to Lamkin et al.	Lamkin
1005	U.S. Patent Publication No. 2003/0195924 to Franke et al.	Franke
1006	U.S. Patent No. 7,219,123 to Fiechter et al.	Fiechter
1007	U.S. Patent Publication No. 2004/0006606 to Marotta et al.	Marotta
1008	U.S. Patent Publication No. 2002/0184457 to Yuasa et al.	Yuasa
1009	U.S. Patent Publication No. 2008/0104219 to Kageyama et al.	Kageyama
1010	U.S. Patent Publication No. 2006/0184972 to Rafey et al.	Rafey
1011	U.S. Patent Publication No. 2007/0238471 to Bae et al.	Bae
1012	International Patent Publication No. WO2006/073040 to Ito et al. with certified English translation.	Ito
1013	U.S. Patent Publication No. 2002/0099952 to Lambert, et. al.	Lambert
1014	U.S. Patent Publication No. 2002/0010819 to Dye	Dye
1015	U.S. Patent Publication No. 2004/0220926 to Lamkin.	Lamkin2

Exhibit	Description	Abbreviation
1016	U.S. Patent Publication No. 2005/0281185 to Kawasaki.	Kawasaki
1017	Microsoft Computer Dictionary, 5th ed., 2002, excerpts.	Microsoft Computer Dictionary
1018	Abraham Silberschatz and Peter B. Galvin, Operating Systems Concepts, 4th Edition, 1994, Addison-Wesley Publishing, excerpts.	Silberschatz
1019	Andrew S. Tanenbaum, *Computer Networks*, 2nd ed., 1988, excerpts.	Tanenbaum
1020	W. Richard Stevens, *TCP/IP Illustrated Volume 1, The Protocols*, 1994, excerpts.	Stevens
1021	William R. Cheswick & Steven M. Bellovin, *Firewalls and Internet Security, Repelling the Wily Hacker*, 1994, excerpts.	Cheswick
1022	U.S. Patent No. 6,687,846 to Adrangi et al.	Adrangi
1023	U.S. Patent No. 6,487,663 to Jaisimha et al.	Jaisimha
1024	U.S. Patent No. 6,732,365 to Belknap et al.	Belknap
1025	U.S. Patent Publication No. 2008/0060081 to Van Den Heuvel.	Van Den Heuvel
1026	Digital Living Network Alliance (DLNA), *Overview and Vision*, White Paper, June 2004	DLNA Overview
1027	UPnP Forum Version 1.0 Approved Standard, *MediaServer:2 Device Template Version 1.01*, Document Version 1.00, May 31, 2006 ("UPnP")	UPnP MediaServer

Exhibit	Description	Abbreviation
1101	Complaint for Patent Infringement (August 20, 2024), Case No. 2-24-CV-00687 (EDTX), Document 1.	Infringement Contentions
1102	Appendix E-3—Claim Chart for U.S. Patent No. 8,230,101 Against Apple HomeKit Secure Video Products.	Infringement Contentions Appendix E-3
1103	Appendix E-2 to Complaint for Patent Infringement—Claim Chart for U.S. Patent No. 8,230,101 Against Products with HTTP Live Streaming (HLS).	Infringement Contentions Appendix E-2
1104	Appendix E-1 to Complaint for Patent Infringement—Claim Chart for U.S. Patent No. 8,230,101 Against Apple Products with iCloud Storage.	Infringement Contentions Appendix E-1

31. Note that my citations to non-patent literature throughout this Declaration reference the absolute page number added to the exhibit (as opposed to the original pagination of the document). For patent literature, I have used the column/line numbers or paragraph numbers.

32. Although this Declaration refers to selected portions of the cited references for the sake of brevity, it should be understood that these are examples, and that one of ordinary skill in the art would have viewed the references cited herein in their entirety and in combination with other references cited herein or cited within the references themselves. The references used in this declaration, therefore, should be viewed as being incorporated herein in their entireties.

33. Counsel has informed me that I should consider these materials through the lens of a person of ordinary skill in the art ("POSITA," which is discussed further in §VII below) as of the Critical Date, and I have done so. Unless otherwise stated, my testimony below refers to the knowledge of a POSITA as of the Critical Date.

V. SUMMARY OF CONCLUSIONS

34. This Declaration explains the conclusions that I have formed based on my analysis. To summarize those conclusions:

Ground	Bases
1A	Claims 1-12 are obvious in view of Lamkin.
1B	Claims 1-12 are obvious in view of Lamkin and Fiechter.
1C	Claims 1-5 and 7-11 are obvious in view of Lamkin and Ito.
2A	Claims 1-12 are obvious in view of Franke.
2B	Claims 1-12 are obvious in view of Franke and Fiechter.
2C	Claims 1-5 and 7-11 are obvious in view of Franke and Ito.

VI. LEGAL PRINCIPLES

35. In forming my analysis and conclusions expressed in this Declaration, I have applied the legal principles described next, which were provided to me by counsel for the Petitioner.

A. Person of Ordinary Skill in the Art

36. I understand that the factors considered in determining the ordinary level of skill in a field of art include: the level of education and experience of persons

working in the field; the types of problems encountered in the field; the teachings of the prior art regarding solutions to such problems; and the sophistication of the technology at the time of the alleged invention's effective filing date. I understand that a person of ordinary skill in the art ("POSITA") is not a specific real individual, but rather is a hypothetical individual having the qualities reflected by the factors above and knowledge of all relevant prior art references, including the references I cite in my analysis in this Declaration.

B. Obviousness

37. I have been informed that a patent claim is invalid as "obvious" under 35 U.S.C. §103 in light of one or more prior art references if it would have been obvious to a POSITA, taking into account (1) the scope and content of the prior art, (2) the differences between the prior art and the claims, (3) the level of ordinary skill in the art, and (4) any so called "secondary considerations" of nonobviousness, which include: (i) "long felt need" for the claimed invention, (ii) commercial success attributable to the claimed invention, (iii) unexpected results of the claimed invention, and (iv) "copying" of the claimed invention by others.

38. I have been informed that a claim can be obvious in light of a single prior art reference or multiple prior art references. To be obvious in light of a single prior art reference or multiple prior art references, there must be a reason to modify the single prior art reference, or combine two or more references, in order to achieve

the claimed invention. This reason may come from a teaching, suggestion, or motivation to combine, or may come from the reference or references themselves, the knowledge or "common sense" of one skilled in the art, or from the nature of the problem to be solved, and may be explicit or implicit from the prior art as a whole. I have been informed that the combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results. I also understand it is improper to rely on hindsight in making the obviousness determination.

39. I understand that an obviousness determination also requires that a POSITA would have had a reasonable expectation of success. This concept has been explained to me as relating to the relative likelihood or predictability—from the perspective of a POSITA—of successfully supplementing and/or modifying the prior art in a manner that would meet the claimed limitations of the patent being challenged (here the '101 patent). I understand that the expectation of success need only be "reasonable" and, thus, does not require the absolute certainty gleaned from physically creating the proposed prior art modification.

C. Claim Construction Standard

40. It is my understanding that the claim construction standard includes construing the claim in accordance with the ordinary and customary meaning of such claim as understood by a person of ordinary skill in the art. As explained above and

further discussed below, I was at least a person of ordinary skill in the art before the earliest filing date of the challenged patent.

41. I understand that, for purposes of my analysis in this *inter partes* review proceeding and with respect to the prior art, IPRs are reviewed under the "*Phillips*" standard. I have been informed by counsel and I understand that, under the *Phillips* standard, claim terms are given their plain and ordinary meaning as understood by a person of ordinary skill in the art at the time of the invention in light of the claim language and the patent specification.

42. In that regard, I understand that the best indicator of claim meaning is its usage in the context of the patent specification as understood by a POSITA. I further understand that the words of the claims should be given their plain meaning unless that meaning is inconsistent with the patent specification or the patent's history of examination before the Patent Office.

43. I also understand that the words of the claims should be interpreted as they would have been interpreted by a POSITA at the time of the invention was made (not today). Because I do not know at what date the invention as claimed was made, I have used the earliest priority date of U.S. Patent No. 8,230,101 as the point in time for claim interpretation purposes. That date was March 2, 2007 ("the Critical Date").

VII. PERSON OF ORDINARY SKILL IN THE ART

A. A POSITA's Ordinary Level of Skill

44. Based on my knowledge and experience in the field and my review of the '101 patent and its file history, I believe that a person of ordinary skill in the art as of the Critical Date ("POSITA") would have had a bachelor's degree in computer science, computer engineering, or a comparable field, and about two to three years of professional experience working with networking and data storage architectures. Additional education could substitute for years or experience, and experience in the field could substitute for formal education. For example, a person with multiple years of experience in building, designing, and working with storage architectures might be a POSITA despite not having completed a formal education, because significant experience can substitute for education.

45. The POSITA described above would be capable of performing various design tasks related to systems of data storage and retrieval, and would understand the basic structural and software components required to implement such a system. The POSITA would have at least a basic knowledge of the history and development of data architectures including the implementation of data architectures in various network environments. From their education and/or experience, a POSITA would have a basic understanding of how to implement software or code including algorithms in a server device to receive, organize, and transfer data for storage,

receive instructions from one or more client devices, and act on such instructions to return requested data to a user device.

46. Accordingly, my analysis and conclusions expressed in this Declaration are based on the perspective of a POSITA having this level of knowledge and skill. My education, technical expertise and personal knowledge discussed in §III above shows that I meet the qualifications of a POSITA.

VIII. TECHNOLOGY OVERVIEW

47. As I describe next, it is my professional opinion that the claims of the '101 patent describe structures and functions that were well-known and conventional in tiered storage architectures by the Critical Date. In this section, I provide an overview of the background art as understood by a POSITA during the relevant timeframe.

A. General Computer Operations

48. Computers are hardware and software devices that execute instructions. The hardware often includes a central processing unit ("CPU") or other microprocessors, typically called "processors," required volatile memory (*e.g.*, RAM (random-access memory), or DRAM (dynamic random-access memory)), non-volatile (persistent) storage (*e.g.*, an internal hard disk or flash drive), and input/output ("I/O") devices (*e.g.*, network interface card, mouse, keyboard, display). RAM, hard disks, and flash drives are considered, generally, "storage" or

"memory" devices because they can store and hold bits of information (whether persistently or not). Computers can have multiple storage devices. Processors are coupled to main memory using a memory bus and coupled to peripherals such as hard disks and network interface cards using one or more I/O busses.

49. Memory and storage devices have considerable differences in their speeds, capacities, and costs. Non-volatile storage devices such as hard disks have the largest capacity, are least expensive per gigabyte, and are slowest (typically operating at millisecond speeds). Volatile DRAM devices are smaller, more expensive, and faster than hard disks (typically operating at microsecond speeds). A CPU's own memory caches and operations are the smallest, most expensive, and fastest (typically operating at nanosecond speeds). Overall, CPUs are about 1,000× faster than DRAM, which in turn is about 1,000× faster than hard disks.

50. Computer programs are often written in a high-level human-readable language (*e.g.*, C, C++, Java, Perl, PHP, Python), then translated using a compiler or script processor to machine instructions understood by the CPU (*e.g.*, Intel or AMD processor). Machine instructions—generally called "software"—are stored in files on persistent media (*e.g.*, HDD or ROM), loaded into DRAM, and then loaded into the CPU where they can be executed. All modern computers operate in this manner, regardless of their purpose: a small laptop, a server in a data center, a mainframe computer, a personal workstation, a proxy device, a gateway, a firewall,

a router, an appliance, a virtual machine, a handheld or mobile device, a smart card, a SD card, etc. Computer systems may run multiple software programs, from background servers and daemons to user applications.

51. Software itself may be distributed or located in a single location, use various modules or units such as libraries and other components. In general, "software" is "[c]omputer programs; instructions that make hardware work. Two main types of software are system software (operating systems), which controls the workings of the computer, and applications, such as word processing programs, spreadsheets, and databases, which perform the tasks for which people use computers. Two additional categories, which are neither system nor application software but contain elements of both, are network software, which enables groups of computers to communicate, and language software, which provides programmers with the tools they need to write programs. In addition to these task-based categories, several types of software are described based on their method of distribution. These include packaged software (canned programs), sold primarily through retail outlets; freeware and public domain software, which are distributed free of charge; shareware, which is also distributed free of charge, although users are requested to pay a small registration fee for continued use of the program; and vaporware, software that is announced by a company or individuals but either never makes it to market or is very late. *See also* application, canned software, freeware,

network software, operating system, shareware, system software, vaporware. *Compare* firmware, hardware, liveware." EX1017 (Microsoft Computer Dictionary), 81.

52. Users (including administrators) can login to one or more computers (*e.g.*, desktop, laptop, server) and execute one or more applications (*e.g.*, word processing, web browsing, backup tool). Some computers offer external interfaces and APIs to communicate with certain devices and especially peripherals. For example, both a USB storage device as well as a printer use a well-defined "protocol" to allow users to communicate with them through the main computer.

53. All computers include internal volatile memory (*e.g.*, RAM). A "primary storage" (or "internal memory") is "Random access memory (RAM); the main general-purpose storage region to which the microprocessor has direct access. A computer's other storage options, such as disks and tape, are called secondary storage or (sometimes) backing storage." EX1017 (Microsoft Computer Dictionary), 50, 64. A "secondary storage" is "[a]ny data storage medium other than a computer's random access memory (RAM)—typically tape or disk. *Compare* primary storage." EX1017 (Microsoft Computer Dictionary), 78.

54. Computers also include a dedicated, internal hard disk drive (HDD). This HDD is connected via a "ribbon cable," "[a] flat cable containing up to 100 parallel wires for data and control lines. For example, ribbon cables are used inside

a computer's case to connect the disk drives to their controllers." EX1017 (Microsoft Computer Dictionary), 73.

55. Some workstations have a dedicated storage device/location on the network, rather than connected internally. A "diskless workstation" is "[a] station on a computer network that is not equipped with a disk drive and that uses files stored in a file server. *See also* file server." EX1017 (Microsoft Computer Dictionary), 23.

56. Here, a "file server" is "[a] file-storage device on a local area network that is accessible to all users on the network. Unlike a disk server, which appears to the user as a remote disk drive, a file server is a sophisticated device that not only stores files but manages them and maintains order as network users request files and make changes to them. To deal with the tasks of handling multiple—sometimes simultaneous—requests for files, a file server contains a processor and controlling software as well as a disk drive for storage. On local area networks, a file server is often a computer with a large hard disk that is dedicated only to the task of managing shared files. *Compare* disk server." EX1017 (Microsoft Computer Dictionary), 32-33.

57. Modern computers include internal memories and storage devices (disks), audio support (*e.g.*, speakers to play sounds and music), and displays. "A modern, general-purpose computer system consists of a CPU and a number of device

controllers that are connected through a common bus that provides access to shared memory (Figure 2.1). Each device controller is in charge of a specific type of device (for example, disk drives, audio devices, and video displays)." EX1018 (Silberschatz), 13-14 (Figure 2.1, reproduced below).

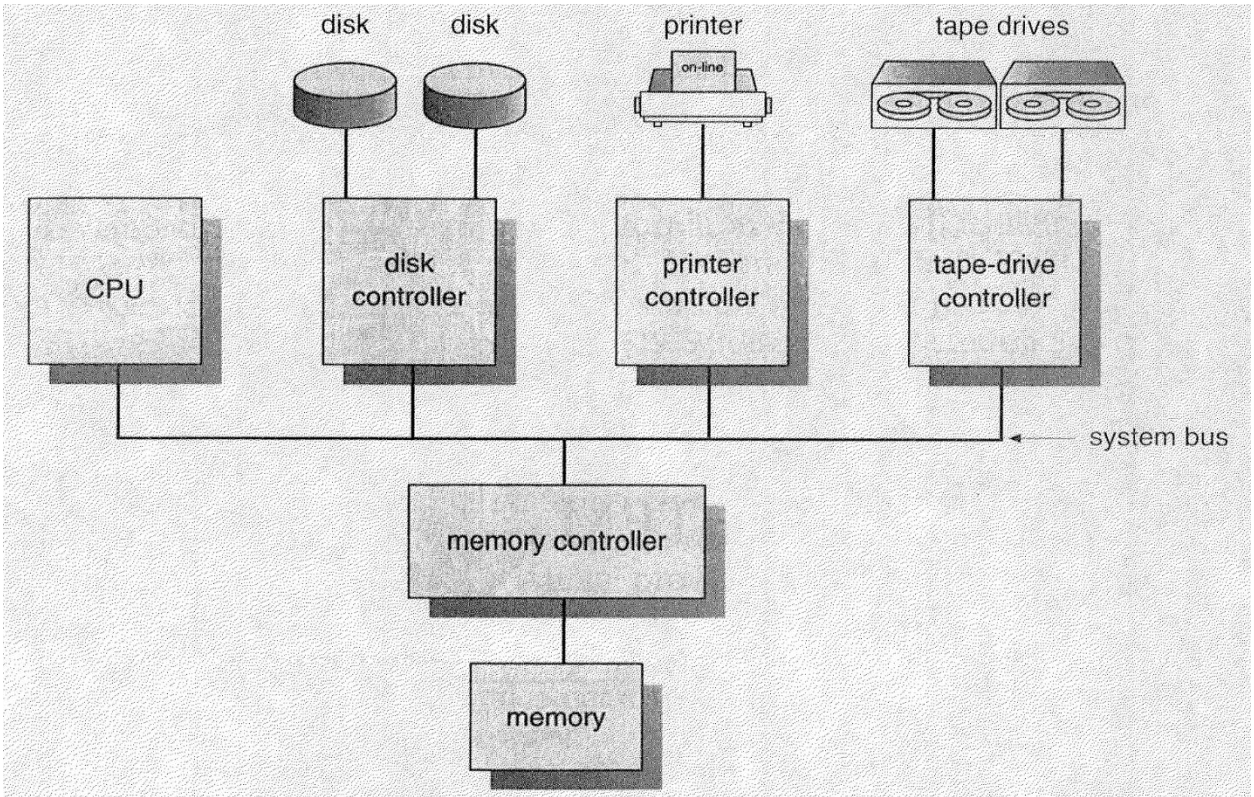


Figure 2.1 A modern computer system.

EX1018 (Silberschatz), FIG. 2.1

58. Computers can play sounds though a "sound card," "[a] type of expansion board on PC-compatible computers that allows the playback and recording of sound, such as from a WAV or MIDI file or a music CD-ROM. Most

PCs sold at retail include a sound card. *Also called*: sound board. *See also* expansion board, MIDI, WAV." EX1017 (Microsoft Computer Dictionary), 82.

59. One example of such software is Apple's QuickTime, which can play audio and video files. "QuickTime" is "[s]oftware components developed by Apple for creating, editing, publishing, and viewing multimedia content. QuickTime, which supports video, animation, graphics, 3-D, VR (virtual reality), MIDI, music, sound, and text, has been part of the Mac OS since version 7 of the operating system and is used in many Macintosh applications. Windows applications can also run QuickTime files but require the installation of special player software. QuickTime is often used on the Web to provide Web pages with video and animation. Most Web browsers support plug-ins for running these types of files. QuickTime is also part of the new MPEG-4 specification. *See also* MPEG-4." EX1017 (Microsoft Computer Dictionary), 70.

B. Networking

1. Networking Overview

60. A Local Area Network (LAN) is typically a network at a local site where network latencies are short and physical distances between computers are small. A LAN and its computers are often protected from the rest of the Internet using a firewall. EX1017 (Microsoft Computer Dictionary), 51 ("**LAN** *n*. Acronym for **l**ocal **a**rea **n**etwork. A group of computers and other devices dispersed over a

relatively limited area and connected by a communications link that enables any device to interact with any other on the network. LANs commonly include PCs and shared resources such as laser printers and large hard disks. The devices on a LAN are known as nodes, and the nodes are connected by cables through which messages are transmitted. *See also* baseband network, broadband network, bus network, collision detection, communications protocol, contention, CSMA/CD, network, peer-to-peer architecture, ring network, star network. *Compare* WAN.").

61. Another form of a local area network is one that operates inside a single computer. A "localhost" is "[t]he name that is used to represent the same computer on which a TCP/IP message originates. An IP packet sent to localhost has the IP address 127.0.0.1 and does not actually go out to the Internet. *See also* IP address, packet (definition 1), TCP/IP." EX1017 (Microsoft Computer Dictionary), 52. Often, when users/processes want to communicate with other users/processes, using TCP/IP communications on the same machine, they can use the localhost address: it is considered a trusted, high-speed internal network.

62. Conversely, a Wide-Area Network (WAN) refers to computers across the entire Internet, which are often distant from each other, exhibit longer latencies, and afford little protection. EX1017 (Microsoft Computer Dictionary), 93 (a "**wide area network**" is "[a] geographically widespread network, one that relies on communications capabilities to link the various network segments. A WAN can be

one large network, or it can consist of a number of linked LANs (local area networks).").

63. A firewall typically intercepts all traffic between computers in a LAN and the WAN, monitors and controls access, and can filter out undesired traffic, and thus acts as a proxy device. Often, LAN computers are far more trusted than WAN computers; therefore, a firewall can be said to separate the trusted network (LAN or localhost) from the untrusted network (WAN). EX1017 (Microsoft Computer Dictionary), 36-37 ("**firewall** *n*. A security system intended to protect an organization's network against external threats, such as hackers, coming from another network, such as the Internet. Usually a combination of hardware and software, a firewall prevents computers in the organization's network from communicating directly with computers external to the network and vice versa. Instead, all communication is routed through a proxy server outside of the organization's network, and the proxy server decides whether it is safe to let a particular message or file pass through to the organization's network. *See also* proxy server.").

64. A server is a computer or software that provides a service to other computers, called clients. There are many servers on the Internet, for example, Web servers and Mail servers. EX1017 (Microsoft Computer Dictionary), 80 ("**server** *n*. 1. On a local area network (LAN), a computer running administrative software that

controls access to the network and its resources, such as printers and disk drives, and provides resources to computers functioning as workstations on the network. 2. On the Internet or other network, a computer or program that responds to commands from a client. For example, a file server may contain an archive of data or program files; when a client submits a request for a file, the server transfers a copy of the file to the client. *See also* client/server architecture. *Compare* client (definition 3)."

65. A client is typically an end-point computer or software, sometimes used by a user, that communicates across LAN and WAN networks; a client may contact servers to request some service. A user's desktop, laptop, and handheld device are examples of clients. (Note that a computer can be both a client and a server, a form of a proxy.) EX1017 (Microsoft Computer Dictionary), 12 ("**client** *n*. [] 2. A process, such as a program or task, that requests a service provided by another program—for example, a word processor that calls on a sort routine built into another program. The client process uses the requested service without having to 'know' any working details about the other program or the service itself. *Compare* child (definition 1), descendant (definition 2). 3. On a local area network or the Internet, a computer that accesses shared network resources provided by another computer (called a *server*). *See also* client/server architecture, server.").

66. A client/server architecture describes how client computers interact with server computers. EX1017 (Microsoft Computer Dictionary), 12

("**client/server architecture** *n*. An arrangement used on LANs (local area networks) that makes use of distributed intelligence to treat both the server and the individual workstations as intelligent, programmable devices, thus exploiting the full computing power of each. This is done by splitting the processing of an application between two distinct components: a 'front-end' client and a 'back-end' server. The client component is a complete, stand-alone personal computer (not a 'dumb' terminal), and it offers the user its full range of power and features for running applications. The server component can be a personal computer, a minicomputer, or a mainframe that provides the traditional strengths offered by minicomputers and mainframes in a time-sharing environment: data management, information sharing between clients, and sophisticated network administration and security features. The client and server machines work together to accomplish the processing of the application being used. Not only does this increase the processing power available over older architectures but it also uses that power more efficiently. The client portion of the application is typically optimized for user interaction, whereas the server portion provides the centralized, multiuser functionality. *See also* distributed intelligence.").

67. An "agent" is "1. A program that performs a background task for a user and reports to the user when the task is done or some expected event has taken place. 2. A program that searches through archives or other repositories of information on

a topic specified by the user. Agents of this sort are used most often on the Internet and are generally dedicated to searching a single type of information repository, such as postings on Usenet groups. Spiders are a type of agent used on the Internet. *Also called* intelligent agent. *See also* spider. 3. In client/server applications, a process that mediates between the client and the server. 4. In Simple Network Management Protocol (SNMP), a program that monitors network traffic. *See also* SNMP." EX1017 (Microsoft Computer Dictionary), 4.

2. Network Failures and Mitigation

68. In networking, it is common that network transmissions are broken up to smaller units, sent to their destination, and then reassembled there. A "packet" is "1. A unit of information transmitted as a whole from one device to another on a network. 2. In packet-switching networks, a transmission unit of fixed maximum size that consists of binary digits representing both data and a header containing an identification number, source and destination addresses, and sometimes error-control data. *See also* packet switching." EX1017 (Microsoft Computer Dictionary), 60.

69. Indeed, fragmenting network (or any other) data for transmission is faster and more efficient, and the most common communication form on the Internet. "Packet switching" is "[a] message-delivery technique in which small units of information (packets) are relayed through stations in a computer network along the

best route available between the source and the destination. A packet-switching network handles information in small units, breaking long messages into multiple packets before routing. Although each packet may travel along a different path, and the packets composing a message may arrive at different times or out of sequence, the receiving computer reassembles the original message correctly. Packet-switching networks are considered to be fast and efficient. To manage the tasks of routing traffic and assembling/ disassembling packets, such a network requires some 'intelligence' from the computers and software that control delivery. The Internet is an example of a packet-switching network. Standards for packet switching on networks are documented in the International Telecommunication Union (ITU) recommendation X.25." EX1017 (Microsoft Computer Dictionary), 61.

70. TCP is a popular transport protocol. "TCP" is "Acronym for **T**ransmission **C**ontrol **P**rotocol. The protocol within TCP/IP that governs the breakup of data messages into packets to be sent via IP (internet Protocol), and the reassembly and verification of the complete messages from packets received by IP. A connection-oriented, reliable protocol (reliable in the sense of ensuring error-free delivery), TCP corresponds to the transport layer in the ISO/OSI reference model. *See also* ISO/OSI model, packet, TCP/IP. *Compare* UDP." EX1017 (Microsoft Computer Dictionary), 87.

71. Packets can be lost or dropped for many reasons, but most often because of network congestion. "Congestion" is "[t]he condition of a network when the current load approaches or exceeds the available resources and bandwidth designed to handle that load at a particular location in the network. Packet loss and delays are associated with congestion." EX1017 (Microsoft Computer Dictionary), 14.

72. Another reason why packets could be lost is a "collision," which is "[t]he result of two devices or network workstations trying to transmit signals at the exact same time on the same channel. The typical outcome is a garbled transmission." EX1017 (Microsoft Computer Dictionary), 13.

73. More advanced techniques have been devised to mitigate packet loss. One is "traffic shaping," "[a] technique for allocating bandwidth and preventing packet loss by enforcing prioritization policies on the transmission of data over a network. *Also called*: bandwidth shaping. *See also* bandwidth management, bandwidth reservation, token passing." EX1017 (Microsoft Computer Dictionary), 89.

74. It is common on the Internet to include multiple methods to detect or avoid network collisions (*e.g.*, predict network failures before they happen, and retransmit lost packets as needed). A "collision detection" technique is "1. The process by which a node on a local area network monitors the communications line to determine when a collision has occurred; that is, when two nodes have attempted

to transmit at the same time. Although network stations usually avoid collisions by monitoring the line and waiting for it to clear before transmitting, the method is not foolproof. When a collision does occur, the two nodes involved usually wait a random amount of time before attempting to retransmit. *See also* contention, CSMA/CD. []" EX1017 (Microsoft Computer Dictionary), 13.

75. "CSMA/CA" is "[a]cronym for **C**arrier **S**ense **M**ultiple **A**ccess with **C**ollision **A**voidance, a protocol for controlling network access similar to CSMA/CD, in that nodes (stations) listen to the network and transmit only when it is free. But in CSMA/CA, nodes avoid data collisions by signaling their intention with a brief Request to Send (RTS) signal and then waiting for acknowledgment before actually transmitting." EX1017 (Microsoft Computer Dictionary), 18.

76. Similarly, "CSMA/CD" is "[a]cronym for **C**arrier **S**ense **M**ultiple **A**ccess with **C**ollision **D**etection. A network protocol for handling situations in which two or more nodes (stations) transmit at the same time, thus causing a collision. With CSMA/CD, each node on the network monitors the line and transmits when it senses that the line is not busy. If a collision occurs because another node is using the same opportunity to transmit, both nodes stop transmitting. To avoid another collision, both then wait for differing random amounts of time before attempting to transmit again. *Compare* token passing." EX1017 (Microsoft Computer Dictionary), 18-19.

3. Network Protocols

77. To be useful, computer systems are connected using a worldwide network known as the Internet or other networks including wired and wireless networks. Without being able to offer services over the network or access networked services, the Internet would not have become so pervasive in our lives. Indeed, one of the early pioneers in the field of networking, a then fledgling startup named Sun Microsystems (now acquired by Oracle), coined the visionary marketing phrase "the network *is* the computer."

78. UDP (User Datagram Protocol) is a connectionless transport layer protocol providing no reliability. *See* EX1020 (Stevens), 12 ("UDP provides no reliability: it sends the datagrams that the application writes to the IP layer, but there is no guarantee that they ever reach their destination."). This means that the packets could be lost, reordered, or duplicated on their way to a destination host, and the sender would have no way of knowing this. Senders using UDP typically have to arrange other means to detect and retransmit lost packets.

79. Network transmissions are fundamentally unreliable, requiring an additional mechanism to make them more reliable. Indeed, the guiding principle taught to Computer Science students is that network systems are designed for delivering packets under a "best effort" regime: trying as best as possible to deliver the packets, but if not possible, then letting higher layers of abstractions discover the

network problems and recover from it. For example, an "unreliable protocol" is "[a] communications protocol that makes a 'best effort' attempt to deliver a transmission but does not provide for verifying that the transmission arrives without error." EX1017 (Microsoft Computer Dictionary), 92.

80. TCP is a connection-oriented transport protocol, layered on top of the connectionless IP protocol. *See* EX1020 (Stevens), 14 ("Even though TCP and UDP use the same network layer (IP), TCP provides a totally different service to the application layer than UDP does. TCP provides a connection-oriented, reliable, byte stream service. The term *connection-oriented* means the two applications using TCP (normally considered a client and a server) must establish a TCP connection with each other before they can exchange data."). For example, TCP tracks all packets to ensure that lost packets are retransmitted and acknowledged, to ensure reliable connections between the two TCP endpoints.

81. Proxies, bridges, and routers share much in common: they receive network packets on one end, process them, then transmit them on another end: their difference often boils down to which OSI network layer they operate at.

82. A "router" is "[a]n intermediary device on a communications network that expedites message delivery. On a single network linking many computers through a mesh of possible connections, a router receives transmitted messages and forwards them to their correct destinations over the most efficient available route.

On an interconnected set of LANs (local area networks)—including those based on differing architectures and protocols—using the same communications protocols, a router serves the somewhat different function of acting as a link between LANs, enabling messages to be sent from one to another. *See also* bridge, gateway." EX1017 (Microsoft Computer Dictionary), 75.

83. A "bridge" is "[a] device that connects networks using the same communications protocols so that information can be passed from one to the other. *Compare* gateway. 2. A device that connects two LANs (local area networks), whether or not they use the same protocols, and allows information to flow between them. The bridge operates at the ISO/OSI data-link layer. *Also called*: layer switch. *See also* data-link layer. *Compare* router." EX1017 (Microsoft Computer Dictionary), 10.

84. A "bridge router" is "[a] device that supports the functions of both a bridge and router. A bridge router links two segments of a local or wide area network, passing packets of data between the segments as necessary, and uses Level 2 addresses for routing. *Also called* Brouter. *See also* bridge (definition 2), router." EX1017 (Microsoft Computer Dictionary), 10.

85. A "gateway" is "[a] device that connects networks using different communications protocols so that information can be passed from one to the other. A gateway both transfers information and converts it to a form compatible with the

protocols used by the receiving network. *Compare* bridge." EX1017 (Microsoft Computer Dictionary), 43.

86. A "proxy server" is "[a] firewall component that manages Internet traffic to and from a local area network (LAN) and can provide other features, such as document caching and access control. A proxy server can improve performance by supplying frequently requested data, such as a popular Web page, and can filter and discard requests that the owner does not consider appropriate, such as requests for unauthorized access to proprietary files. *See also* firewall." EX1017 (Microsoft Computer Dictionary), 66.

87. Over the years, many firewalls, proxies, routers, network devices, and security gateways were developed. Earlier ones tended to operate at one or a few of the lower OSI network layers and applied simpler stateless rules; subsequent firewalls were stateful and understood the legitimate life cycle of a protocol (*e.g.*, TCP's 3-way handshaking). *See* EX1021 (Cheswick), 6-7 ("**Packet-Filtering Gateways**… Packet filters can provide a cheap and useful level of gateway security. Used by themselves, they are cheap: the filtering abilities come with the router software. Since you probably need a router to connect to the Internet in the first place, there is no extra charge."), 7 ("Packet filters work by dropping packets based on their source or destination addresses or ports… Most security policies require finer control than this: they need to define access to specific services for hosts that

are otherwise untrusted."), 8 ("**Filtering Other Protocols**… Other protocols are layered on top of IP as well; depending on your environment, these may need to be filtered also. Of particular import is ICMP, the Internet Control Message Protocol."), 9 ("Many advanced gateway designs rely in part on packet filtering. They are likely to work well, but pure packet filters leave us feeling uncomfortable.").

88. As the field quickly matured, more advanced gateways began to operate at the application layer (*e.g.*, anti-virus, URL filtering). *See* EX1021 (Cheswick), 9 ("**Application-Level Gateways**… An application-level gateway represents the opposite extreme in firewall design. Rather than using a general-purpose mechanism to allow many different kinds of traffic to flow, special-purpose code can be used for each desired application. Although this seems wasteful, it is likely to be far more secure than any of the alternatives… Application gateways have another advantage that in some environments is quite critical: it is easy to log and control *all* incoming and outgoing traffic.").

89. Principles of network quality of service (QoS, sometimes called traffic shaping) were well known in the 1980's, including explicit bandwidth control and reservations. *See* EX1019 (Tanenbaum), 18 ("The *qos* parameter is actually two lists of values that determine the quality of service provided by the connection. The first list gives the goal—what the caller really wants. The second list gives the

minimum values considered acceptable. If the network service is unable to provide at least the minimum value for any parameter specified by either the caller (calling party) or the callee (called party), the connection establishment fails. The values that may be specified include throughput, delay, error rate, secrecy, and cost, among others.").

C. Data Storage Devices, Files, and File Systems

90. As mentioned in §VIII.A above, computers often include two types of storage or memory devices. The first type is a volatile memory device, called Random Access Memory ("RAM"), that is often used to hold executing programs and their data. EX1017 (Microsoft Computer Dictionary), 71 ("RAM").

91. The second type is a non-volatile, or persistent storage device, such as a hard disk drive ("HDD"). HDDs are used to store programs and data persistently. EX1017 (Microsoft Computer Dictionary), 44 ("hard disk").

92. HDDs are electro-mechanical devices with moving parts. Computers can also have other types of non-volatile storage devices that have no moving parts, also called "solid-state devices," such as flash memories. EX1017 (Microsoft Computer Dictionary), 38 ("flash memory").

93. Moreover, any single device (*e.g.*, a hard disk) can be logically broken down into several units called "partitions." EX1017 (Microsoft Computer Dictionary), 24 ("**disk partition** *n.* A logical compartment on a physical disk drive.

A single disk might have two or more logical disk partitions, each of which would be referenced with a different disk drive name. Multiple partitions are divided into a primary (boot) partition and one or more extended partitions.").

94. Above the storage media often lives a logical software layer called a "file system." File systems provide a convenient abstraction to users and applications of files, folders or directories, and naming. EX1017 (Microsoft Computer Dictionary), 34 ("**file system** *n.* In an operating system, the overall structure in which files are named, stored, and organized. A file system consists of files, directories, or folders, and the information needed to locate and access these items. The term can also refer to the portion of an operating system that translates requests for file operations from an application program into low-level, sector-oriented tasks that can be understood by the drivers controlling the disk drives. *See also* driver.").

95. A file is a collection of data and attributes, or meta-data, about the file, including the file's name and extension. EX1017 (Microsoft Computer Dictionary), 31 ("**file** *n.* A complete, named collection of information, such as a program, a set of data used by a program, or a user-created document. A file is the basic unit of storage that enables a computer to distinguish one set of information from another. A file is the 'glue' that binds a conglomeration of instructions, numbers, words, or images

into a coherent unit that a user can retrieve, change, delete, save, or send to an output device.").

96. Files are broken up into units called blocks or sectors, stored on storage media such as hard disks. A "block" is "5. A collection of consecutive bytes of data that are read from or written to a device (such as a disk) as a group." EX1017 (Microsoft Computer Dictionary), 8 (noun definition 5).

97. When files are transmitted to local or network storage, they are usually broken up into block-size units. To "block" is "1. To distribute a file over fixed-size blocks in storage. []." EX1017 (Microsoft Computer Dictionary), 8-9.

98. Hard disks are organized as fixed-size units, so files are broken into and stored as one or more of these units (blocks or sectors). A "sector" is [a] portion of the data storage area on a disk. A disk is divided into sides (top and bottom), tracks (rings on each surface), and sectors (sections of each ring). Sectors are the smallest physical storage units on a disk and are of fixed size; typically, they are capable of holding 512 bytes of information apiece. []." EX1017 (Microsoft Computer Dictionary), 79.

99. File attributes are also called file meta-data and include security attributes, the name (title), size of a file, timestamps, and more. EX1017 (Microsoft Computer Dictionary), 53 ("**meta data** or **metadata** *n.* 1. Data about data. For example, the title, subject, author, and size of a file constitute metadata about the

file. *See also* data dictionary, repository."), 31 ("**file attribute** *n.* A restrictive label attached to a file that describes and regulates its use—for example, hidden, system, read-only, archive, and so forth. In MS-DOS, this information is stored as part of the file's directory entry.").

100. A file's name is an important attribute of a file and conveys a lot of useful information about the file's meaning and purpose. EX1017 (Microsoft Computer Dictionary), 32 ("**file name** *n.* The set of letters, numbers, and allowable symbols assigned to a file to distinguish it from all other files in a particular directory on a disk. A file name is the label under which a computer user saves and requests a block of information. Both programs and data have file names and often extensions that further identify the type or purpose of the file. Naming conventions, such as maximum length and allowable characters of a file name, vary from one operating system to another. *See also* directory, path (definition 5).").

101. A file's path or path-name conveys the location of a file in the name hierarchy and even which file server the file may reside on (*e.g.*, the local C:\ drive or a network drive like N:\). EX1017 (Microsoft Computer Dictionary), 62 ("**path** *n.* […] 5. In file storage, the route followed by the operating system through the directories in finding, sorting, and retrieving files on a disk."), *id.* ("**pathname** *n.* In a hierarchical filing system, a listing of the directories or folders that lead from the current directory to a file. *Also called:* directory path.").

102. Moreover, a file's extension conveys vital information about the file's type and hence its meaning and purpose. For example, a file ending with "xls" is known to be a Microsoft Excel spreadsheet, files with "doc" extensions are Microsoft Word files, and those ending with "html" are Web server hyper-text markup language files. EX1017 (Microsoft Computer Dictionary), 31 ("**file extension** *n.* *See* extension (definition 1)."), p. 32 ("**filename extension** *n.* *See* extension (definition 1)."), p. 29 ("**extension** *n.* 1. A set of characters added to a filename that serves to extend or clarify its meaning or to identify a file as a member of a category. An extension may be assigned by the user or by a program, as, for example, .com or .exe for executable programs that MS-DOS can load and run.").

103. The (usually non-volatile) storage media that holds the actual bits of file data is often called the "secondary storage" or the "backing storage" (because it is in the "back end"); this is different than "backup storage" that's often used to backup files for disaster recovery purposes. A "primary storage" (or "internal memory") is "[r]andom access memory (RAM); the main general-purpose storage region to which the microprocessor has direct access. A computer's other storage options, such as disks and tape, are called *secondary storage* or (sometimes) *backing storage*." EX1017 (Microsoft Computer Dictionary), 64, 50. A "secondary storage" is "[a]ny data storage medium other than a computer's random access memory

(RAM)—typically tape or disk. *Compare* primary storage." EX1017 (Microsoft Computer Dictionary), 78.

1. Hierarchical File Systems

104. Above the storage media often lives a logical software layer called a "file system." File systems provide a convenient abstraction to users and applications of files, folders or directories, and naming. EX1017 (Microsoft Computer Dictionary), 34 ("**file system** *n.* In an operating system, the overall structure in which files are named, stored, and organized. A file system consists of files, directories, or folders, and the information needed to locate and access these items. The term can also refer to the portion of an operating system that translates requests for file operations from an application program into low-level, sector-oriented tasks that can be understood by the drivers controlling the disk drives. *See also* driver.").

105. Next, before any storage media or partition can be used to access files and folders, it must be formatted. To "format" is "2. To prepare a disk for use by organizing its storage space into a collection of data 'compartments,' each of which can be located by the operating system so that data can be sorted and retrieved. When a previously used disk is formatted, any preexisting information on it is lost." EX1017 (Microsoft Computer Dictionary), 41 ("Format" definition 2).

106. A "pathname," "[i]n a hierarchical filing system, a listing of the directories or folders that lead from the current directory to a file. *Also called*: directory path." EX1017 (Microsoft Computer Dictionary), 62.

107. All common and modern file systems include directories (folders). A "directory" is "1. A catalog for filenames and other directories stored on a disk. A directory is a way of organizing and grouping the files so that the user is not overwhelmed by a long list of them. The uppermost directory is called the *root directory*; the directories within a directory are called *subdirectories*. Depending on how an operating system supports directories, filenames in a directory can be viewed and ordered in various ways—for example, alphabetically, by date, by size, or as icons in a graphical user interface. What the user views as a directory is supported in the operating system by tables of data, stored on the disk, that indicate characteristics and the location of each file. In the Macintosh and Windows 9x operating systems, directories are called *folders*. 2. On a network, an index of names and pertinent information related to authorized users and network resources." EX1017 (Microsoft Computer Dictionary), 21.

108. File systems can have a "flat file directory" comprising a list of file names: "[a] directory that cannot contain sub-directories but simply contains a list of file names. *Compare* hierarchical file system." EX1017 (Microsoft Computer Dictionary), 39.

109. It was well known that applications, including network ones, could transmit and receive lists of files as well as actual data files. One popular example is "FTP," an "1. [a]cronym for **F**ile **T**ransfer **P**rotocol, a fast, application-level protocol widely used for copying files to and from remote computer systems on a network using TCP/IP, such as the Internet. This protocol also allows users to use FTP commands to work with files, such as listing files and directories on the remote system. *See also* TCP/IP. []" EX1017 (Microsoft Computer Dictionary), 42 (noun definition 1).

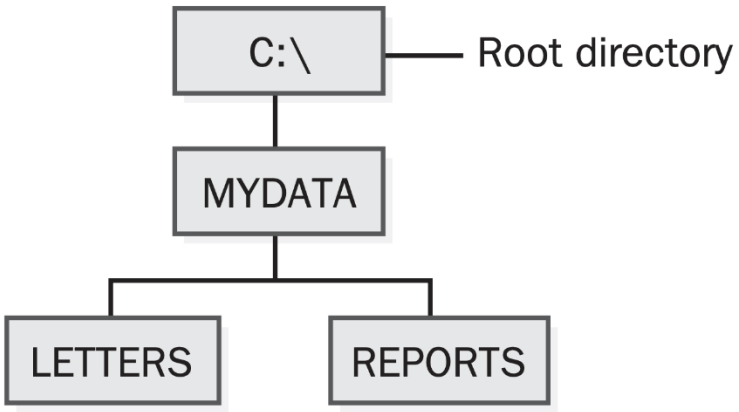
110. Modern file systems are hierarchical, using a tree structure of folders (directories) and files. A "hierarchical file system" is "[a] system for organizing files on a disk in which files are contained in directories or folders, each of which can contain other directories as well as files. The main directory for the disk is called the root; the chain of directories from the root to a particular file is called the path. *See also* hierarchy, path (definition 2), root. *Compare* flat file system." EX1017 (Microsoft Computer Dictionary), 45.

111. Microsoft's FAT file system and NTFS are examples of hierarchical file systems. *Id*., 30, 59. The "FAT file system" is "[t]he system used by MS-DOS to organize and manage files. The FAT (file allocation table) is a data structure that MS-DOS creates on the disk when the disk is formatted. When MS-DOS stores a file on a formatted disk, the operating system places information about the stored

file in the FAT so that MS-DOS can retrieve the file later when requested. The FAT is the only file system MS-DOS can use; OS/2, Windows NT, and Windows 9x operating systems can use the FAT file system in addition to their own file systems (HPFS, NTFS, and VFAT, respectively). *See also* file allocation table, HPFS, NTFS, OS/2, VFAT, Windows."). EX1017 (Microsoft Computer Dictionary), 30.

112. Apple's "Hierarchical File System" (HFS) is another example; it is "[a] tree-structured file system used on the Apple Macintosh in which folders can be nested within other folders. *Acronym*: HFS. *See also* hierarchy, path (definition 2), root. *Compare* flat file system." EX1017 (Microsoft Computer Dictionary), 45.

113. The "root directory" of a file system is "[t]he point of entry into the directory tree in a disk-based hierarchical directory structure. Branching from this root are various directories and subdirectories, each of which can contain one or more files and subdirectories of its own. For example, in the MS-DOS operating system the root directory is identified by a name consisting of a single backslash character (\). Beneath the root are other directories, which may contain further directories, and so on. See the illustration." EX1017 (Microsoft Computer Dictionary), 74 (cited illustration reproduced below).



EX1017 (Microsoft Computer Dictionary), FIG. reproduced from page 457

114. A file system's "root folder" is "[t]he folder on a drive from which all other folders branch. The root folder's name consists of a single backslash character (\). For example, on drive C, this folder would be represented in the file system as C:\." EX1017 (Microsoft Computer Dictionary), 74.

115. Indeed, a hierarchical file system view is nothing more than a tree-based view. A "tree view" is "[a] hierarchical representation of the folders, files, disk drives, and other resources connected to a computer or network. For example, Windows Explorer uses a tree view to display the resources that are attached to a computer or a network. *See also* resource." EX1017 (Microsoft Computer Dictionary), 90.

D. Networked and Distributed Storage Systems

116. As discussed in §VIII.C above, storage devices provide raw storage to computers. One of the most popular storage device protocols is "SCSI," which is an "[a]cronym for **S**mall **C**omputer **S**ystem **I**nterface, a standard high-speed parallel

interface defined by the X3T9.2 committee of the American National Standards Institute (ANSI). A SCSI (pronounced 'scuzzy') interface is used to connect microcomputers to SCSI peripheral devices, such as many hard disks and printers, and to other computers and local area networks. *Also called:* SCSI-1, SCSI I. *Compare* ESDI, IDE." EX1017 (Microsoft Computer Dictionary), 77.

117. A "SCSI network" is "[a] set of devices on a SCSI bus, which acts like a local area network. *See also* SCSI." EX1017 (Microsoft Computer Dictionary), 77.

118. File systems can appear as local volumes (*e.g.*, the C:\ drive in Windows), but they can also be networked, where the physical storage resides on a remote file server. Network file systems allow files and volumes to be shared and accessible from multiple networked computers. EX1017 (Microsoft Computer Dictionary), 57 ("**Network File System** *n.* *See* NFS."), 58 ("**NFS** *n.* Acronym for **N**etwork **F**ile **S**ystem. A distributed file system that allows users to access remote files and directories on a network as if they were local. NFS is compatible with Microsoft Windows and UNIX-based systems, including Linux and Mac OS X.").

119. NFS is an example of a network-attached storage, used to hold, transmit, and stream any files. A "NAS" is "[a]cronym for **N**etwork-**A**ttached **S**torage. A platform-independent storage appliance connected to a network. NAS uses a storage unit with a built-in server that can communicate with clients over a

network. NAS devices are popular for ease of maintenance, manageability, and scalability. *Compare* SAN." EX1017 (Microsoft Computer Dictionary), 55, 56.

120. Using such network protocols, users and applications can transmit and receive data and files, also known as pushing and pulling data/files. To "push" is "[] 2. In networks and the Internet, to send data or a program from a server to a client at the instigation of the server. *See also* push[2]. *Compare* pull." EX1017 (Microsoft Computer Dictionary), 69 (verb definition 2). Conversely, a "pull" is "[t]he process of retrieving data from a network server. *Compare* push (definition 2). *See* pop." EX1017 (Microsoft Computer Dictionary), 68.

121. For larger, more demanding network storage systems, a storage-area network is used. A "storage area network" or "SAN" is "[a] high-speed network that provides a direct connection between servers and storage, including shared storage, clusters, and disaster-recovery devices. A storage area network, or SAN, includes components such as hubs and routers that are also used in local area networks (LANs), but it differs in being something of a 'subnetwork' dedicated to providing a high-speed connection between storage elements and servers. Most SANs rely on fiber-channel connections that deliver speeds up to 1000 Mbps and can support up to 128 devices. SANs are implemented to provide the scalability, speed, and manageability required in environments that demand high data

availability. *Acronym*: SAN. *Also called*: system area network." EX1017 (Microsoft Computer Dictionary), 83-84.

122. A NAS, NFS, or SAN storage can be used by any computer over a network. Such network storage is designed for sharing, or for multiple computers to access the same network storage concurrently. However, it is also possible for such network storage or portion of it to be dedicated for user by a single computer. In that case, such network storage acts similarly to a dedicated internal storage such as an internal hard disk.

123. When the amount of data to access grew large enough, and especially if the data had to be concurrently accessed from multiple users/applications, storage researchers and companies leveraged the principles of distributed systems to improve the performance, resiliency, and fault-tolerance of the storage system.

124. A single computer's performance is limited by the physical maximum speeds of its network, processor, memory, and I/O devices. A distributed system, however, improves overall performance by using multiple computers concurrently to process many requests, thus increasing the aggerate throughput manyfold.

125. A single computer also represents a single point of failure: if that one computer is down, off, or broken into, then the service it offers is unavailable. Even regular networks can fail and packets get lost. A distributed system, conversely, improves resiliency and security by ensuring that the failure of any one (or even a

few) components do not affect the overall service provided. This principle is called "fault tolerance." EX1017 (Microsoft Computer Dictionary), 30 ("**fault tolerance** *n.* The ability of a computer or an operating system to respond to a catastrophic event or fault, such as a power outage or a hardware failure, in a way that ensures that no data is lost and any work in progress is not corrupted. This can be accomplished with a battery-backed power supply, backup hardware, provisions in the operating system, or any combination of these. In a fault-tolerant network, the system has the ability either to continue the system's operation without loss of data or to shut the system down and restart it, recovering all processing that was in progress when the fault occurred.").

126. NFS is but one example of a networked or distributed file system. Many complex software systems have been designed to operate in a distributed manner, where they are more resilient to failures and scale better than software running on a single node.

127. Generally, a "distributed network" is "[a] network in which processing, storage, and other functions are handled by separate units (nodes) rather than by a single main computer." EX1017 (Microsoft Computer Dictionary), 26.

128. The "Distributed Computing Environment" is "[a] set of standards from the Open Group (formerly the Open Software Foundation) for development of

distributed applications that can operate on more than one platform. *Acronym*: DCE. *See also* distributed processing." EX1017 (Microsoft Computer Dictionary), 25.

129. "[D]istributed processing" is "[a] form of information processing in which work is performed by separate computers linked through a communications network. Distributed processing is usually categorized as either plain distributed processing or true distributed processing. Plain distributed processing shares the workload among computers that can communicate with one another. True distributed processing has separate computers perform different tasks in such a way that their combined work can contribute to a larger goal. The latter type of processing requires a highly structured environment that allows hardware and software to communicate, share resources, and exchange information freely." EX1017 (Microsoft Computer Dictionary), 26.

130. Finally, a "distributed file system" is "[a] file management system in which files may be located on multiple computers connected over a local or wide area network." EX1017 (Microsoft Computer Dictionary), 26.

1. Multi-Tier and Caching Architectures

131. As soon as computer networking grew in popularity, a new computing paradigm emerged where central servers would provide common services to many clients (endpoints). A "two-tier client/server" is "[a] client/business logic layer and the database layer. Fourth-generation languages (4GL) have helped to popularize

the two-tier client/server architecture. *Compare* three-tier client/server." EX1017 (Microsoft Computer Dictionary), 91.

132. The two-tier client/server model grew to include additional computing tiers. For example, a Web browser would communicate with a Web server (*e.g.*, Apache), which in turn would communicate with an application server, the latter communicates with a database server (*e.g.*, MySQL or Oracle).

133. A "multi-tier" or "three-tier client/server" is "[a] client/server architecture in which software systems are structured into three tiers or layers: the user interface layer, the business logic layer, and the database layer. Layers may have one or more components. For example, there can be one or more user interfaces in the top tier, each user interface may communicate with more than one application in the middle tier at the same time, and the applications in the middle tier may use more than one database at a time. Components in a tier may run on a computer that is separate from the other tiers, communicating with the other components over a network. *See also* client/server architecture. *Compare* two-tier client/server." EX1017 (Microsoft Computer Dictionary), 54, 88.

134. The intermediate nodes or servers in a multi-tier architecture are sometimes called proxy because of their role; proxies can act as both clients and servers, and in both directions. A "proxy" is "[a] computer (or the software that runs on it) that acts as a barrier between a network and the Internet by presenting only a

single network address to external sites. By acting as a go-between representing all internal computers, the proxy protects network identities while still providing access to the Internet. *See also* proxy server." EX1017 (Microsoft Computer Dictionary), 65.

135. A "proxy server" is, for example, "[a] firewall component that manages Internet traffic to and from a local area network (LAN) and can provide other features, such as document caching and access control. A proxy server can improve performance by supplying frequently requested data, such as a popular Web page, and can filter and discard requests that the owner does not consider appropriate, such as requests for unauthorized access to proprietary files. *See also* firewall." EX1017 (Microsoft Computer Dictionary), 65.

136. Proxies often reside closer to clients than servers and hence are useful to cache (*e.g.*, Web) data for faster access. "ICP" is "[a]cronym for **I**nternet **C**ache **P**rotocol. A networking protocol used by cache servers to locate specific Web objects in neighboring caches. Typically implemented over UDP, ICP also can be used for cache selection. ICP was developed for the Harvest research project at the University of Southern California. It has been implemented in SQUID and other Web proxy caches." EX1017 (Microsoft Computer Dictionary), 49.

137. Proxies and multi-tier architectures can be applied to more than just Web technologies, for example for data storage and backups. "HSM" is "[s]hort for

Hierarchical **S**torage **M**anagement. A technology for managing online data and data storage in which the medium on which the information resides is linked to the frequency with which the information is accessed. By migrating data to and from primary (rapidly accessed but expensive) and secondary (slower but less expensive) storage, HSM maintains often-used information on primary storage media and less frequently used data on secondary storage such as tape or an optical jukebox. Although information resides on different storage media, all of it appears to be on line and remains accessible to the user. When users request data residing on secondary storage, HSM moves the information back to the primary storage medium." EX1017 (Microsoft Computer Dictionary), 47, 45. HSM systems, therefore, can automatically manage storage by migrating it between locations to, for example, free up space.

E. Multi-Media Streaming

138. As discussed in §VIII.C above, files are fundamentally stored as a sequence of blocks or sectors. This allows transmitting the files over the networks more efficiently, one data unit at a time. Such transmission techniques enabled streaming multi-media content over the Internet.

139. A "stream" represents "[a]ny data transmission, such as the movement of a file between disk and memory, that occurs in a continuous flow. Manipulating a data stream is a programming task. Consumers, however, are likely to encounter

references to streams and streaming in connection to the Internet, which has increased reliance on stream techniques to enable users (even those with slower equipment) to access large multimedia files—especially those containing audio and video components—and to display or play them before all the data has been transferred." EX1017 (Microsoft Computer Dictionary), 84 (noun, definition 1).

140. To "stream" is "[t]o transfer data continuously, beginning to end, in a steady flow. Many aspects of computing rely on the ability to stream data: file input and output, for example, and communications. If necessary, an application receiving a stream must be able to save the information to a buffer in order to prevent loss of data. On the Internet, streaming enables users to begin accessing and using a file before it has been transmitted in its entirety." EX1017 (Microsoft Computer Dictionary), 84 (defining "stream" (verb, definition 2)).

141. Streaming data can be over networks or storage media. For example, "streaming" is, "1. On the Internet, the process of delivering information, especially multimedia sound or video, in a steady flow that the recipient can access as the file is being transmitted. 2. In magnetic tape storage devices, a low-cost technique to control the motion of the tape by removing tape buffers. Although streaming tape compromises start/stop performance, it achieves highly reliable storage and retrieval of data, and is useful when a steady supply of data is required by a particular application or computer." EX1017 (Microsoft Computer Dictionary), 84.

142. Due to the growing popularity of the Internet, in the 1990s, several efforts were made to standardize streaming formats.

143. The "Advanced Streaming Format" is "[a]n open file format specification for streaming multimedia files containing text, graphics, sound, video, and animation. Advanced Streaming Format (ASF) does not define the format for any media streams within the file. Rather, it defines a standardized, extensible file 'container' that is not dependent on a particular operating system or communication protocol, or on a particular method (such as HTML or MPEG-4) used to compose the data stream in the file. An ASF file consists of three objects: a Header object containing information about the file itself, a Data object containing the media streams, and an optional Index object that can help support random access to data within the file. The ASF specification has been submitted to the ISO (International Organization for Standardization) for consideration. *Acronym*: ASF. *See also* streaming." EX1017 (Microsoft Computer Dictionary), 3.

144. Of course, the world-wide-web was the natural platform to deliver multi-media content, allowing media files to be played even while they were only partially downloaded. "HTTP streaming" is "[t]he process of downloading streaming digital media using an HTTP server (a standard Internet server) rather than a server designed specifically to transmit streaming media. HTTP streaming downloads the media file onto a computer, which plays the downloaded file as it

becomes available. *See also* real-time streaming." EX1017 (Microsoft Computer Dictionary), 48.

145. Streaming content can be done in two ways. First, when the content is being recorded live (*e.g.*, a Superbowl game or Presidential debate), then the transmission must be transmitted and received in real time. For example, "real-time streaming" is "[t]he process of delivering a streaming media file via a specialized streaming media server using real-time streaming protocol (RTSP). With real-time streaming, the file itself actually plays on the streaming media server, even though it is viewed on the computer that opened the file. Real-time streaming transmits at a higher bandwidth than HTTP streaming. It is often used to broadcast live events, such as concerts or keynote conference addresses. *See also* HTTP streaming." EX1017 (Microsoft Computer Dictionary), 72.

146. Second, when streaming pre-recorded content, then the content can be partially (or even fully) downloaded and then begins to play for the user. Here, "pseudo-streaming" is "[a] method used for real-time display of audio and video over the Web. Unlike sound or video files that are downloaded to a computer in their entirety before they can be played back, pseudo-streaming enables playback after only a portion of the file—enough to fill a buffer on the receiving computer—has been downloaded. Pseudo-streaming, unlike 'true,' or Web, streaming, does not depend on server software to dynamically monitor the transmission. It can, however,

play back only from the beginning of the file, rather than from any point, as can be the case with true streaming. *See also* stream." EX1017 (Microsoft Computer Dictionary), 66-67.

147. Not surprisingly, large technology companies invested heavily in streaming technologies. Apple's "Streaming Server" is "[a] server technology designed by Apple Computer to send streaming QuickTime media files over the Internet. Built on RTP and RTSP standard Internet protocols, Streaming Server can set up a QuickTime streaming media Web broadcasting station capable of streaming digital videos and music files to more than 3000 users via the Internet. Streaming Server may be used with Mac OS X and other UNIX-based operating systems. *Also called*: Darwin Streaming Server, QuickTime Streaming Server." EX1017 (Microsoft Computer Dictionary), 86.

148. Similarly, Microsoft's "Windows Media Technologies" is "Microsoft technologies for the creation, delivery, and playing of streaming audio and video over a network, including both intranets and the Internet. Windows Media Technologies, downloadable from the Microsoft Web site, support both live and on-demand (delivered from storage) content and are based on files delivered in Advanced Streaming Format (ASF). Three major components—Windows Media Tools, Windows Media Services, and Windows Media Player—comprise Windows

Media Technologies. [] *See also* Advanced Streaming Format. *Compare* Real-System G2." EX1017 (Microsoft Computer Dictionary), 95.

149. Microsoft built an assortment of services for clients and servers to access and stream multimedia files. "Windows Media Services" is "[a] digital media platform that runs on a server, such as Windows 2000, to support streaming media, such as video and audio." EX1017 (Microsoft Computer Dictionary), 95.

150. "Windows Media Player" is "[a] client/control that receives a stream from a Windows Media server or local content for playback. It can run as a stand-alone client executable program. Windows Media Player can also be embedded in a Web page, a C++ program, or a Microsoft Visual Basic program that uses the client ActiveX control." EX1017 (Microsoft Computer Dictionary), 95. Serving clients, a "Windows Media server" is "[a] server on which Windows Media Services has been installed." EX1017 (Microsoft Computer Dictionary), 95.

1. Copy Protection and Digital Rights Management

151. A lot of human effort goes into producing artifacts such as software, writing, music, and video. Authors of such artifacts often wish to protect their creations against unauthorized copying and use.

152. A "copyright" is "[a] method of protecting the rights of an originator of a creative work, such as a text, a piece of music, a painting, or a computer program, through law. In many countries the originator of a work has copyright in the work

as soon as it is fixed in a tangible medium (such as a piece of paper or a disk file); that rule applies in the United States for works created after 1977. Registration of a copyright, or the use of a copyright symbol, is not needed to create the copyright but does strengthen the originator's legal powers. Unauthorized copying and distribution of copyrighted material can lead to severe penalties, whether done for profit or not. Copyrights affect the computer community in three ways: the copyright protection of software, the copyright status of material (such as song lyrics) distributed over a network such as the Internet, and the copyright status of original material distributed over a network (such as a newsgroup post). The latter two involve electronic media that are arguably not tangible, and legislation protecting the information disseminated through electronic media is still evolving. *See also* fair use, General Public License." EX1017 (Microsoft Computer Dictionary), 15.

153. In the context of software, a "copy protection" is "[a] software lock placed on a computer program by its developer to prevent the product from being copied and distributed without approval or authorization." EX1017 (Microsoft Computer Dictionary), 15.

154. Techniques have long existed for protecting digital intellectual property, for example, using encryption and ensuring that only one copy of protected data can exist at any time. "DRM" is "[a]cronym for **D**igital **R**ights **M**anagement.

A group of technologies developed to protect intellectual property from online piracy by controlling who can view protected content and in what form. A DRM package may allow the purchaser to view protected content, but prevent printing or forwarding. Content may also be set to expire after a set amount of time or if distributed to multiple users. DRM technology is meant to protect multiple forms of digital and analog content, and includes encryption, digital watermarking, and content tracking software." EX1017 (Microsoft Computer Dictionary), 27, 20.

155. In particular, considering that networks can fail (see §VIII.B.2 above), copyrighted and/or encrypted data cannot be easily copied from one computer to another. If such copying were permitted, copyright violations would take place. However, transferring a copyrighted artifact between computers can be allowed in some settings, as long as only the total number of licenses cannot be violated. For example, if a user has the right to view one movie file, and that movie file is transferred to another computer, the user should not be able to view the movie concurrently at the source and destination computers: that would likely violate the license by permitting two (instead of one) instances to be viewed concurrently. Only when the file was transmitted fully to its destination, and the source file was, say, deleted, can the movie at the destination be viewed. This poses a problem when the transmitted failed halfway: if the destination file is only partially transferred, the movie cannot be fully viewed, suggesting that the (DRM) right to view the movie at

its destination cannot be activated until the file has transferred fully, and the source DRM right was deactivated.

2. UPnP Standard

156. The Digital Living Network Alliance ("DLNA") is a group of member entities that agreed to collaborate on standards and guidelines for industry for providing interoperable networked home devices, including networked media devices. EX1026 (DLNA Overview), 3. In a 2004 White Paper published by the DLNA, the group envisioned a future in which "digital homes will contain one or more intelligent platforms… [and] will manage and distribute rich digital content to [other] devices" in the home. *Id.*, 5. One functional component developed by the DLNA is the Universal Plug-and-play ("UPnP") media-management architecture, which includes specifications for interaction between UPnP devices and control point applications. "UPnP AV devices can include TVs, VCRs, CD/DVD players, set-top boxes, stereo systems, still[] cameras, electronic picture frames and PCs." The UPnP specifications define two type of devices for use in the home network, including Media Servers and Media Renderers. *Id.*, 14.

157. UPnP Media Servers and Media Renderer specifications define four main services:

(1) "**Content Directory Service**: This enumerates the available content (videos, music, pictures and so forth).

(2) **Connection Manager Service**: This determines how the content can be transferred from Media Server to Media Renderer devices.

(3) **AV Transport Service**: This controls the flow of the content (play, stop, pause, seek and so on).

(4) **Rendering Control Service**: This controls how the content is played (volume/mute, brightness, for example)."

EX1026 (DLNA Overview), 14.

158. The below FIG. 1 from the DLNA UPnP Media Server Specification, dated in 2006, illustrates various sub-units of the Media Server.

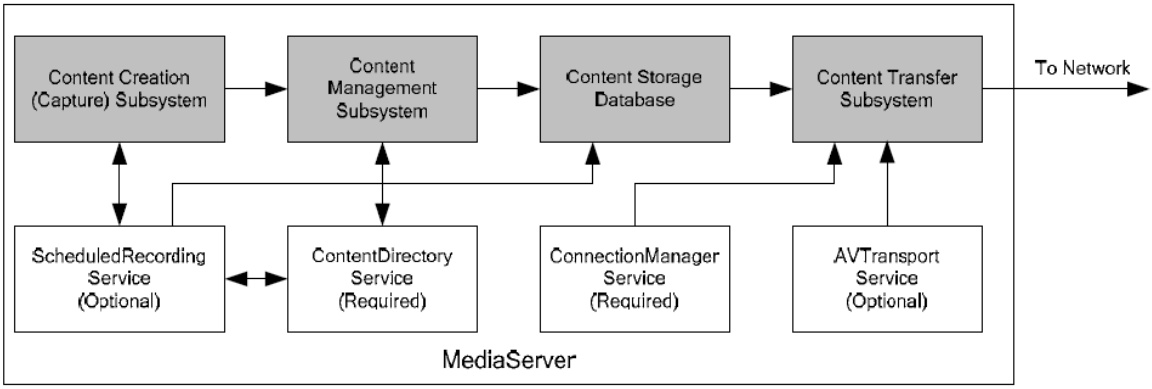


Figure 1: MediaServer Functional Diagram

EX1027 (UPnP Media Server), FIG. 1

F. Caching Technologies

159. Computers store the 0 and 1 values of bits in memory devices. Memory devices can be volatile (lose data when device power is off) or non-volatile (data is not lost when power is off). There are many storage media in existence: tapes, hard

disks, floppies, Flash technologies (*e.g.*, SSDs), optical (*e.g.*, CD-ROM and DVDs), DRAM (*e.g.*, system memory), CPU registers, and more. All these devices can be placed on a specific point or range in a 3-dimensional continuum comprising their (a) capacity, (b) performance, and (c) price. Almost universally, the faster performing devices (*e.g.*, DRAM) are smaller and cost more per unit storage, whereas the larger-capacity devices (*e.g.*, HDD) are slower and cheaper per gigabyte of storage capacity. The difference in speed/cost/capacity among devices is very significant and ranges along at least 9 orders of magnitude—or a billion times bigger, faster, or costlier per unit storage. *See* §VIII.A above.

160. Therefore, "[t]he wide variety of storage systems in a computer system can be organized in a hierarchy (Figure 2.6) according to their speed and their cost. The higher levels are expensive, but are fast. As we move down the hierarchy, the cost per bit decreases, whereas the access time increases. This tradeoff is reasonable; if a given storage system were both faster and less expensive than another — other properties being the same — then there would be no reason to use the slower, more expensive memory." EX1018 (Silberschatz), 15, FIG. 2.6 (reproduced below).

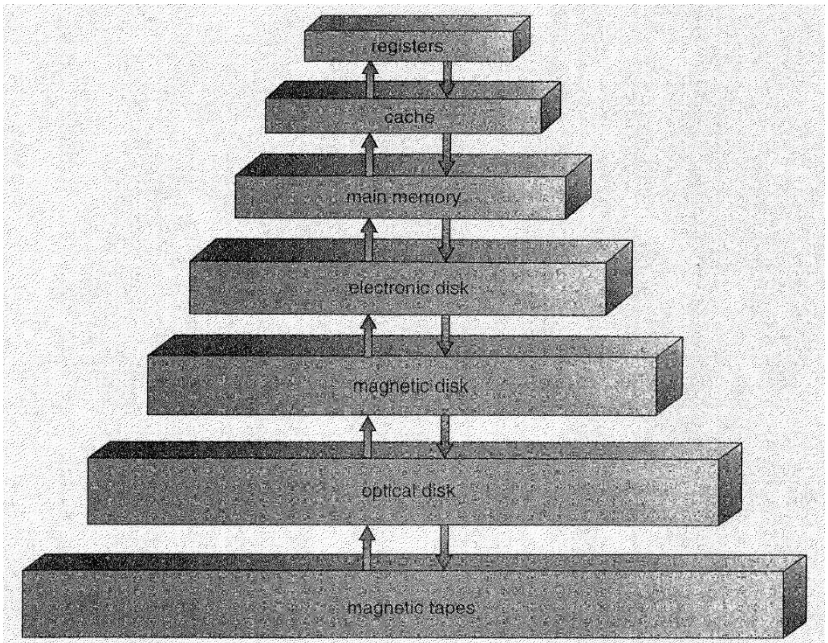


Figure 2.6 Storage-device hierarchy.

EX1018 (Silberschatz), FIG. 2.6

161. Managing all these factors is challenging: cost, speed, access speed/latency, size, and volatility. "The design of a complete memory system attempts to balance all these factors: It uses only as much expensive memory as necessary, while providing as much inexpensive, nonvolatile, memory as possible. Caches can be installed to ameliorate performance differences where there is a large access-time or transfer-rate disparity between two components". EX1018 (Silberschatz), 15-16.

162. "*Caching* is an important principle of computer systems, in both hardware and software. Information is normally kept in some storage system (such as main memory). As it is used, it is copied into faster storage system, the cache, on a temporary basis. When we need a particular piece of information, we first check

whether it is in the cache. If it is, we used the information directly from the cache; if it is not, we use the information from the main storage system, putting a copy in the cache under the assumption that there is a high probability that it will be needed again". EX1018 (Silberschatz), 16.

163. A "cache" is "[a] special memory subsystem in which frequently used data values are duplicated for quick access. A memory cache stores the contents of frequently accessed RAM locations and the addresses where these data items are stored. When the processor references an address in memory, the cache checks to see whether it holds that address. If it does hold the address, the data is returned to the processor; if it does not, a regular memory access occurs. A cache is useful when RAM accesses are slow compared with the microprocessor speed, because cache memory is always faster than main RAM memory. *See also* disk cache, wait state." EX1017 (Microsoft Computer Dictionary), 11.

164. Caches can exist at many levels. A "CPU cache" is "[a] section of fast memory linking the central processing unit (CPU) and main memory that temporarily stores data and instructions the CPU needs to execute upcoming commands and programs. Considerably faster than main memory, the CPU cache contains data that is transferred in blocks, thereby speeding execution. The system anticipates the data it will need through algorithms. *Also called* cache memory, memory cache. *See also* cache, CPU, VCACHE." EX1017 (Microsoft Computer

Dictionary), 16. CPUs internally can have 2 or more caches, marked "level 1" or L1, "level 2" or L2, etc. *See also* EX1017 (Microsoft Computer Dictionary), 51 ("L1 cache" and "L2 cache" definitions).

165. A "disk cache" "[a] portion of a computer's random access memory (RAM) set aside for temporarily holding information read from disk. A disk cache does not hold entire files, as does a RAM disk (a portion of memory that acts as if it were a disk drive). Instead, a disk cache is used to hold information that either has recently been requested from disk or has previously been written to disk. If the required information remains in a disk cache, access time is considerably faster than if the program must wait for the disk drive mechanism to fetch the information from disk. *See also* cache. *Compare* disk buffer." EX1017 (Microsoft Computer Dictionary), 23.

166. A "RAM cache" is "[c]ache memory that is used by the system to store and retrieve data from the RAM. Frequently accessed segments of data may be stored in the cache for quicker access compared with secondary storage devices such as disks. *See also* cache, RAM." EX1017 (Microsoft Computer Dictionary), 71.

167. In any cache hierarchy, typically the cache (above) is faster and smaller than the backend location (below) from where the data was retrieved. Therefore, caches must be managed appropriately. If a data item is found in a cache it is called a "cache hit"—otherwise it is a "cache miss" that requires retrieving the backend

data and copying it into the cache. A "hit" is "1. A successful retrieval of data from a cache rather than from the slower hard disk or RAM. *See also* cache, hard disk, RAM. 2. A successful retrieval of a record matching a query in a database. *See also* query (definition 1), record[]. 3. Retrieval of a file from a Web site. Each separate file accessed on a Web page, including HTML documents and graphics, counts as a hit." EX1017 (Microsoft Computer Dictionary), 46.

168. Cached data is typically stored temporarily. At some point a cache space fills up, and the system must determine which items to "evict"; often the oldest, least used items are evicted first. "Since caches have limited size, *cache management* is an important design problem. Careful selection of the cache size and of a replacement policy can result in 80 to 99 percent of all accesses being in the cache, resulting in extremely high performance. Various replacement algorithms for software-controlled caches are discussed in Chapter 9." EX1018 (Silberschatz), 17.

169. Such caching techniques are particularly effective when considering users' work habits and workload patterns. Studies have shown that users and applications often operate on a small set of data files (called a "working set") within a narrow time period. For example, while drafting this very declaration, I am editing the same Word document repeatedly over a period of several days and weeks. Thus, my OS (and any OS prior to 1998) recognizes this access pattern and keeps this Word file cached in DRAM for long time periods. Conversely, it makes less sense

for the OS to waste precious DRAM to cache documents I've not worked on for months. However, if, for any reason, I try to access a document I've not opened in a long time, and the document is not already cached in DRAM, then indeed my OS will have to incur a longer latency—and I will have to wait for however long it takes to retrieve the older document from the slower HDD (and cache it into DRAM).

170. Cache management considers when to "admit" data into the cache from backend sources and when to "evict" older data in the cache to make room for new items. One of the most popular and successful cache eviction algorithms is the Least-Recently-Used ("LRU") one, a heuristic stating that items used more recently are more likely to be used in the near future, and therefore should be kept—while lesser used items should be evicted. LRU uses various thresholds to determine when and what to evict, based on space available in the cache, the desire to make extra room in the cache for new items, and/or the length of time that items have been in the cache and when were they last used. "LRU replacement associates with each page the time of that page's last use. When a page must be replaced, LRU chooses that page that has not been used for the longest period of time." *See* EX1018 (Silberschatz), 23; *see also, generally*, *id.*, 22-28.

171. Admitting an item from a backend into a cache, or retrieving it from the cache itself, can be seen as a "get" (or read) operation. Conversely, an application inserting an item into a cache, is also called "put" (or write) operation. When new

items are inserted into a cache by an application, they are said to be "dirty" because their content is newer than the corresponding backend item. Such dirty items must be "flushed" or written to the backend before they are removed from the cache. A "dirty bit" is "[a] bit used to mark modified data in a cache so that the modifications may be carried over to primary memory. *See also* bit, cache." EX1017 (Microsoft Computer Dictionary), 22.

172. To "flush" is "[t]o clear a portion of memory. For example, to flush a disk file buffer is to save its contents on disk and then clear the buffer for filling again." EX1017 (Microsoft Computer Dictionary), 40 (definition 2). In some caches, items in the cache could become "stale" if the corresponding backend item could have been updated: in that case, some caches attempt to revalidate any cached item before returning it to an application, and as needed remove stale items.

173. Caching is applicable anywhere there is room in one memory device to access data much faster than its source location. Other than RAM-HDD caches, one can cache items on a local storage device that would otherwise take even longer to retrieve than the HDD's access speeds (*e.g.*, network storage, distant servers), as well as caching on intermediate devices such as proxies. A "proxy server" is "[a] firewall component that manages Internet traffic to and from a local area network (LAN) and can provide other features, such as document caching and access control. A proxy server can improve performance by supplying frequently requested data,

such as a popular Web page, and can filter and discard requests that the owner does not consider appropriate, such as requests for unauthorized access to proprietary files. *See also* firewall." EX1017 (Microsoft Computer Dictionary), 66.

174. There are dozens of cache-management algorithms in existence, some decades old. Silberschatz describes several algorithms used to manage memory resources in an OS. *See* EX1018 (Silberschatz), 18-28 ("9.5 Page-Replacement Algorithms", describing techniques such as FIFO, LRU, LFU, MFU, and more), 18 ("There are many different page-replacement algorithms. Probably every operating system has its own unique replacement scheme. How do we select a particular replacement algorithm? In general, we want the one with the lowest *page-fault rate*.").

1. Least-Recently Used (LRU)

175. One of the oldest, most effective, and best-known algorithms is the Least-Recently-Used (LRU) algorithm. LRU leverages the aforementioned property that users are more likely to work on data files that were in active use in the recent past. LRU orders all cached items by the last time they were accessed. When the cache needs to make room for new items, LRU evicts enough items starting from the least-recently (*e.g.*, oldest) items first. *See* EX1018 (Silberschatz), 20-21 ("**9.5.3 LRU Algorithm** [] If the optimal algorithm is not feasible, perhaps an approximation to the optimal algorithm is possible. … If we use the recent past as

an approximation of the near future, then we will replace the page that *has not been used* for the longest period of time (Figure 9.11). This approach is the *least recently used (LRU)* algorithm. LRU replacement associates with each page the time of that page's last use. When a page must be replaced, LRU chooses that page that has not been used for the longest period of time.").

2. Least-Frequently Used (LFU)

176. Another popular algorithm is Least-Frequently-Used ("LFU"). LFU counts not the recency but the frequency of access of cached items. LFU evicts items that have the lowest frequency of use first. EX1018 (Silberschatz), 28 ("**LFU Algorithm**: The *least frequently used (LFU)* page-replacement algorithm requires that the page with the smallest count be replaced. The reason for this selection is that an actively used page should have a large reference count.").

177. Neither LRU nor LFU are perfect. Both work well for certain access patterns but may make poor decisions for other access patterns, possibly evicting items that are needed shortly afterwards. For that reason, researchers have developed cache-management techniques that consider other properties and thresholds: the size of the cached object, object priorities, who are the users who requested the item, inter-relationship among items (*e.g.*, items A and B are needed together), and numerous permutations and combinations (*e.g.*, LRU+LFU) over many years, starting in the 1970s. In sum, all cache management techniques attempt

to prioritize cached data—which data should be evicted (because it is less important) and which should be kept (because it is deemed more important).

178. While these techniques are described here in the context of cache management, they are generally applicable anywhere resources are managed. When resources are managed, resource consumers are often prioritized. Such priorities are vital to determine which resources to move elsewhere, while minimizing disruption to operations; it is obvious that, to minimize disruption, resource consumers with the lowest priority should be removed (or discarded) first. For example, one can use LRU, LFU, etc. to prioritize resource users and decide which applications or VMs to migrate from one host (or data center) to another, which network packets to transmit first, which applications to schedule to run first, which web pages to cache locally, which files to cache on a proxy device, etc.

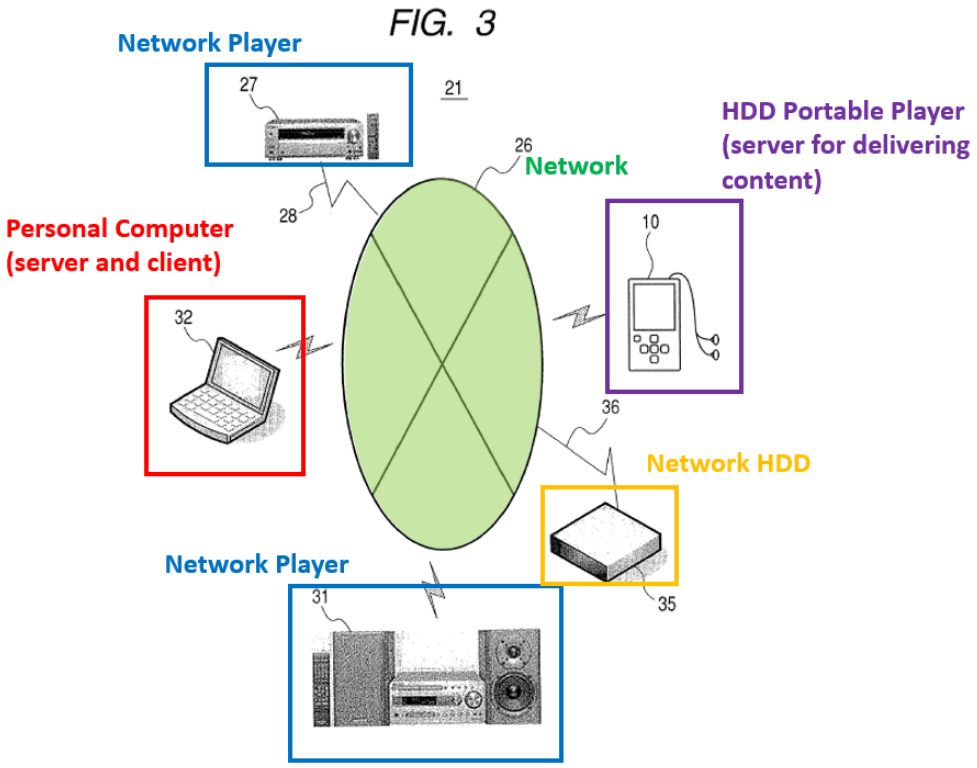
179. And once again, poor priorities and "predictions" can result in less favorable outcomes. For example, a VM that wasn't deemed a high priority before could be offloaded to a distant data center with fewer resources; but if that VM (or the applications it runs) suddenly becomes more important (*e.g.*, due to a spike of activity), then that VM and its application will suffer poor performance, whereas if that VM was not migrated in the first place, it would have performed much better now. The same is true for any other resource than VMs (*e.g.*, files).

IX. OVERVIEW OF THE ’101 PATENT

A. Overview of the ’101 Patent

180. The ’101 patent discusses “a server device for media such as an HDD portable player” and “a method for controlling a server for media… capable of smoothly dealing with [] large amounts of digital contents.” EX1001 (’101 patent), 1:7-12. “Although the capacity of the HDD … is large, it is limited [and t]hus, the HDD can store limited amounts of music pieces.” *Id.*, 1:51-53, 3:52-55.

181. “The HDD portable player 10, a network player 31, and a personal computer 32 are wirelessly connected to the network 26.” *Id.*, 4:17-19. The ’101 patent describes that “[t]he HDD portable player 10 functions as a server for delivering music pieces, pictures and other contents” For “network-playback” via network players 27 and 31 or through “self-playback” of the HDD portable player 10. *Id.*, 4:23-25, 4:32-37. FIG 3., reproduced and annotated below illustrates the ’102 patent’s key components.



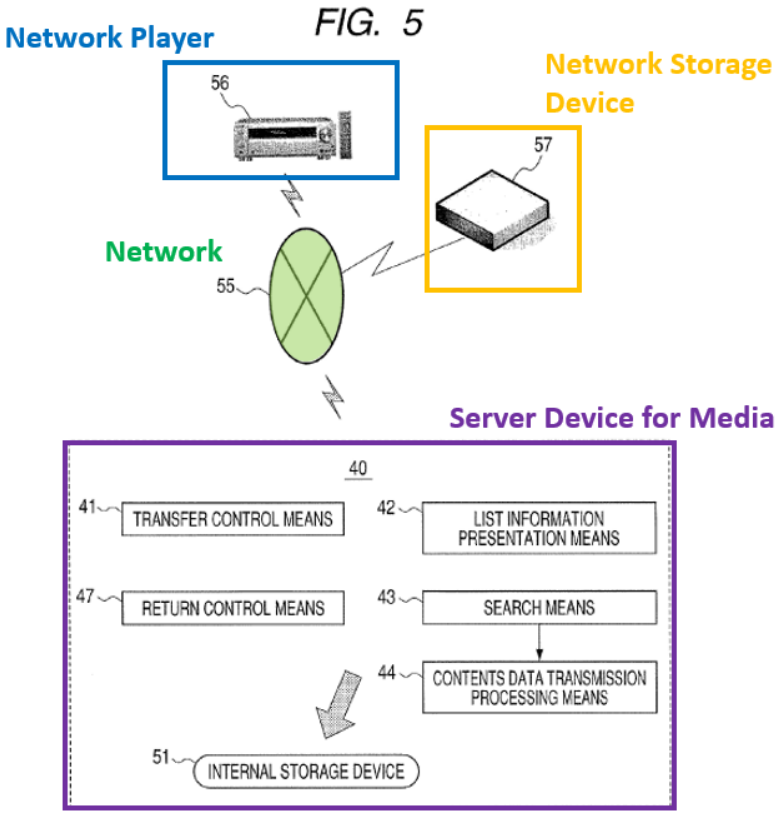
EX1001 ('101 patent), FIG. 3 (annotated)

182. In one example, to listen to music, "[a] user operates the network player 27 to request the HDD portable player 10 to transmit list information about the held music pieces of the HDD portable player 10 [and] the HDD portable player 10 transmits the list information to the network player 27 through the network 26." *Id.*, 4:52-56. After the user selects a song to be played at the network player 27, "the HDD portable player 10 transmits the corresponding data to the network player 27 by the stream-delivery." *Id.*, 4:57-66.

183. The '101 patent discusses that "[t]he HDD portable player 10 exemplifies the server device for media 40." *Id.*, 6:21-22. "The server device for

media… is equipped with an internal storage device for storing digital contents." *Id.*, 6:9-11, 2:45-48.

184. "FIG. 5 is a block diagram showing a server device for media 40. The server device for media 40 is equipped with a network storage device 57 for storing the digital contents, and, during connection to the network, responds to the data transmission request from the network player 56 by stream-delivering the corresponding data of the corresponding digital contents from the internal storage device 51 to the network player 56." *Id.*, 6:9-16, FIG. 5 (annotated below).

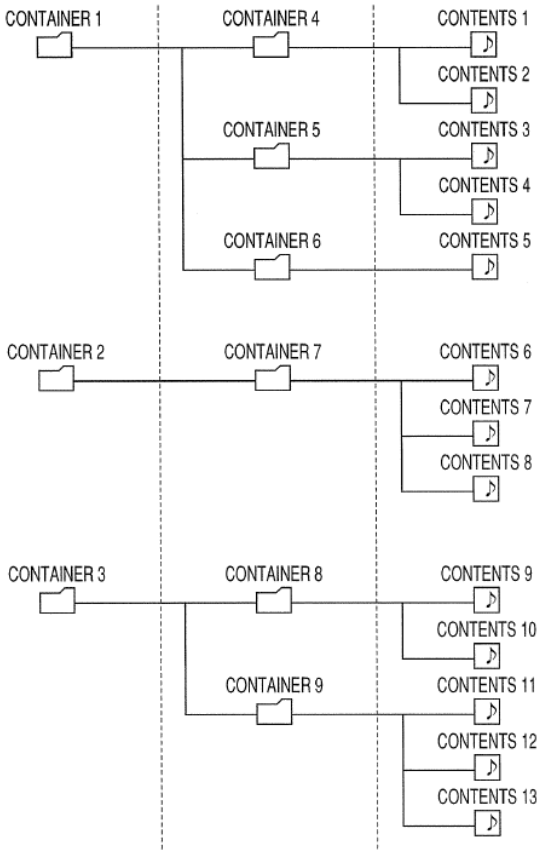


EX1001 ('101 patent), FIG. 5 (annotated)

185. I have annotated the figures of the '101 patent with colors assigned to each of the structures recited in the claims (*e.g.*, blue for network player, yellow for network storage device, etc.). I continue these matching color-mappings in the references that I discuss in this Declaration for convenience.

186. The '101 patent goes on to describe that "[t]he server device for media 40 has transfer control means 41, list information presentation means 42, search means 43, and contents data transmission processing means 44." *Id.*, 6:17-19. As described above, the server device for media includes internal storage for storing digital contents, and "[t]he transfer control means 41 transfers part of held digital contents in the internal storage device to the network storage device 57, which is connected to the network 55 and is capable of storing data, and stores the contents there." *Id.*, 6:28-31, 6:57-60, 7:4-6. The server device for media's list information presentation means "responds to [a] list presentation request for the held digital contents of the server device for media 40 issued from the network player 56 by transmitting the list information" to the network player. *Id.*, 6:31-35. The list information lists the digital contents that are stored in the internal storage of the server device for media as well as the digital contents that were transferred to the network storage device in a directory having a tree structure. *Id.*, 6:35-40, 5:47-67, 8:25-30, 9:4-11, FIG. 4 (reproduced below).

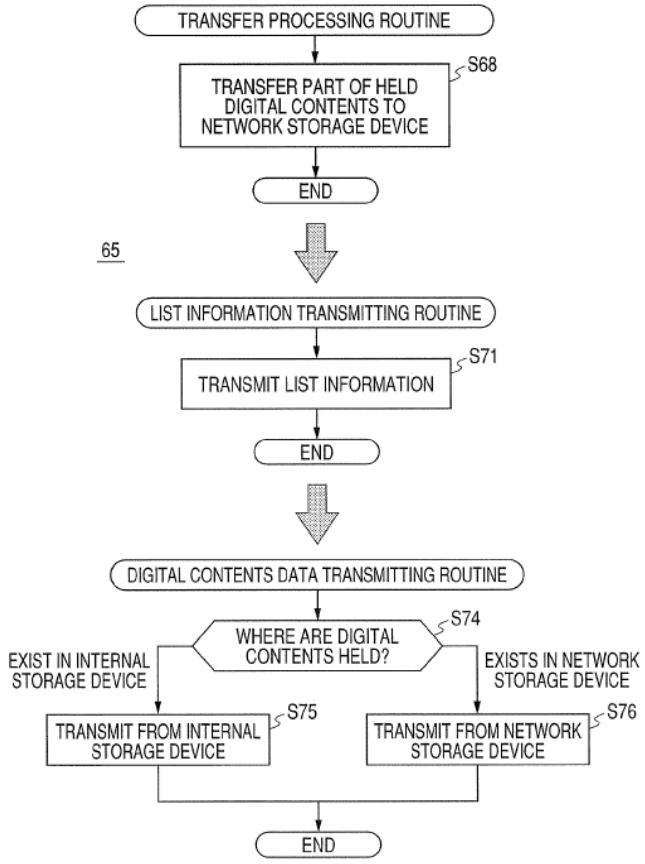
FIG. 4



EX1001 ('101 patent), FIG. 4

187. The '101 patent describes that "search means 43 responds to [a] data transmission request for the held digital contents issued from the network player 56 by searching for the location where the held digital contents are currently stored [and if] the result of search shows the network storage device 57 [are located in the network storage device], the contents data transmission processing means 44 makes the corresponding data stream-delivered from the network storage device 57 to the network player 56." *Id.*, 6:41-48, FIG. 6 (reproduced below).

FIG. 6



EX1001 (’101 patent), FIG. 6

188. The ’101 patent describes that “[t]he program according to the present invention causes a computer to function as each means of the above-described server device for media [and] also causes a computer to execute each step of the above-described method.” *Id.*, 3:5-10.

B. Summary of Prosecution History of the ’101 Patent

189. I have reviewed the file history of the ’101 patent (EX1002). I summarize certain relevant information that I have reviewed pertaining to this file history below.

190. The ’101 patent issued from a U.S. National Stage application, Application 12/527,777 (“the ’777 application”), filed September 2, 2009. The independent claims and various dependent claims were initially rejected as obvious over U.S. 2006/0184972 (EX1010, “Rafey”) in view of U.S. 2007/0238471 (EX1011, “Bae”). Dependent claims 6-7 and 14-15, directed to scenarios in which some of the digital contents would be unrecoverable in the event of a network failure, were rejected as obvious over Rafey and Bae in further view of U.S. 7,219,129 (EX1006, “Fiechter”). EX1002 (’101 Patent File History), 169-188. In response, Applicant amended the independent claims to specify that “the list information maintains a tree structure of the digital contents in the internal storage device before transferring the digital contents to the network storage device” and argued that the cited references did not teach this feature. EX1002 (’101 Patent File History), 85-88, 89-92. Applicant did not include amendments or arguments to overcome the rejections of dependent claims 6-7.

191. In a subsequent Office Action, the independent claims and particular dependents were rejected over Rafey, Bae, and U.S. 2008/0104219 (EX1009,

"Kageyama"). EX1002 ('101 Patent File History), 62-78. Although all claims 1-8 were *listed* as rejected (*Id*., 63), no specific rejection of dependent claims 6-7 over prior art or otherwise was included in the Office Action. *Id.*, 62-78. Applicant responded to the rejection by amending the independent claims to incorporate the subject matter of claim 6, rewriting claim 7 in independent form, and adding a new method claim mirroring rewritten claim 7. *Id.*, 46-56. Applicant argued that "although the Official Action summary indicates that claims 1-8 stand rejected, dependent claims 6 and 7 do not appear to have been rejected based on any prior art grounds and instead appear to stand rejected for being dependent upon claim 1, which is rejected under §112… it is believed that the subject matter of dependent claims 6 and 7 is otherwise in condition for allowance if placed in independent form and the §112 rejection overcome." *Id.*, 54 (emphasis in original).

192. The application was allowed with an Examiner's amendment to move subject matter in the preamble of the independent claims into the body of the claims to recite "an internal storage device for storing digital contents, wherein the server device for media response to a data transmission requires from a network player by stream-delivering corresponding data in corresponding digital contents from the internal storage device to the network player during connection to a network" and "wherein the server device for media is a media player." *Id.*, 7-19.

193. That is, the Office inexplicably omitted a prior art rejection of claims 6 and 7 in the November 16, 2011 Office Action without previous argument or amendment by Applicant relating to those claims. EX1002 ('101 Patent File History), 62-78. Applicant then apparently capitalized on the Office's omission by amending the independent claims to incorporate the subject matter of claim 6, rewriting claim 7 as independent and adding a new method claim mirroring claim 7 to obtain an allowance. *Id.*, 46-56.

194. However, the features of dependent claims 6 and 7 were well-known by the Critical Date. For example, the feature of "said transfer control unit does not transfer, from the internal storage device to the network storage device, the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device" (recited in the '101 patent's claims in §I above at element [1b-iii]) is described by Lamkin, Fiechter, Ito, and Franke. Additionally, the feature of "wherein the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device is transferred after obtaining permission from a user" (recited in the '101 patent's claims in §I above at element [6b-iii]) is described by Lamkin, Fiechter, and Franke, among other references.

C. Priority Date of the Challenged Claims

195. The application that issued as the '101 patent was filed in the USPTO on September 2, 2009 as a U.S. National Stage application under §371 of PCT/JP2007/054603, filed March 2, 2007. EX1001 ('101 patent), 1.

196. I have been instructed by counsel to assume, for the purpose of this *inter partes* review proceeding, that the priority date of claims 1-12 of the '101 patent is the March 2, 2007 filing date of the PCT application PCT/JP2007/054603 ("Critical Date").

X. SUMMARY OF THE PRIOR ART

1. Overview of Lamkin (EX1004)

197. Lamkin describes a "network media service [for] managing media content over a network." EX1004 (Lamkin), [0002]. Lamkin notes that "places of business and many homes have more than one computer [which] can be networked together to allow the sharing of resources." *Id.*, [0003], FIG. 1 (annotated below).

198. In Lamkin's annotated FIG. 1 below, I have annotated structures of the Lamkin system in the same colors that I used above in annotating the figures of the '101 patent to highlight similar features and structures. These color mappings continue throughout this Declaration for consistency.

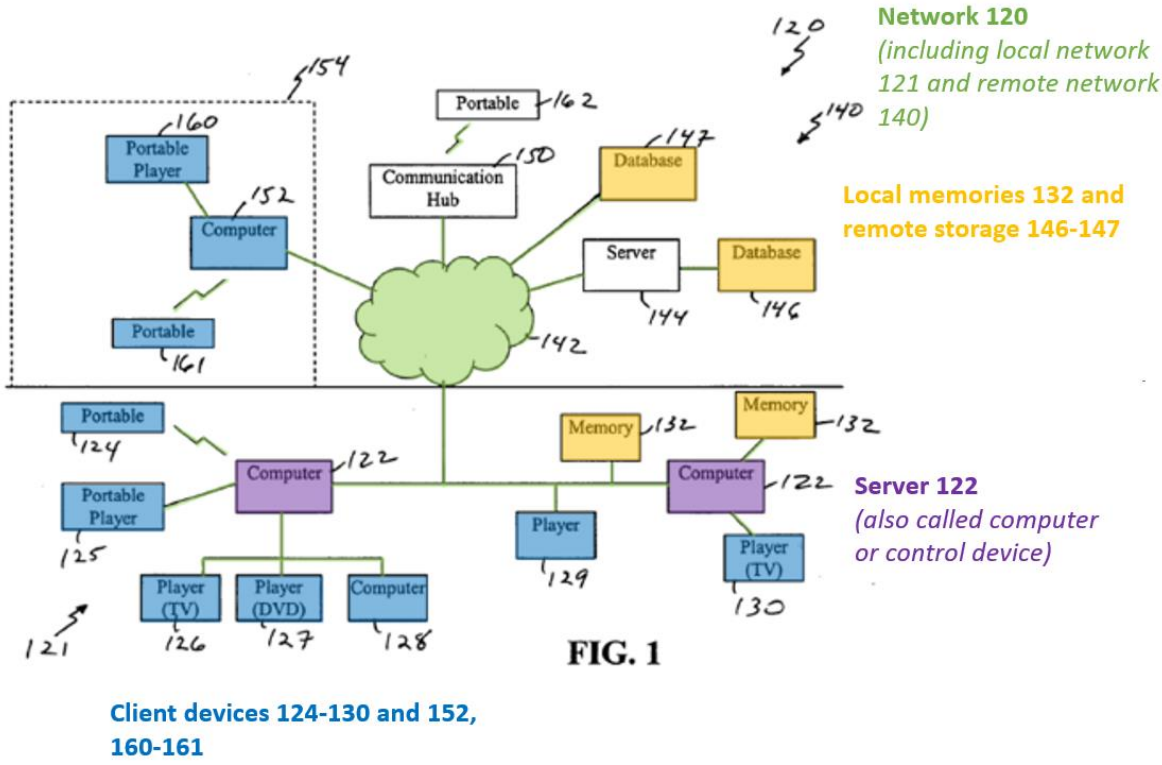


FIG. 1

EX1004 (Lamkin), FIG. 1 (annotated).

199. Lamkin's annotated FIG. 1 shows a network 120 of shared resources. "The network allows communication of content, such as multimedia content, between components of the network [which includes] a local network 121, and in some implementations, a remote network 140." EX1004 (Lamkin), [0049]. "The local network 121 typically includes one or more local servers, computers or other devices 122 and client devices 124-130." *Id.* "The one or more local servers, computers or other control devices 122 provide control over the local network 121 and/or remote network 140, communicate with the client devices 124-130 and

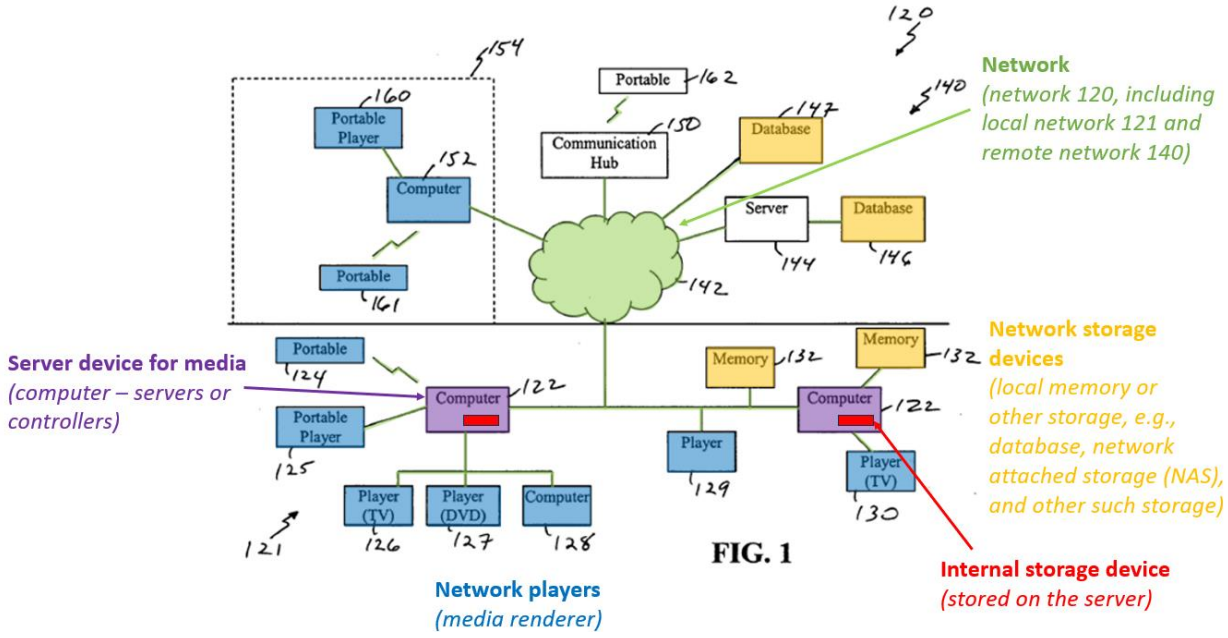
storage or other memory 132 (e.g., database, network attached storage (NAS), and other such storage) that store content." *Id.*

200. Lamkin discloses or renders obvious each element of claims 1-12, for the reasons discussed in the Petition, and as I further explain below. For ease of reference, I provide the following table mapping elements of Lamkin's disclosed system to the major structures of the independent claims of the '101 patent, along with the colors that I use to highlight these features in the Lamkin reference, where applicable.

Claim Term	**Mapping in Lamkin**
server (purple)	Server 122 (also called computer 122)
internal storage (red)	Stored in the server
network (green)	Network 120 (including local network 121 and remote network 140)
network storage (yellow)	Local memory 132 and remote storage/database 146-147
network player (blue)	Portable players / client devices 124-130, 152, 160, 161
transfer control unit	Content management service 2640
list information transmission unit	Content directory service (referred to as CDS)
search unit	Server controls 650 (including server use of universal locator by, for example, a content search engine)

Claim Term	Mapping in Lamkin
digital contents data transmission processing unit	Content management service 2640 / AV transport service 2734

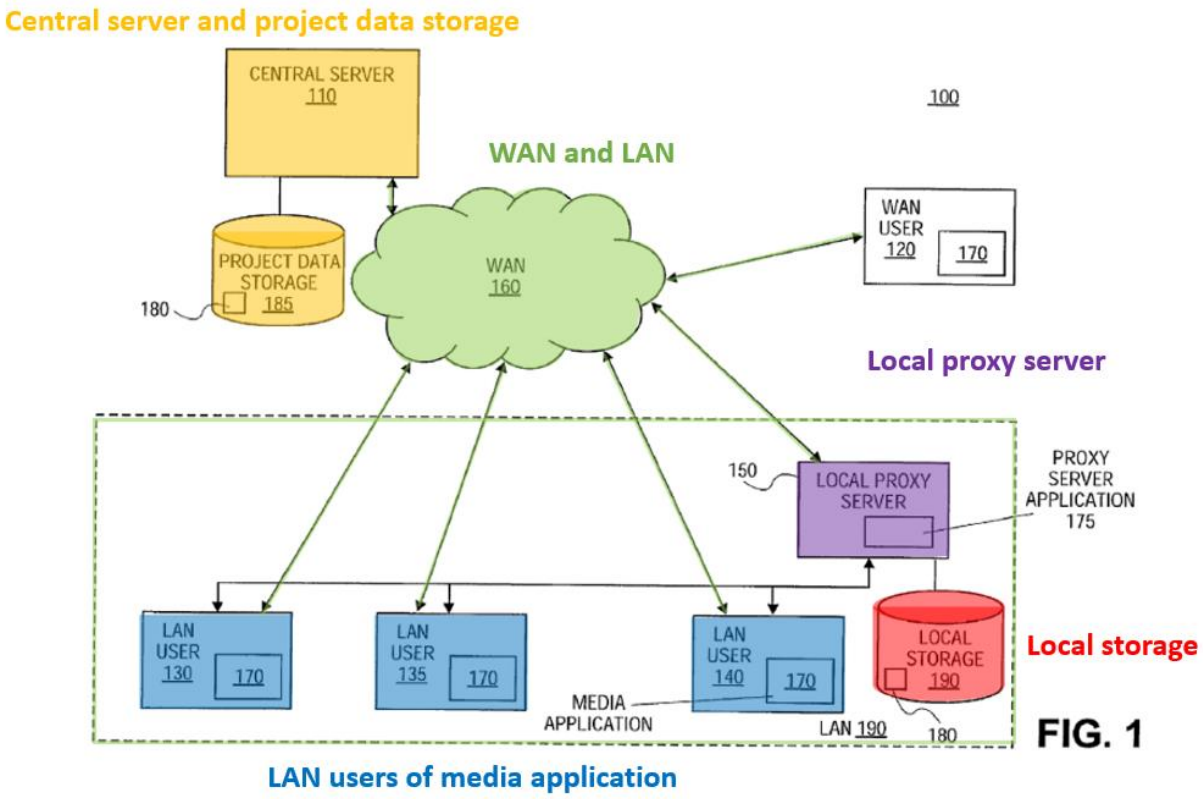
201. The mapping, at a high level, is shown in the annotated Lamkin FIG. 1 below. I have annotated the figure so that the claim element is in bold and the component of Lamkin that maps to the claimed element is listed in parenthesized italics below the element. I have also added the two red boxes labeled as "Internal storage device" to the figure below; these boxes do not appear in Lamkin's figure, but as I will explain further below, the existence of internal storage devices for storing content in or on computers 122 is described by Lamkin. *See e.g.*, EX1004 (Lamkin), [0050], [0052], [0077]-[0078]. Moreover, a POSITA would have known that computers have internal memory and storage devices. *See* §VIII.A above.



EX1004 (Lamkin), FIG. 1 (annotated)

2. Overview of Franke (EX1005)

202. Franke describes "data sharing systems…using a local proxy server to process media data for local area users." EX1005 (Franke), [0002]. As illustrated below in annotated FIG. 1, the system includes "LAN users 130-140 … coupled to a local proxy server 150 [which] acts as intermediary between central server 110 and LAN users 130-140." *Id.*, [0030]. "Central server 110 is a computing system that can communicate with LAN users 130-140, WAN user 120, local proxy server 150, and any other devices connected to WAN 160." *Id.*, [0032]. Franke's "central server 110 can store, manage, and distribute media data and object data in a project database to facilitate collaboration between LAN users 130-140 and WAN user 120 on the same project." *Id.*, [0033], [0034].



EX1005 (Franke), FIG. 1 (annotated)

203. Franke describes that "LAN users 130-140 and WAN user 120 can share data by collaborating on a project including project data [which] may comprise media data and object data." *Id.*, [0031]. "LAN users 130-140, when posting media data, can store media data on local proxy server 150 [which] can then store the media data for the users on central server 110 as media data 180." *Id.*, [0037], [0042]. Franke's "local proxy server 150 can copy or download media data or object being stored on central server 110 [such that] LAN users 130-140 can copy project data being stored on central server 110 from local proxy server 150, thereby accessing project data locally, e.g., on LAN 190." *Id.*, [0042].

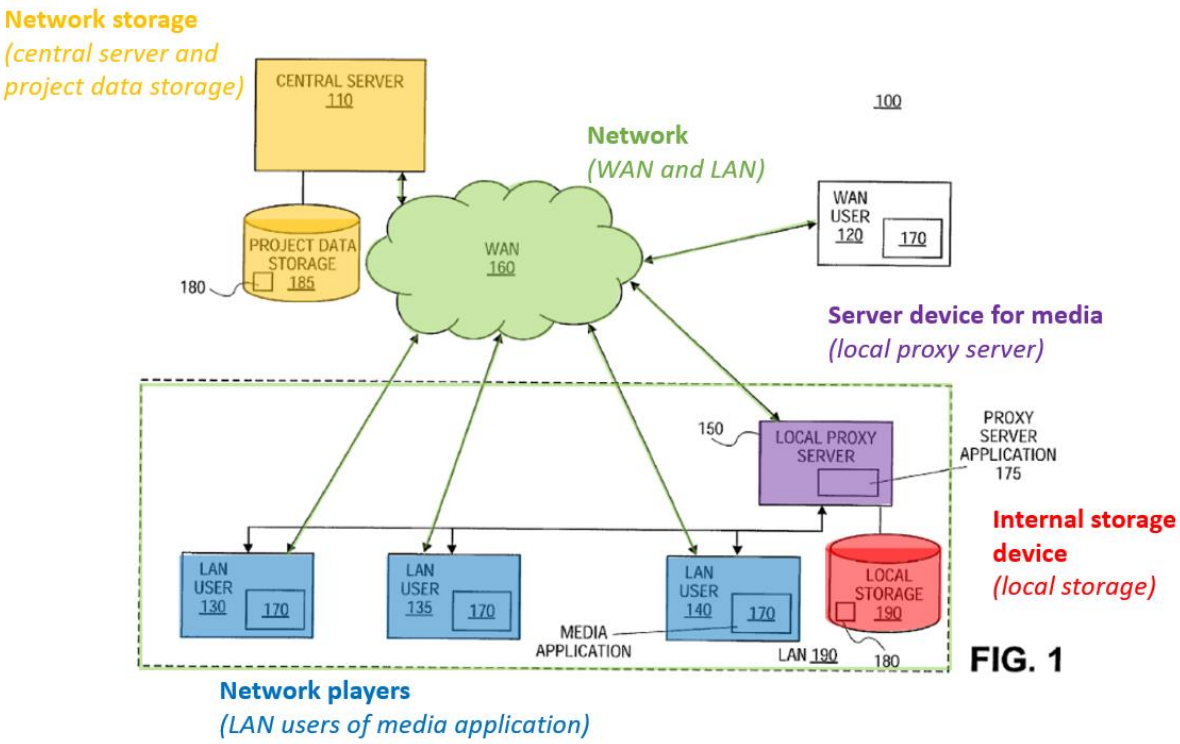
204. "Local storage 190 may represent a database storage system having one or more storage devices, a stand-alone storage device providing shared data access for LAN users 130-140, or other appropriate shared storage systems or devices [such as] storage devices contained within local proxy server 150." *Id*., [0045]. Franke's "[l]ocal proxy server 150 includes a server application 175, which can communicate with media application 170 to handle requests for media data or object data from LAN users 130-140 or store media data or object on central server 110 for the LAN users 130-140." *Id*., [0047]. For example, "[p]roxy server application 175 may also be used to transfer such data files to LAN users 130-140 or other users [and] can be used to identify, organize, and reference data files in local storage 190." *Id*.

205. "To download project data, LAN users 130-140 and WAN user 120 can request project data from central server 110 directly [or] can also request project data from local proxy server 150." *Id*., [0053], [0054]. "If local proxy server 150 does not contain the requested project data, local proxy server 150 can request it from central server 110 and then distribute it to LAN users 130-140." *Id.*, [0053].

206. Franke discloses or renders obvious each element of claims 1-12, as described in the Petition and as I further discuss below. For ease of reference, I provide the following table mapping elements of Franke's disclosed system to the major structures of the independent claims of the '101 patent, along with the colors that I use to highlight these features in the Franke reference, where applicable.

Claim Term	Mapping in Franke
server (purple)	Proxy server 150
internal storage (red)	Local storage 190
network (green)	WAN 160 and LAN 190
network storage (yellow)	Central server 130 (plus project storage 185)
network player (blue)	LAN users 130-140 using media application 170
transfer control unit	Proxy server application 175
list information transmission unit	Proxy server application 175
search unit	Proxy server API and application 175
digital contents data transmission processing unit	Proxy server application 175

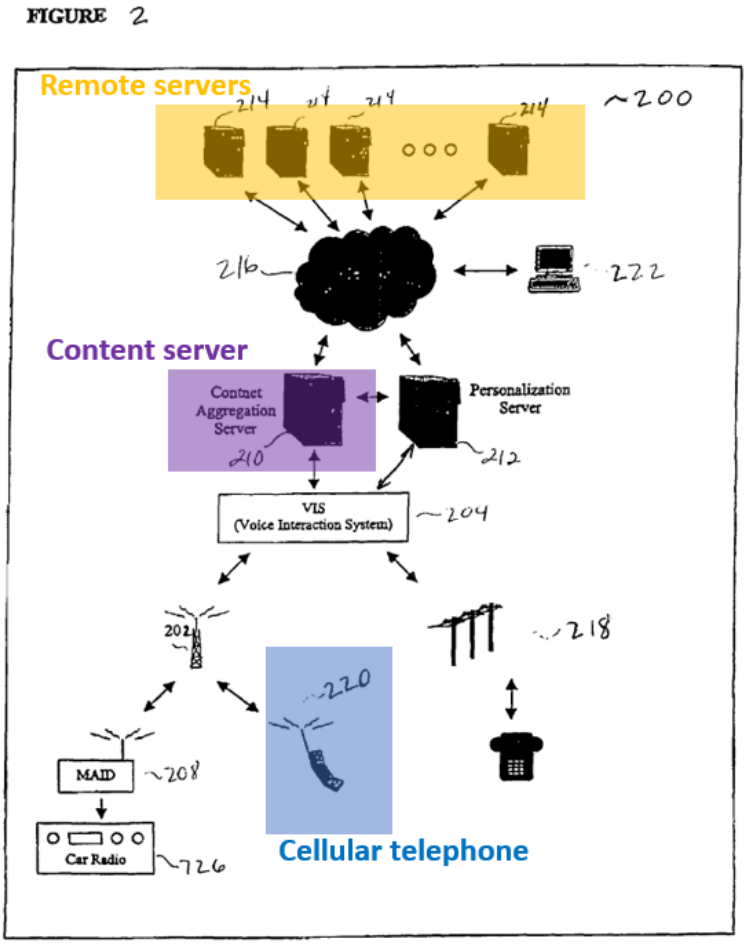
207. The mapping, at a high level, is shown in the below annotated Franke FIG. 1. I have annotated the figure so that the claim element is in bold and the component of Franke that maps to the claimed element is listed in italics below.



EX1005 (Franke), FIG. 1 (annotated)

3. Overview of Fiechter (EX1006)

208. Fiechter describes a browser device "capable of transmitting a request for information via a wireless communication interface from one or more servers in an information network." EX1006 (Fiechter), 3:10-14. As shown in FIG. 2, Fiechter's network browser system includes a user device such as cellular telephone system 220 that receives user commands and transmits the commands to content server 210 which executes the command, including providing requested information from servers 210, 212, and 214. *Id.*, 5:46-57, 5:61-64, 7:52-66, FIG. 2 (annotated below).



EX1006 (Fiechter), FIG. 2 (annotated)

209. As highlighted in Fiechter's annotated Figure 2 above, Fiechter describes a system in which a network player (cellular telephone, highlighted in blue) accesses content stored in network storage (remote servers, highlighted in yellow) through an intermediate media server (content server, highlighted in purple). Fiechter's system allows a user to customize a playlist of content that includes categories and subcategories. *Id.*, 6:58-64.

210. Fiechter describes that "mobile audio device 904 includes capability to detect and compensate for data transmission errors in wireless communication network 906," including detection that the device is in an area "where there is a high incidence of data loss due to propagation errors." *Id.*, 17:50-59.

211. "When mobile audio device 904 is likely to experience a data loss rate that is higher than a pre-selected value, one of several options may be taken including informing the user of the likelihood of errors or gaps in the information, asking the user if they would prefer to wait for the information until transmission of the data will be more reliable, or buffering a greater than normal amount of information to be able to continue undisrupted output until the connection is re-established." *Id.*, 17:62-18:3, 12:41-45.

4. Overview of Ito (EX1012)

212. Ito describes a backup system includes a backup device and a recording and playback device which includes storage means and "content transmitting means for … transmitting the read out content o[f the storage means] to the backup device." EX1012 (Ito), [0007], [0001]. Ito describes that, in some cases, content is lost "due to mishandling or failure" including "a failure of the recording and playback device or the network to which the recording and playback device is connected." *Id.*, [0008].

213. Ito explains that, in some cases, copy control information related to digital content "indicates that copying is allowed only once (Copy Once)." *Id.*, [0002]. Ito describes that, for such copy-protected content, if the information associated with the content "indicates that backup is prohibited, [then] the transmission is terminated." *Id.*, [0252]; *see also id.* ("If the backup process indicates permission to generate backup information corresponding to the encrypted content to be backed up, the encrypted content is sent to the backup device").

XI. GROUND 1A: LAMKIN RENDERS CLAIMS 1-12 OBVIOUS

214. In my opinion, a person of ordinary skill in the art would have found claims 1-12 of the '101 patent to have been obvious based on Lamkin. In my opinion, Lamkin renders each of the limitations of claims 1-12 obvious for at least the reasons set forth in the Petition, which I have read and fully endorse. To streamline the analysis in this declaration, I do not repeat all of the reasons stated in the Petition as to why a POSITA would have found claims 1-12 obvious based on Lamkin. Nonetheless, by way of example and to highlight certain points, I make the following additional observations.

A. Claim 1

1. Claim [1pre]: A server device for media, the server device for media comprising

215. To the extent the preamble is limiting, Lamkin discloses [1pre].

216. Lamkin provides a server ("server device") for storage and management of multimedia content ("for media"). *See* EX1004 (Lamkin), [0241] ("server 122, such as a media server…"), [0227] ("audio/video (AN)… servers that can be referred to as media servers"), [0049]. For example, Lamkin's Fig. 1 shows a computer 122 (annotated in purple in FIG. 1 below).

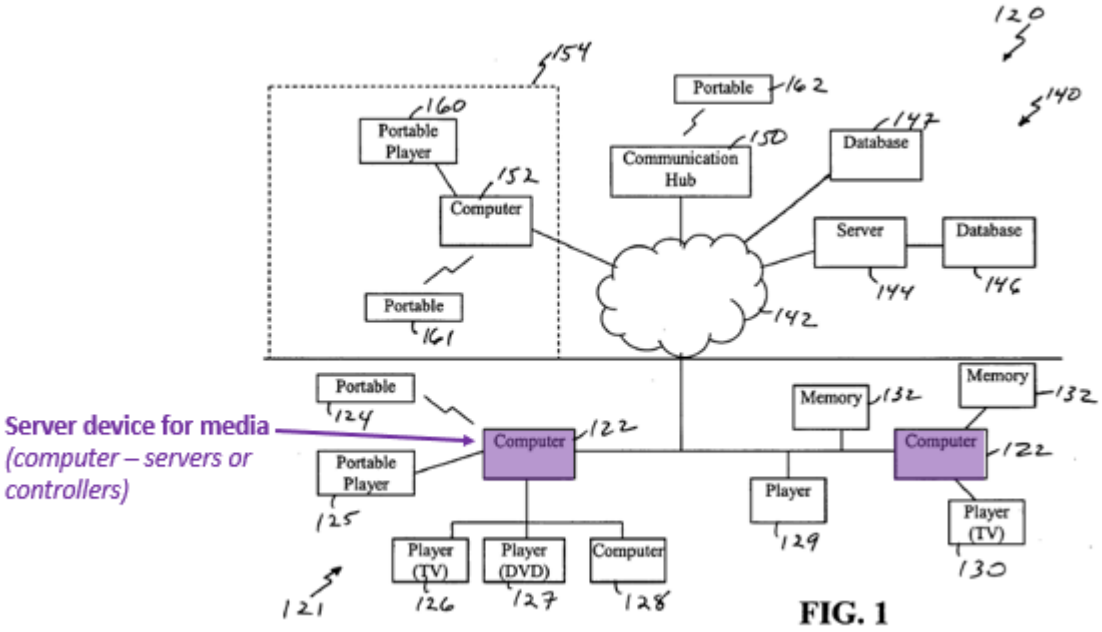


FIG. 1

EX1004 (Lamkin), FIG. 1 (annotated)

217. Lamkin's local server 122 interacts with client devices and local or remote storage device to store media content, and is accordingly a server device operable on a network. EX1004 (Lamkin), [0049], [0050], [0002], [0052]. As described above, a POSITA would have been familiar with server devices and would have understood the operation of such devices as an intermediate (or sometimes

called “proxy”) layer facilitating interactions between client devices and backend storage devices, just as Lamkin describes. *See* §VIII.D above.

218. Lamkin’s description of multimedia content including such content types as audio and video content is consistent with the ’101 patent’s description of the type of media stored and handled by a server device for media, including music pieces and images, moving pictures, or videos. *Compare id.*, [0049] (“video content, audio content”) *with* EX1001 (’101 patent), 1:48 (“music pieces held therein”), 5:19-21 (“A content is not limited to a music piece, but can be other contents to be played such as pictures including moving pictures”).

219. **In sum**, regarding element [1pre], Lamkin discloses “a server device [Lamkin’s computer 122] for media [Lamkin’s video/audio content].”

220. For similar reasons, to the extent the preamble is limiting, Lamkin also discloses claim limitations [6pre], [7pre], and [12pre]. *See* §I above (Table of Claims).

2. Claim [1a-i]: an internal storage device for storing digital contents

221. Lamkin discloses claim limitation [1a-i].

222. For example, Lamkin refers to the storage of content in or on the server 122 itself, in addition to other storage options outside of the server, indicating that the server includes “an internal storage device” to store the content being held in or on the server. *See* §X.1 above (Lamkin overview); *see also* EX1004 (Lamkin),

[0050] ("content can be received from the client devices and **stored in the servers 122**")[1], [0052] ("Content can further be stored in a local memory or other storage 132, and/or be **stored on the one or more servers 122**"), [0078] ("one of the servers 122 can be designated as a central storage **to store content**"), [0077] ("content is transferred and/or stored to one or more storage locations for centralization, such as **on the server**").

223. As seen in annotated FIG. 1 below, Lamkin describes computers or servers 122 (colored purple) which Lamkin describes as including internal storage (boxes added in red).

[1] All emphases in this Declaration in bold and/or underline in the grounds is mine unless otherwise noted.

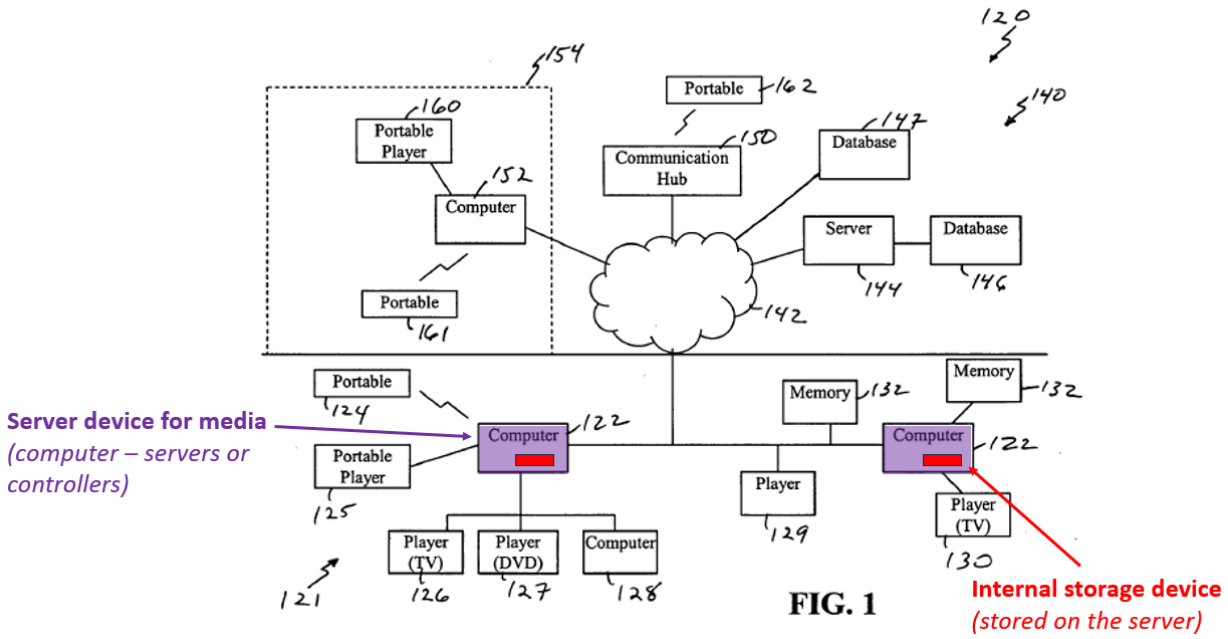


FIG. 1

EX1004 (Lamkin), FIG. 1 (annotated and modified to include internal storage device)

224. As I describe in §VIII.A above, a POSITA would have understood that it was well-known that computers (such as server 122) have internal storage including RAM (volatile storage) and hard disks (non-volatile storage) for storing data and other digital contents.

225. Other references, including some used during prosecution, confirm this. EX1013 (Lambert), [0033] ("drives and their associated computer storage media… provide storage of computer-readable instructions, data structures, program modules and other data for the computer 110"), [0031] (describing that computer storage media includes "volatile and/or nonvolatile memory such as read only memory (ROM) 131 and random access memory (RAM)" where "basic routines that help to

transfer information between elements within computer 110… is typically stored in ROM [and] RAM 132 typically contains data and/or program modules"); EX1014 (Dye), [0003] ("Since their introduction in 1981, the architecture of personal computer systems has remained substantially unchanged [and the] current state of the art in computer system architectures includes a central processing unit (CPU) which coupled to a memory controller interface that in turn couples to a system memory"), [0044]-[0046] (describing that computer system architectures include "programs and data … generally stored on the hard disk 120").

226. Accordingly, a POSITA, understanding the basic memory structures common to computers and servers, would have understood from Lamkin's description of storing data in or on the server that Lamkin's server includes "an internal storage device for storing digital contents" as claimed.

227. **In sum**, regarding claim limitation [1a-i], Lamkin discloses or renders obvious "an internal storage device [RAM of Lamkin's server 122] for storing [stored on Lamkin's server 122] digital contents [Lamkin's audio/video content]."

228. For similar reasons, Lamkin also discloses claim limitation [6a-i] and corresponding features of claim limitations [7a] and [12a]. *See* §I above (Table of Claims).

3. Claim [1a-ii]: wherein the server device for media responds to a data transmission request from a network player by stream-delivering corresponding data in corresponding digital contents from the internal storage device to the network player during connection to a network

229. Lamkin discloses claim limitation [1a-ii].

230. Lamkin describes that the "**media server** stores, retrieves and **provides media on request**." EX1004 (Lamkin), [0241], [0049] ("network allows communication of content, such as multimedia content, between components of the network"), [0013] (describing FIG. 2 as demonstrating the "**pulling of content from a media server 22 to a client device or media renderer**"), [0058] (describing FIG. 3 as demonstrating the "**pushing of content between a media server 122 to a client device or media renderer 124**"), [0064] ("Content can be distributed through the network 120 … for many reasons, such as … user or client device demands"), [0286], FIGS. 1-3, 27.

231. As shown in annotated FIG. 1 below, Lamkin's system includes a computer 122 (server device for media, colored purple) connected by a network 140 (colored green) to players 124-130, 152, and 160 (network players, colored blue).

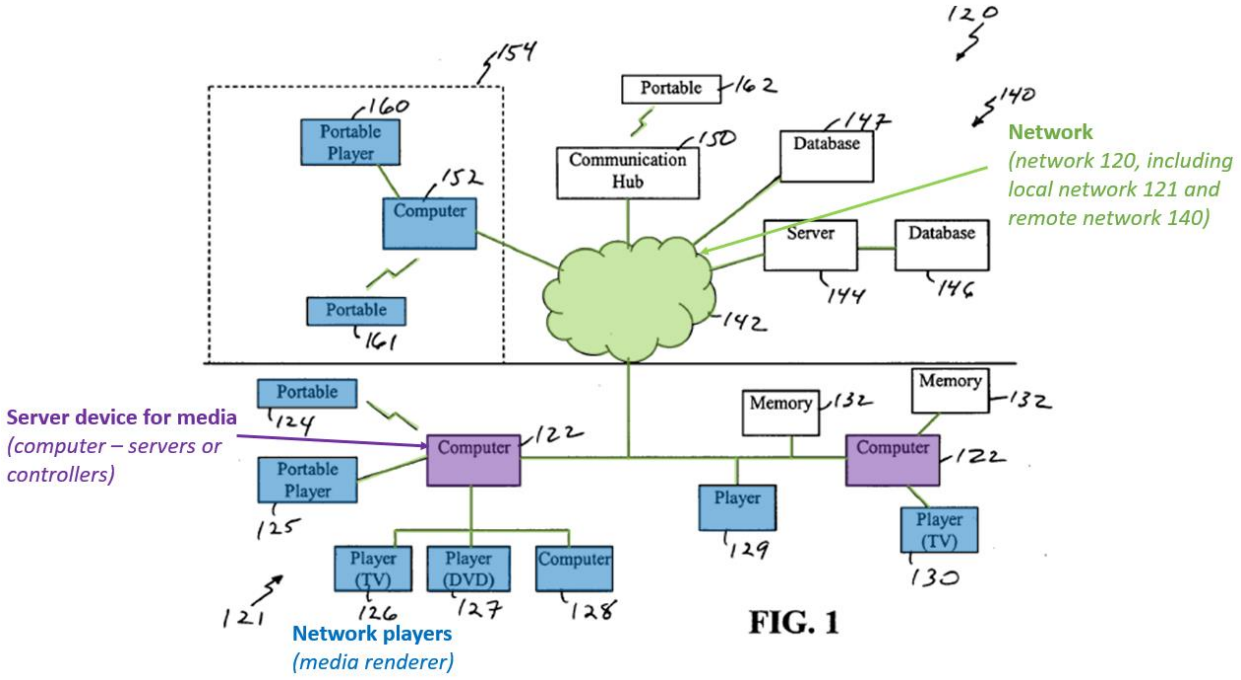


FIG. 1

EX1004 (Lamkin), FIG. 1 (annotated)

232. Lamkin describes both "push" and "pull" schemes for distributing data in a system. EX1004 (Lamkin), [0056]-[0058], FIGS. 2-3 (reproduced below). For example, in "the pulling of content from a media server 122 to a client device or media renderer 124," "a user 224 at the client device 124 is provided a user interface that accesses and/or shows content accessible… through a CDS [content directory service] from various devices and/or local content, and allows the user to select the content to be pulled to the client device to be accessed and/or recorded." *Id.*, [0056], FIG. 2. "The user typically selects and/or sets configuration values, and the selected content is pulled over the network 120 to be recorded or burned." *Id.*, [0056], FIG. 3. Lamkin's description of the user selection of content to be pulled from the server through a user interface in the pull scheme provides the claimed "data transmission

request from a network player" to which the server 122 responds. Moreover, a POSITA would have recognized that push/pull are the same as transmit/receive of networked content between clients and servers. *See* §VIII.D above.



FIG. 2

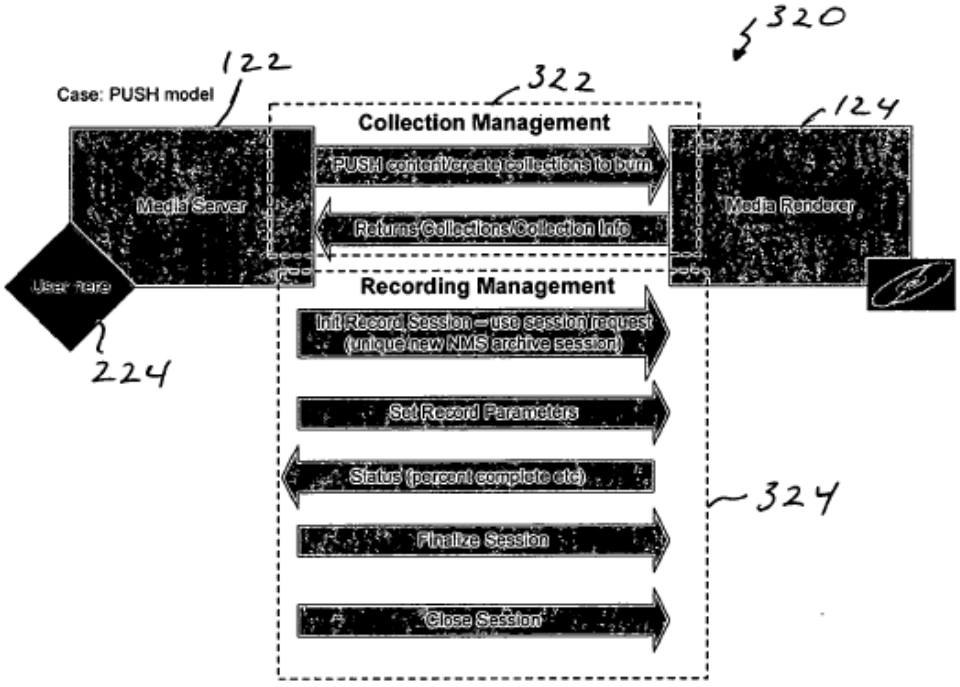


FIG. 3

EX1004, FIGS. 2 and 3

233. Lamkin further describes that content is distributed in response to user requests in the system (*see e.g.*, EX1004 (Lamkin), [0058], [0078], [0083], [0230]-[0231], [0241]), and a POSITA would have understood from Lamkin's disclosure that, in a pull scheme, a server transmits content to a client device upon receipt of an instruction from the client device to do so. *See e.g.*, EX1008 (Yuasa), [0111] ("reproduce content upon receipt of a user operation"). Lamkin thus describes that the server ("server device for media") receives a user-initiated request for content from the client device ("responds to a data transmission request from a network player") and provides requested content ("delivering corresponding data in corresponding digital contents from the internal storage device") to the user at the client device ("network player").

234. Lamkin describes that the server 122 and client devices (*e.g.*, 124-130, 152, 160-161) are connected by a network 120 including local network 121 and remote network 140. EX1004 (Lamkin), [0049]. Lamkin describes that the content is transmitted from the server to the client device over the network in a pull scheme (or a push scheme for that matter). A POSITA would have understood that because Lamkin's network 120 "allows communication of content… between components of the network," the requested content can only be delivered to the client device when the server and the client device are connected to the network ("delivering

corresponding data in corresponding digital contents from the internal storage device to the network player during connection to a network"). *Id.*

235. A POSITA would have understood from Lamkin's discussion of acceptable types of client devices including "wireless devices 128" and acceptable communication networks including a "wireless network" that Lamkin's server is capable of stream delivery of content to such devices. EX1004 (Lamkin), [0049], [0051], [0174] ("content can be received on a portable device, such as a wireless and/or cellular phone"). Accordingly, Lamkin describes that the server ("the server device for media") responds to a user-initiated request ("responds to a data transmission request from a network player") by delivering requested content by real time streaming ("by stream-delivering corresponding data in corresponding digital contents") from the server to the media renderer ("from the internal storage device to the network player") both connected within a network ("during connection to a network"). *See* EX1004 (Lamkin), [0049], [0055]-[0058], [0064], [0160], [0241], [0286], [0056].

236. **In sum**, regarding claim limitation [1a-ii], Lamkin discloses "the server device for media [Lamkin's computer or server 122] responds to a data transmission request from a network player [reproduce content upon receipt of a user operation, per Lamkin] by stream-delivering [over Lamkin's wireless network] corresponding data in corresponding digital contents [reproducing selected content, per Lamkin]

from the internal storage device [from RAM of Lamkin's computer or server 122] to the network player [Lamkin's players 124-130, 152, 160-161] during connection to a network [distributed through the network 120, per Lamkin]."

237. For similar reasons, Lamkin also discloses claim limitation [6a-ii] and corresponding features of claim limitations [7a] and [12a]. *See* §I above (Table of Claims).

4. Claim [1b-i]: a transfer control unit adapted to transfer and store part of held digital contents in the internal storage device to a network storage device

238. Lamkin discloses claim limitation [1b-i].

239. For example, Lamkin describes that "content management can be implemented in part through one or more agents and/or bots from which services are requested and the agent performs the task" of transferring content from the server to other memory or remote storage to store. EX1004 (Lamkin), [0220], [0217]-[0220], [0127], [0166], FIG. 26.

240. Notably, the '101 patent does not identify particular algorithms or software for performing the functions of the "transfer control unit" and other claimed "units" and instead labels each 'unit' based on its assigned function and describes merely that a computer performs the functions based on a program or instructions. *See* EX1001 ('101 patent), 3:5-9.

241. Lamkin's "content management service" performs the same function as the claimed "transfer control unit" and the implementation of Lamkin's "content management service" using agents or bots means that the content management service is a software-implemented unit—the same as the '101 patent's "transfer control unit." *Compare* EX1004 (Lamkin), [0220] ("content management can be implemented in part through one or more agents and/or bots from which services are requested and the agent performs the task"), [0077] ("**content is transferred and/or stored to one or more storage locations** for centralization, such as on the server, **in the local storage 132, remote storage 146-147, or other such storage**"), *with* EX1001 ('101 patent), 3:5-9 ("The program … causes a computer to function as each means of the above-described server device for media"), 6:28-40 ("transfer control means 41 transfers part of held digital contents in the internal storage device to the network storage device 57").

242. As I described in §VIII.B.1 above, a POSITA would have appreciated that an "agent" or "bot" is simply a computer-implemented software program to automate tasks—a way to "cause[] a computer to function" in a particular way to achieve a task. EX1001 ('101 patent), 3:5-9 ("The program according to the present invention causes a computer to function as each means of the above-described server device for media"). Accordingly, a POSITA would have appreciated that Lamkin's

"content management service" implemented as an agent or bot is software implemented in a computer, just as the '101 patent's "transfer control unit" must be.

243. As shown in annotated FIG. 26 below, Lamkin describes a content management service 2640 (highlighted in orange) and a related content management application 2664. Lamkin's content management service 2640 ("transfer control unit") and related application 2666 distributes content over the network for storage. *See* EX1004 (Lamkin), [0220] ("**content management service 2640** at least in part **performs active management of the content and/or files on the network**, distributes content over the network, tracks content on the network, **initiates archiving of content** …").

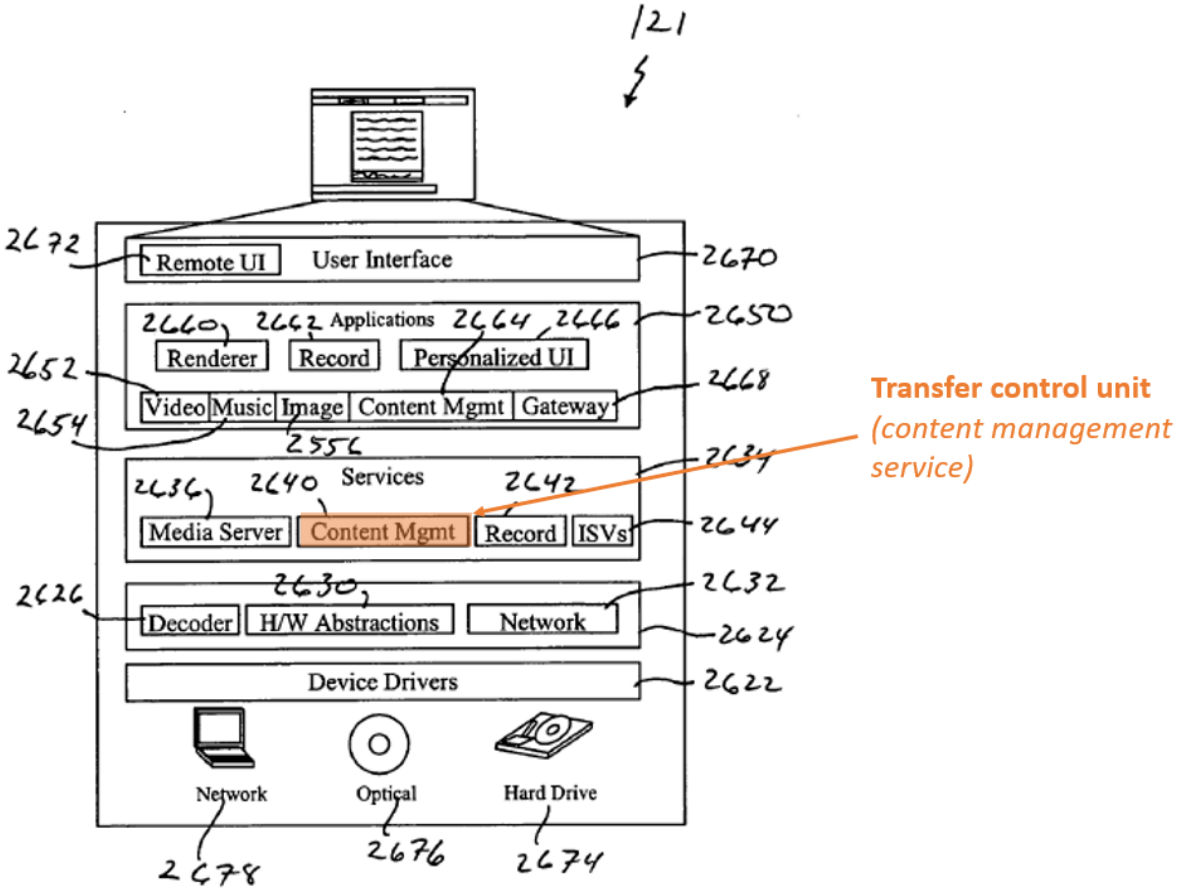


FIG. 26

EX1004 (Lamkin), FIG. 26 (annotated)

244. As described above, the type of tiered storage architecture described by Lamkin (and referred to in the claims of the '101 patent) was well-known and commonly practiced by the Critical Date. *See* §VIII.D.1 above. For example, it was a common storage technique for media content for a server device (which might itself be a storage device) to receive content and, using software like Lamkin's content management service, to distribute the content to one or more backup devices for reasons such as preserving or optimizing space on the server, moving content to

where the user is likely to access it, or providing redundant backup copies of content. *See e.g.,* EX1008 (Yuasa), [0005] ("the receiving apparatus receives and accumulates the broadcast contents in a storage medium such as a large-capacity hard disc, so that the contents can be displayed or reproduced when requested by the user"), [0016] ("a transferring unit for transferring the designated content from the first storage medium to the second storage medium before the designated content is deleted").

245. A POSITA would have understood that Lamkin's content management service transfers content held in the server to storage on the local storage 132, remote storage 146-147, or other such storage based on Lamkin's description of the content management service distributing content and initiating archive of content on the network. EX1004 (Lamkin), [0220], [0077], [0141]. As shown below in annotated FIG. 1, Lamkin's system includes computer 122 (server device for media, colored purple) coupled to memories 132 in the local network and to databases 146-147 in the wide area network (network storage devices, colored yellow).

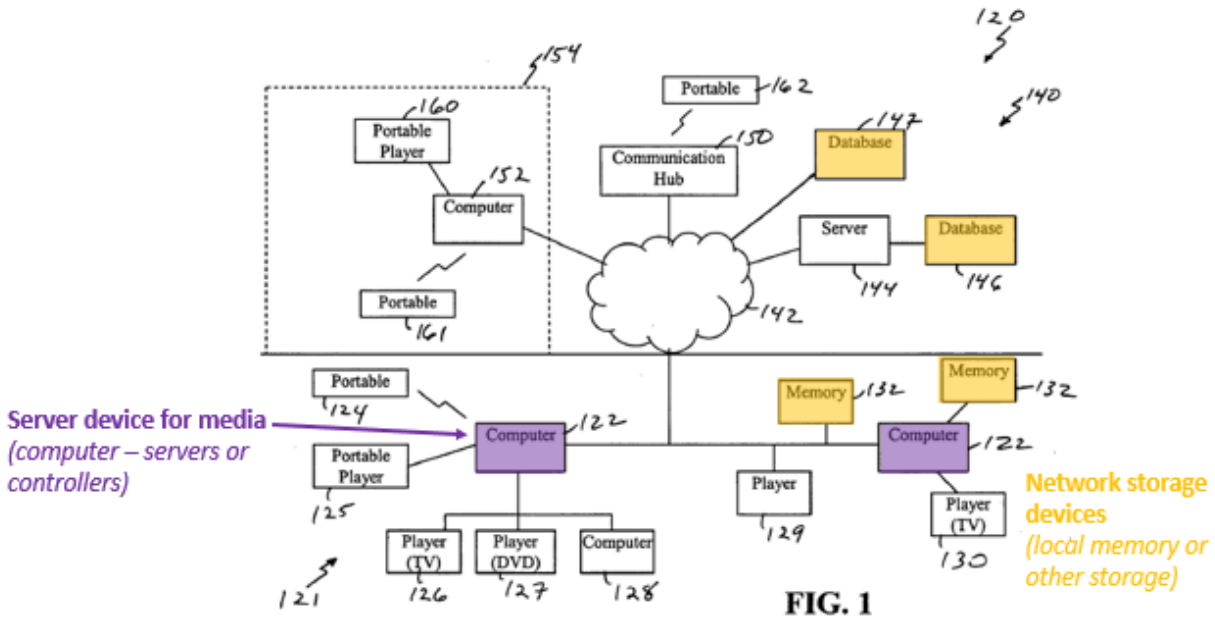


FIG. 1

EX1004 (Lamkin), FIG. 1 (annotated)

246. For example, Lamkin explicitly states that "[t]he **management of the content can move content from being stored on** local client devices 124-130 and/or **servers 122 to local storage devices 132, and/or to remote storage 146-147**." EX1004 (Lamkin), [0141]; *see also id.*, ("predictive archiving can copy, move or otherwise distribute the content to storage devices for backup, free-up local storage space and other such factors"), [0144], [0146], [0219]-[0220]. Lamkin discloses that the content management service ("transfer control unit") transfers at least a portion of the contents from the internal storage of the server to other memory 132 or remote storage 146-147 for storage ("adapted to transfer and store part of held digital contents in the internal storage device to a network storage device"). *See* EX1004 (Lamkin), [0078], [0141].

247. **In sum**, regarding claim limitation [1b-i], Lamkin discloses "a transfer control unit [Lamkin's content management services] adapted to transfer [move content, per Lamkin] and store [store content, per Lamkin] part of held digital contents [Lamkin's content, such as audio/video content] in the internal storage device [RAM of Lamkin's computer or server 122] to a network storage device [Lamkin's local memory 132 or remote database storage 146-147]."

248. For similar reasons, Lamkin also discloses claim limitations [6b-i], [7b-i], and [12b-i]. *See* §I above (Table of Claims).

5. Claim [1b-ii]: wherein the network storage device is connected to the network and is capable of storing data

249. Lamkin discloses claim limitation [1b-ii].

250. For example, Lamkin describes that local memories 132 are part of the local network 121 and the remote memories 142 are part of the remote network 140. EX1004 (Lamkin), [0049].

251. Lamkin describes that "[t]he one or more local servers, computers or other control devices 122 provide control over the local network 121 and/or remote network 140, communicate with the client devices 124-130 and storage or other memory 132 (e.g., database, network attached storage (NAS), and other such storage) that store content." *Id.*, [0049], [0051] ("The remote network 140 further includes one or more servers 142, storage devices or databases 144-145"), [0052]

("Content can be retained, recorded and/or stored throughout the local network 121 (and/or remote network 140)."), [0069].

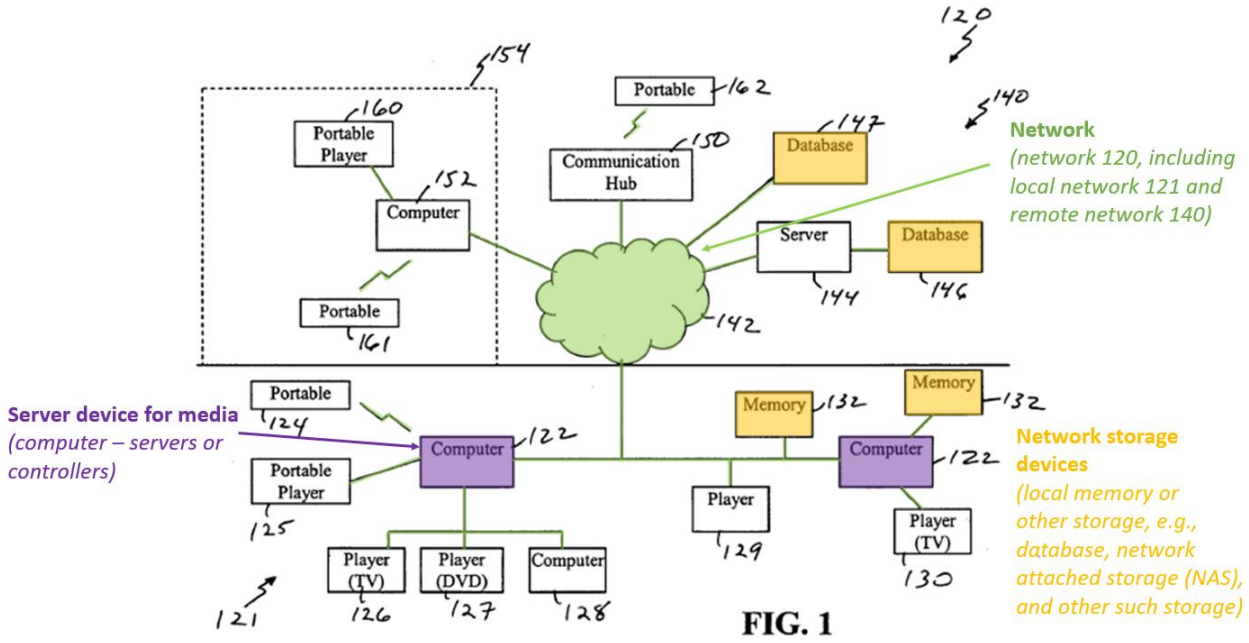


FIG. 1

EX1004 (Lamkin), FIG. 1 (annotated)

252. As shown above in annotated FIG. 1, Lamkin's other memory 132 or remote database storage 146-147 ("network storage device"), colored yellow, are accordingly accessible via the network ("connected to the network") which is colored green and includes local network 121 and remote network 140, and able to receive and store data ("capable of storing data") when Lamkin's content management service moves content from the server ("server device for media"), colored purple, for storage on the other memory 132 or remote storage 146-147. *See e.g.*, EX1004 (Lamkin), [0141], [0051].

253. **In sum**, regarding claim limitation [1b-ii], Lamkin discloses "the network storage device [Lamkin's local memory 132 or remote storage 146-147] is connected to the network [Lamkin's network 120] and is capable of storing data [are devices that store content, per Lamkin]."

254. For similar reasons, Lamkin also discloses claim limitations [6b-ii], [7b-ii], and [12b-ii]. *See* §I above (Table of Claims).

6. Claim [1b-iii]: wherein said transfer control unit does not transfer, from the internal storage device to the network storage device, the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device.

255. Lamkin renders obvious claim limitation [1b-iii].

256. Lamkin describes that, in some situations, properties of content selected for transfer are relevant to whether the system will transfer the content; these properties make the content at risk of being lost or corrupted if a network failure occurred during the transfer ("digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device"). *See* §VIII.B.2 above. In such cases, Lamkin's system does not transfer certain contents because properties of the content that increase risk of loss of the content during transfer—for example large file size or certain copyright protections associated with the content. *See* EX1004 (Lamkin), [0074], [0127], [0220].

257. As I explain in §VIII.B.2 above, there are multiple reasons for network issues that may lead to loss of data. Thus, it was conventional for storage and transfer systems to include mechanisms to protect against data loss, just as Lamkin does.

258. A POSITA would have understood that, in networking, network transmissions (*e.g.*, Lamkin's content) are commonly broken up into parts to be transmitted and reassembled at the destination. *See* §VIII.B.2 above; *see e.g.*, EX1004 (Lamkin), [0082] ("the cache and buffer management additionally or alternatively determine when and/or how to transfer content and files based on network bandwidth utilization to improve transfer rates, for example, by transferring files in off hours or at lower bandwidths"), [0160] (in a "Progressive" download, "a file can be segmented into portions or chunks, and HTTP GET is applied one chunk at a time" and "[i]n those instances where digital rights management (DRM) is applied to some or the entity body of the HTTP GET, some embodiments concatenate the collection of entity bodies to restore a full media stream, then convert the stream for storing"), [0105]. At least in those instances where transmissions are broken up into packets of equal size, the number of packets that need to be sent increases with the size of the file or content to be transmitted. *See* §VIII.B.2 above.

259. As illustrated below in annotated FIG. 4, in Lamkin's system, if the file size of the content to be transmitted exceeds a threshold size (see step 430), the content is not transferred (bypasses step 446). EX1004 (Lamkin), [0074] ("the process continues to … step 430 to determine whether the size or amount of new content exceeds a content threshold… [w]hen step 430 is performed and it is determined that the content exceeds the threshold, the process continues to step 450"), [0077] ("In step 450, a log file, database listing, CDS or other tracking is updated identifying the new content"), FIG. 4 (annotated below).

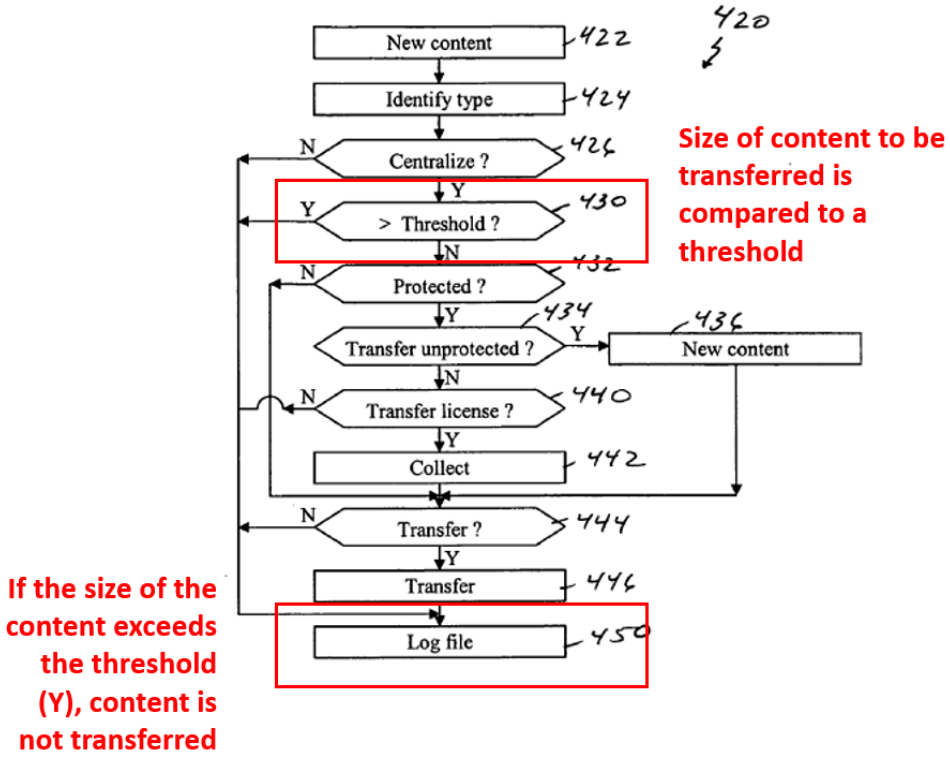


FIG. 4

EX1004 (Lamkin), FIG. 4 (annotated)

260. Accordingly, a POSITA would have understood that digital contents that are broken into a high number of chunks for transfer are more likely to be unrecoverable if a network failure occurs during the transferring of the digital contents from the server to the local/remote storage because the large file takes longer to transfer and includes a larger number of segments that are individually transferred, and that Lamkin describes not transferring such large files. *See* §VIII.B.2 above; EX1004 (Lamkin), [0072]-[0077], FIG. 4.

261. I note that the claims do not require that the unrecoverability of digital contents be related to DRM status and none of the claims mention "DRM." Nonetheless, I provide an alternative mapping based on Lamkin's disclosure of not transferring content with protected status.

262. Lamkin's transfer of content subject to digital rights management ("DRM") considerations, offers an alternative mapping, including preventing the transfer of content that is protected. EX1004 (Lamkin), [0055], [0067] ("some content may be content protected and thus unable to be distributed or properly distributed without further authorization"), [0075]-[0076], [0107] ("When it is determined that the content is not to be further distributed, the process 720 terminates"), [0114], [0117] ("In step 828, it is determined whether content is protected content [and if] the content is protected, step 830 is entered where it is

determined whether the content and/or any license can be transferred"), [0006], [0123], [0173], [0237], FIG. 8.

263. For example, Lamkin describes "some **protected files** are streamed for playback and **are not allowed to be stored locally for later playback**." *Id.*, [0055]. As I described in §VIII.E.1 above, copy protection schemes were well-known by the Critical Date, and were often used to prevent unauthorized transfer of protected contents, just as Lamkin describes.

264. Other references confirm this understanding. *See e.g.*, EX1016 (Kawasaki), [0009] ("if contents are transferred from the storage unit of a PC to the content reproducing apparatus such as a portable device as outlined above, then the contents not allowed to be reproduced by the content reproducing apparatus under copyright will also be transferred to the apparatus for unauthorized reproduction"), [0047]-[0049], [0055] ("the reproduction of content data is controlled on the basis of a source ID assigned uniquely to each of the content data providers involved"); EX1023 (Jaisimha), Abstract ("The media player requests the media file from a media server [in response to which the] media server transmits the header of the media file, and the media player extracts the access code to determine whether the media file may be transmitted according to the desired type of transmission. If so, the media server transmits the media file using the desired type of transmission, and the media player renders the media data into video, sound or image signals").

265. Next, Lamkin describes using connectivity tests to monitor network connections and report information to the user; Lamkin also discloses a configuration wizard that "can further indicate a loss of connectivity and reselect a wizard to go through the process again." EX1004 (Lamkin), [0240], [0082], [0105], [0254], [0290]. A POSITA would have understood Lamkin's discussion of network connection monitoring as an indication that Lamkin is concerned about the effects of network failure on transfers within the system and recognized that any network is susceptible to failures and the issues that accompany such failures. *See* §VIII.B.2 above.

266. **In sum**, regarding claim limitation [1b-iii], Lamkin discloses or renders obvious "said transfer control unit [Lamkin's content management service] does not transfer [bypasses transfer step, per Lamkin], from the internal storage device [RAM of Lamkin's computer/server 122] to the network storage device [Lamkin's local memory 132 or remote storage 146-147], the digital contents [Lamkin's contents including audio/video content] that cannot be recovered [Lamkin's content that has too large a file size or is protected] if a network failure [Lamkin's loss of connectivity] occurs during the transferring [during distribution, per Lamkin] of the digital contents [Lamkin's contents including audio/video content] from the internal storage device [RAM of Lamkin's computer/server 122] to the network storage device [Lamkin's local memory 132 or remote storage 146-147]."

267. For similar reasons, Lamkin also discloses claim limitation [7b-iii]. *See* §I above (Table of Claims).

7. Claim [1c-i]: a list information transmission unit adapted to respond to a list presentation request for the held digital contents of the server device for media from the network player by transmitting list information to the network player

268. Lamkin discloses claim limitation [1c-i]. As an initial matter, a POSITA would have known that listing files and transmitting such lists was well known in the art and was commonplace by the Critical Date. *See* §VIII.C.1 above.

269. As noted above with regard to the transfer control unit (see §XI.A.4 above), the '101 patent does not identify particular algorithms or software for performing the functions of the claimed "units" and instead labels each 'unit' based on its assigned function and describes merely that a computer performs the functions based on a program or instructions. *See* EX1001 ('101 patent), 3:5-9.

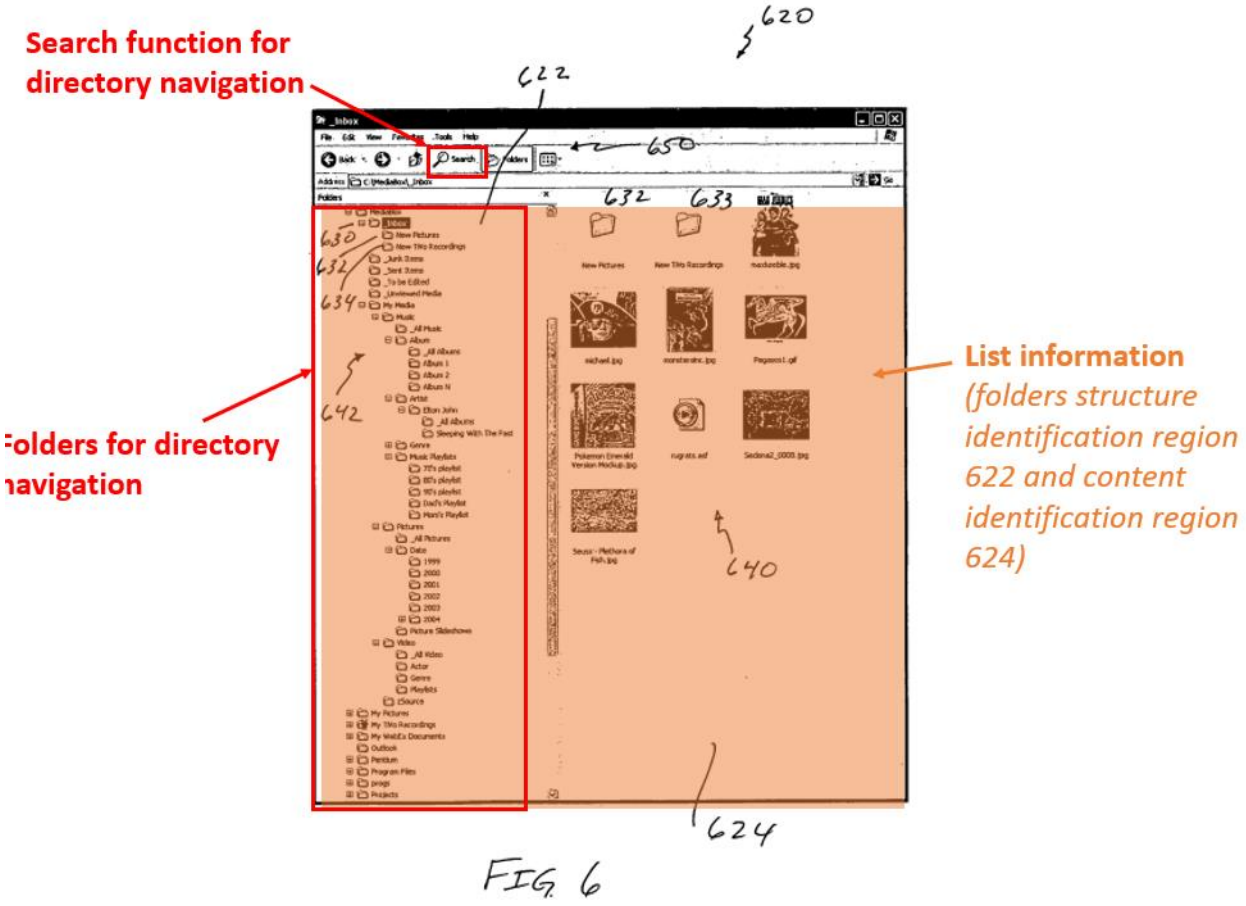
270. Lamkin describes that a content directory service (*e.g.*, a UPnP CDS) ("list information transmission unit") identifies a list of available files ("list information") to a user in response to a user request to access a list of available content (*e.g.*, accessing a list of available content on a user display) ("respond to a list presentation request for the held digital contents") and provides the list to the user device ("transmitting list information to the network player"). *See* EX1004 (Lamkin), [0068] ("Some embodiments further **map the content structure (e.g., DVD content structure) into a UPnP CDS, other listing or database so that it**

can be easily appended and accessed while mounted"), [0077] ("database listing, CDS"), [0054] ("A database, listing or other methods can be employed… [i]n implementations employing UPnP[,] a content directory service (CDS) of a device can be used to identify and/or display the available files and/or associated metadata"), [0087] ("The listing 522 includes a listing of one or more content that can potentially be distributed over the local network 121 and/or remote network 140"), [0266] ("The database can create the structure upon the request of multiple clients"), [0088], [0262], [0296] ("the user has… the option to browse the live server as necessary"), FIGS. 5-6, 8, 27, 31.

271. A POSITA would have understood that modern file systems include a directory to allow a user to understand the contents of a computer and attached storage devices and to direct the computer to take actions so that the user can engage with the contents. *See* §VIII.C.1 above. A POSITA would have further understood that Lamkin's content directory service provides a directory (*e.g.*, a database or listing) of content and software for organizing and displaying the list of contents to the user. *See* EX1004 (Lamkin), [0054]; *see also* §VIII.E.2 above.

272. Lamkin describes user interaction with the interface, including the ability to select content, stating that "[t]he user interface 520 can include a listing of content 522 and optional commands 524 that are selectable by the user." *Id.*, [0087]. A user accessing the listing of content or the media inbox is shown in annotated

FIG. 6 below: this is a user request to view the listing ("respond to a list presentation request for the held digital contents of the server device for media from the network player").



EX1004 (Lamkin), FIG. 6 (annotated)

273. A POSITA would have understood that the user can navigate through the contents listed in the display (*e.g.*, of Lamkin's FIG. 6) by selecting folders along the left-hand side of the user interface (annotated in Lamkin's FIG. 6 above with red box and text) or using search functions at the top of the user interface (also annotated in Lamkin's FIG. 6 above with red box and text), and that different views of the

contents held by the server are available depending on the user's selection. *Id.*, [0095]-[0097].

274. **In sum**, regarding claim limitation [1c-i], Lamkin discloses "a list information transmission unit [Lamkin's content directory service] adapted to respond to a list presentation request [user request to view a listing of content, per Lamkin] for the held digital contents of the server device for media [content accessible by computer 122, per Lamkin] from the network player [Lamkin's user interface of the player] by transmitting [displaying on the user interface, per Lamkin] list information [Lamkin's listing of content] to the network player [Lamkin's players 124-130, 152, 160-161]."

275. For similar reasons, Lamkin also discloses claim limitations [6c-i], [7c-i], and [12c-i]. *See* §I above (Table of Claims).

8. Claim [1c-ii]: wherein the list information lists the digital contents left in the internal storage device and the digital contents transferred from the internal storage device to the network storage device and stored in the network storage device

276. Lamkin discloses claim limitation [1c-ii].

277. Lamkin describes that the content directory service ("list information transmission unit") provides a list ("list information") that includes content stored in various devices, including within the server 122 ("digital contents left in the internal storage device") and content stored in local or remote storage ("digital contents

transferred from the internal storage device to the network storage device and stored in the network storage device"). EX1004 (Lamkin), [0303], [0068], [0087].

278. Lamkin's "listing 522 includes a listing of one or more content that can potentially be distributed over the local network 121 and/or remote network 140." EX1004 (Lamkin), [0087], [0113] ("The listing can be a centralized listing, one or more CDSs, one or more device listings, listings accessed over the remote network 142, and/or other such listings"), [0204]-[0205], [0266]-[0268]. A POSITA would have understood from Lamkin's disclosure that the aggregated database provides to the user a single list that includes available content that remains stored in the server 122 and content that is stored in other memories and remote servers, because Lamkin's database is the aggregate of all available content.

279. Other references confirm this understanding. *See e.g.*, EX1025 (Van Den Heuvel), [0044] (explaining that a "CDS provides a uniform mechanism for allowing devices implementing a control point to browse and search the content in the home network and to obtain detailed information about individual content objects"). It would appear to the user of Lamkin's system as if all content were centrally held from the list, regardless of where the content were actually stored.

280. **In sum**, regarding claim limitation [1c-ii], Lamkin discloses "the list information [Lamkin's listing of content] lists the digital contents [Lamkin's contents] left in [stored on, per Lamkin] the internal storage device [RAM of

Lamkin's computer 122] and the digital contents [Lamkin's contents] transferred from [moved from, per Lamkin] the internal storage device [RAM of Lamkin's computer 122] to the network storage device [Lamkin's memory 132 or remote storage 146-147] and stored in [stored in, per Lamkin] the network storage device [Lamkin's memory 132 or remote storage 146-147]."

281. For similar reasons, Lamkin also discloses claim limitation [6c-ii], [7c-ii] and [12c-ii]. *See* §I above (Table of Claims).

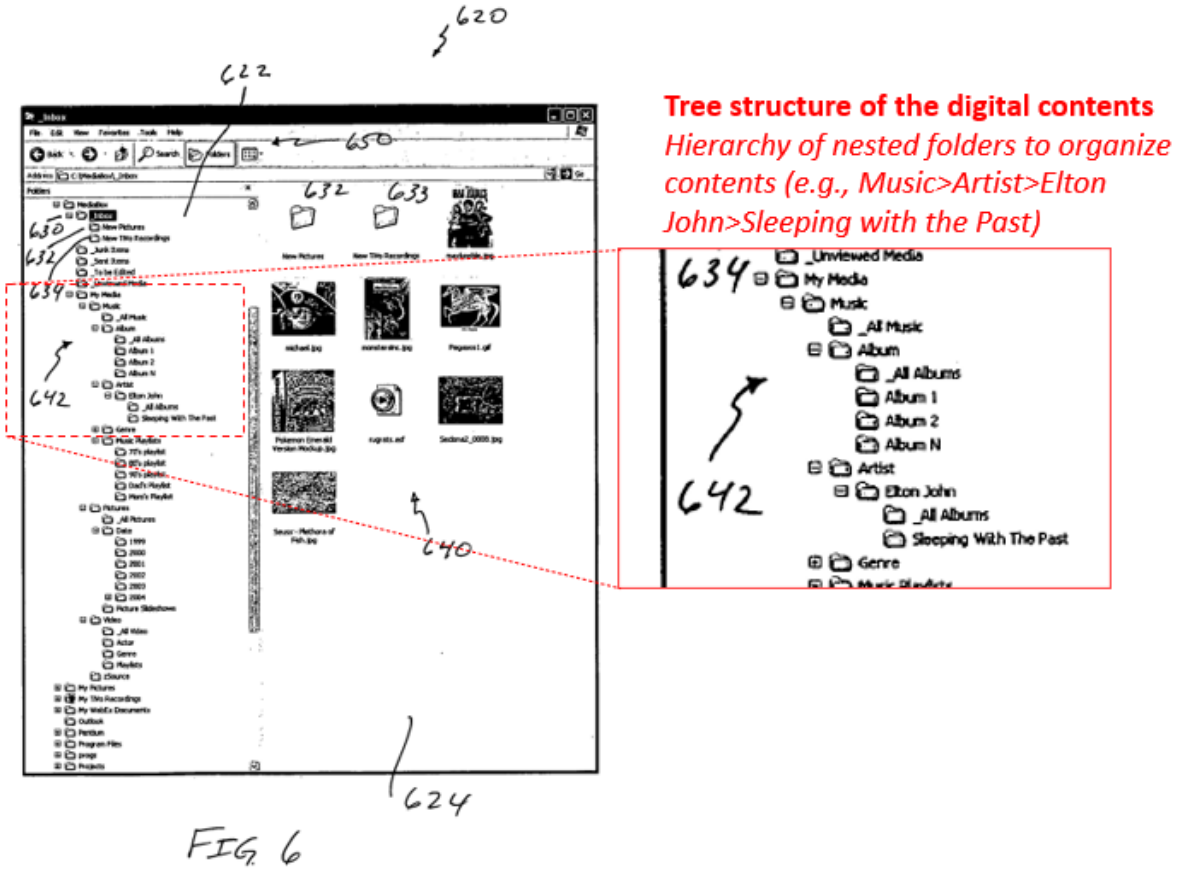
9. Claim [1c-iii]: wherein the list information maintains a tree structure of the digital contents in the internal storage device before transferring the digital contents to the network storage device.

282. Lamkin discloses claim limitation [1c-iii].

283. As an initial matter, a POSITA would have known that file tree structures and hierarchical file systems were well known in the art and commonplace. *See* §VIII.C.1 above.

284. Lamkin describes that the content directory service ("list information transmission unit") provides a content directory list ("list information") arranged in a hierarchical structure of folders ("maintains a tree structure") that list content accessible by the server regardless of where the content is currently stored in the network ("of the digital contents in the internal storage device before transferring the digital contents to the network storage device"). EX1004 (Lamkin), [0264], [0232].

285. For example, Lamkin discloses that content can be listed in the content directory according to "simple naming structures, for example, of 'My Pictures\Year\Month\Day\ Picture Time.<format>' [using an] automated media management system" (EX1004 (Lamkin), [0071]) and that the listing of content can be displayed on a user interface. EX1004 (Lamkin), [0087], [0088] ("The content can be listed by date added to the network or altered, alphabetically by title or identification, by source, content type and/or other such organization [and] the user can alter or select how the content is listed and organized"). A POSITA would have understood that this naming structure is similar to a file pathname indicating where in a tree structured hierarchy that content is located. *See* §VIII.C.1 above. A similar tree structure is shown below in Lamkin's annotated FIG. 6.



EX1004 (Lamkin), FIG. 6 (annotated, inset added)

286. As I described in §VIII.C.1 above, hierarchical file systems—including "tree structures"—were well-known and conventional by the Critical Date. *See also, e.g.*, EX1009 (Kageyama), [0084] ("a tree structure containing a plurality of containers arranged in hierarchical levels… [t]he tree structure… has a 'ROOT' container that forms a first level [below which] a plurality of containers are arranged").

287. A POSITA would have further understood that Lamkin's description of the CDS describes a hierarchy directory systems using nested containers (*e.g.*,

folders), as shown in FIG. 6 (annotated above), as a content directory having a tree structure ("maintains a tree structure of the digital contents"). *See* §VIII.C.1 above; *see also* EX1004 (Lamkin), [0264] ("tree structure"), [0265]-[0272]; EX1009 (Kageyama), [0009] ("The content management server includes a storage unit configured to store meta data of the content items, a tree structure generating unit configured to generate a tree structure including a plurality of containers arranged in hierarchical levels, each container having an identifier representing a type of content assigned thereto"), [0105]-[0109], FIGS. 4A-C, 6-7.

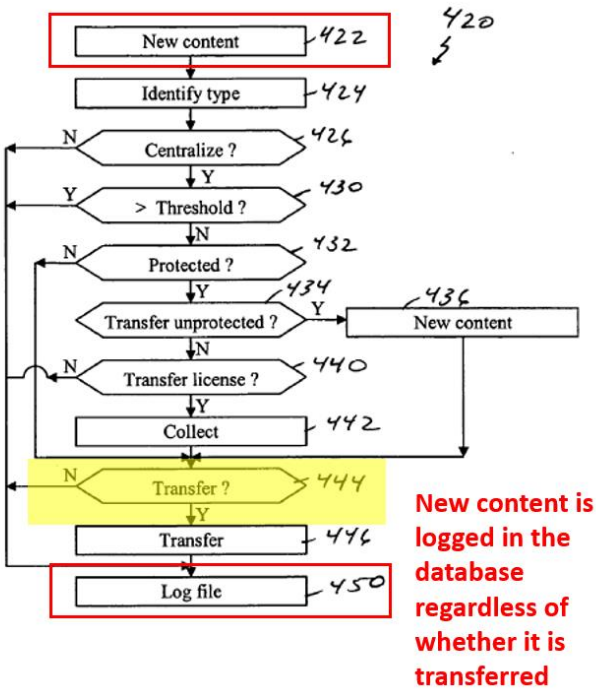


FIG. 4

EX1004 (Lamkin), FIG. 4 (annotated)

288. As depicted in Lamkin's annotated FIG. 4 above, content added to the system is logged in the database regardless of whether it is stored in the server or is

transferred from the server for storage elsewhere. EX1004 (Lamkin), [0077], [0124].

289. **In sum**, regarding claim limitation [1c-iii], Lamkin discloses "the list information [Lamkin's listing of content] maintains a tree structure [forms a tree structure, per Lamkin] of the digital contents [Lamkin's content such as media files] in the internal storage device [RAM of Lamkin's computer or server 122] before transferring [moving, per Lamkin] the digital contents [Lamkin's content such as media files] to the network storage device [Lamkin's memory 132 or remote storage 146-147]."

290. For similar reasons, Lamkin also discloses claim limitations [6c-iii], [7c-iii], and [12c-iii]. *See* §I above (Table of Claims).

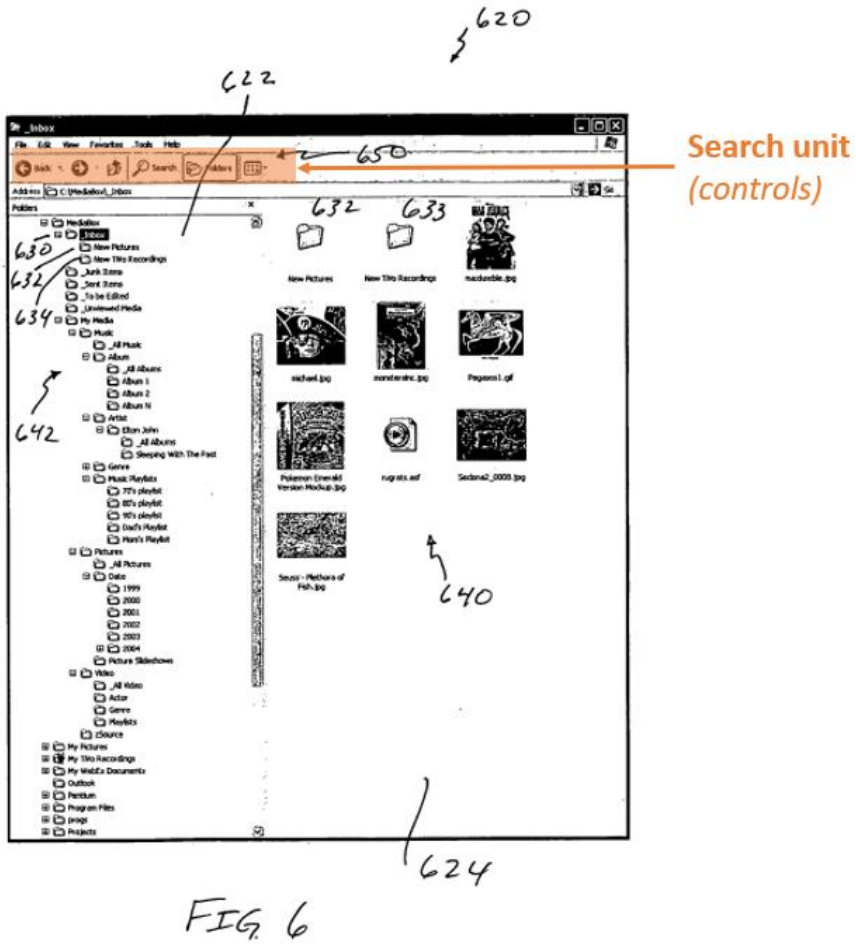
10. Claim [1d]: a search unit adapted to respond to a data transmission request for the held digital contents from the network player by searching for a location where the held digital contents are currently stored

291. Lamkin discloses claim limitation [1d].

292. The '101 patent does not identify particular algorithms or software for performing the functions of the claimed "units" and instead labels each 'unit' based on its assigned function and describes merely that a computer performs the functions based on a program or instructions. *See* EX1001 ('101 patent), 3:5-9.

293. Lamkin describes that a user is able to search available content including using "controls 650," shown in orange in annotated FIG. 6 below, and that

the server uses a universal locator to identify a location of requested content. EX1004 (Lamkin), [0097, [0131], FIGS. 6, 28. As a POSITA would have understood, controls are typically a name used for computer-implemented software that allows a computer to perform tasks or functions, such as searching within a server database to identify a location of a resource or content. *See* §VIII.A above; *see e.g.*, EX1001 ('101 patent), 3:5-9, 6:41-48. In this case, Lamkin's controls allow the server to search within the content database provided by the server, and locating content selected by the user with the universal locator. EX1004 (Lamkin), [0131].



EX1004 (Lamkin), FIG. 6 (annotated)

294. As illustrated in Lamkin's annotated FIG. 6 above, the controls 650 include buttons allowing a user to perform a search or to search by navigating through folders. *See id.*, FIG. 6. A POSITA would have understood from Lamkin's disclosure that the controls 650, though labeled in the user interface of FIG. 6, allow for content to be searched within the server database (not just on the user interface). *See id.*, [0097] ("The controls 650 can allow the user to perform searches for content, navigation through other folders and/or storage systems and other such controls").

295. Indeed, Lamkin describes that a "user has… the option to **browse the live server as necessary**." *Id.*, [0296]. A POSITA would have appreciated that it was typical for controls such as Lamkin describes to perform a search for a location of selected content, because such search methodology was well-known and conventional. *See* §VIII.C above; *see also e.g.*, EX1024 (Belknap), 6:64-7:4 ("After receiving the request to retrieve … the media object, the object server 220 will identify whether the media object is located locally within the object store 222 or at a remote location within the media archive 140 or one of the media servers 130").

296. Because the user of Lamkin's system can select content without knowing where the content is located, a POSITA would have understood that it is common sense that the server includes software to identify the location of the selected content so that it can be delivered to the user. *See* §§VIII.A–VIII.B above. Lamkin clearly describes that the content selected by the user from the aggregate view (including all content regardless of storage location) is located in a network-connected storage so that it can be provided to the user, and is located by using a universal locator associated with content. *See* EX1004 (Lamkin), [0131]. For example, Lamkin describes that "some metadata can include a universal locator and/or a universal locator can be generated and added to the content [and the] universal locator can be utilized in the aggregate view to, in part, access and/or locate content, as well as building collections." *Id*. "When a user attempts to access

content through the aggregate view, the universal locator can be utilized to direct the user to and/or retrieve the content." *Id*.

297. To provide additional explanation of how content is located in the network, Lamkin incorporates by reference another patent publication listing Lamkin as an inventor (EX1015; Lamkin publication incorporated by reference in Lamkin Patent, referred to herein as "Lamkin2") which further describes "[t]he use of metadata and/or the universal location." *See* EX1004 (Lamkin), [0131]. For example, Lamkin2 describes a content search engine which "**searches various levels for content**, for example, local storage, removable storage, trusted peer network, and general Internet access." EX1015 (Lamkin2), [0201]. This is likely similar to how Lamkin's server would operate to use the universal locator for locating content in networked storage. Whether accomplished by a content search engine or controls using software ("search unit"), Lamkin describes that after a user has selected content, "desired content is searched for and located." EX1004 (Lamkin), [0236].

298. **In sum**, regarding claim limitation [1d], Lamkin discloses "a search unit [controls using Lamkin's universal locator] adapted to respond to a data transmission request [a user attempt to access content, per Lamkin] for the held digital contents [Lamkin's content stored in the network] from the network player [Lamkin's players 124-130, 152, 160-161] by searching [utilizes Lamkin's universal

locator] for a location [to identify the device, per Lamkin] where the held digital contents [Lamkin's content stored in the network] are currently stored [where the content is located, per Lamkin]."

299. For similar reasons, Lamkin also discloses claim limitations [6d], [7d], and [12d]. *See* §I above (Table of Claims).

11. Claim [1e]: a digital contents data transmission processing unit adapted to allow the corresponding data in held digital contents to be stream-delivered from the network storage device to the network player, if the result of search shows the network storage device

300. Lamkin discloses claim limitation [1e].

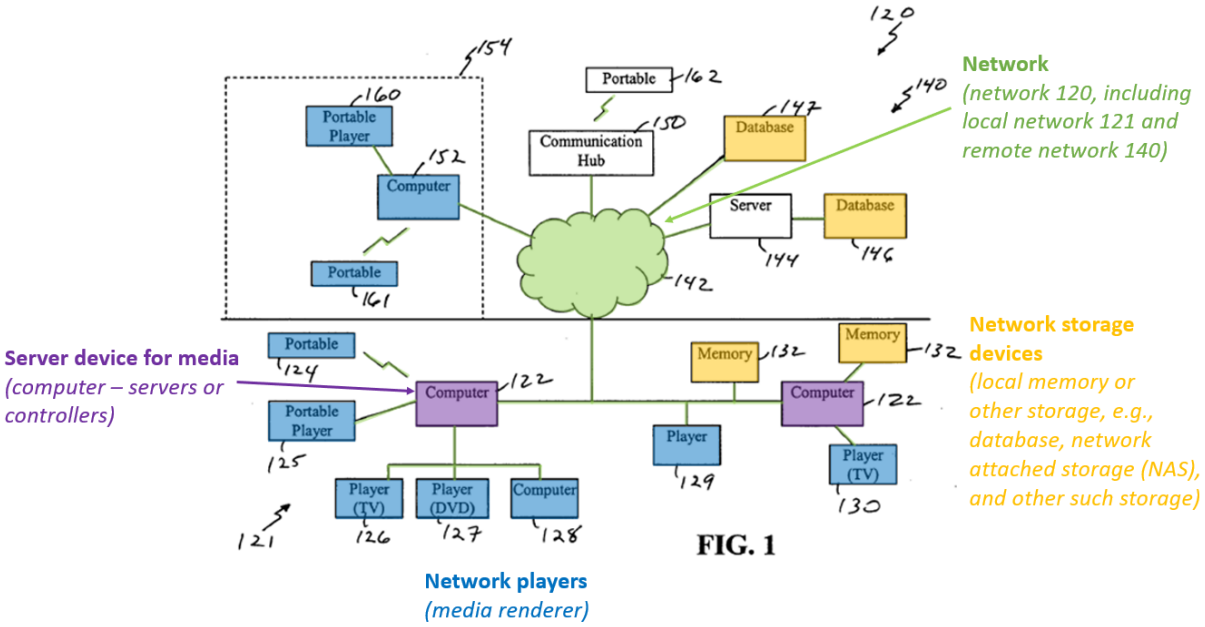
301. The '101 patent does not identify particular algorithms or software for performing the functions of the claimed "units" and instead labels each 'unit' based on its assigned function and describes merely that a computer performs the functions based on a program or instructions. *See* EX1001 ('101 patent), 3:5-9.

302. Lamkin describes an AV (audio/video) transport service ("a digital contents data transmission processing unit") that controls the transfer of the content once it has been located—in the server or in local/remote storage. EX1004 (Lamkin), [0087], [0235]-[0236]. Thus, if the search described in element [1d] (*see* §XI.A.10 above) for a location of desired content shows the content to be located in the local memory 132 or remote database 146-147 ("if the result of search shows the network storage device"), the content in Lamkin is delivered to the user device by streaming ("allow the corresponding data in held digital contents to be stream-

delivered from the network storage device to the network player"). EX1004 (Lamkin), [0160] ("real time streaming protocol (RTSP) streaming"), [0128], [0220], [0231] ("Communication between the server and the client device is over an in-band communication link 2724"), [0241].

303. Lamkin's "AV transport service" performs the same function as the claimed "digital contents data transmission processing unit," and is instantiated as an AV transport algorithm, which a POSITA would have understood is computer-implemented software for performing a transport task. EX1004 (Lamkin), [0236] ("a control algorithm 2820 for an AV architecture"). A POSITA would have understood that an algorithm is a way to "cause[] a computer to function" in a particular way to achieve a task. EX1001 ('101 patent), 3:5-9. Indeed, an "algorithm" is "[a] finite sequence of steps for solving a logical or mathematical problem or performing a task." EX1017 (Microsoft Computer Dictionary), 5.

304. Accordingly, a POSITA would have appreciated that Lamkin's "AV transport," implemented as an algorithm, is software implemented in a computer, just as the '101 patent's "digital contents data transmission processing unit" must be. Lamkin's AV transport service uses the control algorithm to perform functions including transferring content between devices. EX1004 (Lamkin), [0236].



EX1004 (Lamkin), FIG. 1 (annotated)

305. As illustrated above in Lamkin's annotated FIG. 1, Lamkin's computer/server 122 (colored purple) is connected via the network 120 (colored green) to both players 124-130, 152 and 160 (colored blue) and to local memory 132 and remote storage 146-147 (colored yellow). EX1004 (Lamkin), FIG. 1 (annotated above). Lamkin is clear that "the **AV transport service 2734** initiates and **controls the transfer of the located content** using the negotiated transfer protocol." EX1004 (Lamkin), [0236], [0235] ("The **AV transport service 2734** and 2744 **enable control over the transport** of audio, video and/or other **content streams**"). Lamkin's AV transport service allows the computer 122 (purple) to transfer desired content from storage locations (local memory/remote storage or storage on the computer 122) on the network (green) to the players (blue).

306. **In sum**, regarding claim limitation [1e], Lamkin discloses "a digital contents data transmission processing unit [Lamkin's AV transport service] adapted to allow [initiates and controls, per Lamkin] the corresponding data [Lamkin's content such as media files] in held digital contents [Lamkin's content accessible by computer 122] to be stream-delivered [streams, per Lamkin] from the network storage device [Lamkin's memory 132 or remote storage 146-147] to the network player [Lamkin's players 124-130, 152, 160-161], if the result of search [controls use of universal locator associated with requested content, per Lamkin] shows the network storage device [Lamkin's content located in the memory 132 or remote storage 146-147]."

307. For similar reasons, Lamkin also discloses claim limitations [6e], [7e], and [12e]. *See* §I above (Table of Claims).

12. Claim [1f]: wherein the server device for media is a media player.

308. Lamkin discloses claim limitation [1f].

309. Lamkin describes that the server 122 can be a computer, CD player or DVD player. EX1004 (Lamkin), [0241] ("Potential servers include computers, DVD players, CD players, and other such relevant devices"). A POSITA would have understood these types of devices to be media players because they include input/output devices that allow them to play media. Moreover, Lamkin describes that audio playback systems, video playback systems and portable media players can

include compact disc (CD) players and DVD players. *See* EX1004 (Lamkin), [0049] ("the client devices can include one or more computers, televisions, audio playback systems (e.g., a compact disc (CD) player, MP3 players, and the like), video playback systems (e.g., a digital versatile disc (DVD) player), portable media players 127 (e.g., an MP3 player, a portable DVD player, and other such devices)")

310. As I describe in §VIII.A above, there are various characteristics common to computers, including having input/output devices like displays or speakers that enable a user to interact with the computer and play multimedia content. Lamkin's disclosure of the server 122 including devices like computers, DVD players, and CD players is consistent with the '101 patent's description of an HDD portable player (which it describes as an example server device for media) allowing a user to "listen[] to the held music pieces of the HDD portable player 10 through the headphones 12." EX1001 ('101 patent), 4:32-34.

311. For example, Lamkin describes that "[s]ince playback is currently for media and/or content types that are supported by the client devices, **some embodiments have the server and/or computer 122 playback the content**" as a mechanism for handling DRM issues at the server device rather than at a client device. EX1004 (Lamkin), [0281].

312. **In sum**, regarding claim limitation [1f], Lamkin discloses or renders obvious that "the server device for media [Lamkin's computer/server 122] is a media player [Lamkin's DVD/CD player]."

313. For similar reasons, Lamkin also discloses claim limitations [6f], [7f], and [12f]. *See* §I above (Table of Claims).

B. Claim 6

314. As shown in §I above (Table of Claims), the claim 6 elements [6pre]-[6b-ii], and [6c-i]-[6f] are the same as elements [1pre]-[1b-ii] and [1c-i]-[1f]. Accordingly, to the extent the preamble is limiting, Lamkin discloses the limitations of [6pre]-[6b-ii] and [6c-i]-[6f] for the same reasons explained above with regard to the corresponding elements of claim 1, and I do not repeat the analysis here. *See* §XI.A above.

1. Claim [6b-iii]: wherein the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device is transferred after obtaining permission from a user

315. Lamkin renders obvious claim limitation [6b-iii].

316. Lamkin describes that protected content that cannot be transferred and accessed by a user without demonstration of further rights ("the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents") are only transferred from the server 122 to the other memory 132 or remote storage 146-147 after authorization or other demonstration by the user that

the appropriate rights are possessed for the transfer ("from the internal storage device to the network storage device is transferred after obtaining permission from a user"). EX1004 (Lamkin), [0253]-[0254], [0292]-[0293], [0316].

317. As I explain in §VIII.B.2 above, there are many reasons for network issues that may lead to loss of data, and it was conventional for storage and transfer systems to include mechanisms—and in some cases, multiple mechanisms—to protect against data loss, just as Lamkin does.

318. Lamkin describes that when the system experiences connectivity or network issues, "[s]ome embodiments further implement warnings [to] be presented to define what the system is waiting for after a predefined time (e.g., 5 seconds) allowing the user to make an informed decision to continue to wait or to cancel the current operation." EX1004 (Lamkin), [0253], *see also* [0292]. In particular, "[i]n handling loss of connectivity, some embodiments further provide a visual indication when a client device has been lost, while being robust enough to avoid hanging the user interface when connectivity has been lost [and in some cases a] network device status/control panel can be presented." *Id.*, [0254], *see also* [0293]. In these situations, Lamkin describes that the user is presented with relevant information related to a network issue interfering with transfer of content or another operation, and is further presented with options to allow the user to decide how or whether to proceed with the operation.

319. A POSITA would have understood that it was well-known and conventional to present a user with an option to provide instructions to the system if a failure had occurred or was likely to occur. *See e.g.,* EX1007 (Marotta), [0481]-[0482], [0691]-[0710], FIGS. 20, 41. In some situations, Lamkin also describes that the user will be prompted to provide additional information in addition to the selection of whether to proceed described above. *See* EX1004 (Lamkin), [0316] ("the agent interface… manages the backup/archiving, and initiates requests or informs the user **when additional information is needed**").

320. As I described in §VIII.E.1 above, content that is copyright protected or subject to other digital rights protections is often prevented from being transferred unless there is some identifier that shows that the transfer is legitimate and within the rights or licenses of the user making the transfer. For example, content subject to copyright may require the presence of a certificate or encryption key in order to transfer the content.

321. A POSITA would have understood, based on Lamkin's disclosure, that requiring authorization prior to transferring content that could not be successfully transferred or accessed upon transfer without authorization provides the claimed "digital contents … transferred after obtaining permission from a user" because, were the content to be transferred without authorization, it would not be readable by a client device (in the event of network failure or otherwise) and the protected

content is thus an example of "digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device." For example, Lamkin describes a situation in which "a user obtains the necessary rights to that content (for personal use and some specific number of copies/backups)" which a POSITA would have understood indicates that in Lamkin's system, protected content is only transferred when a user has provided authentication or otherwise demonstrated that the appropriate licenses and rights have been purchased to do so, as was conventional by the Critical Date. EX1004 (Lamkin), [0318]; *see also* §VIII.E.1 above.

322. **In sum**, regarding claim limitation [6b-iii], Lamkin discloses "the digital contents [Lamkin's contents including audio/video content] that cannot be recovered [content that is protected, per Lamkin] if a network failure [Lamkin's loss of connectivity] occurs during the transferring [during distribution, per Lamkin] of the digital contents [Lamkin's contents including audio/video content] from the internal storage device [RAM of Lamkin's computer/server 122] to the network storage device [Lamkin's local memory 132 or remote storage 146-147] is transferred [backed up or archived, per Lamkin] after obtaining permission from a user [authentication from the user is received, per Lamkin]."

323. For similar reasons, Lamkin renders obvious claim limitation [12b-iii]. *See* §I above (Table of Claims).

C. Dependent Claims 2-5 and 8-11

1. Claim 2: The server device for media according to claim 1, wherein said digital contents data transmission processing unit causes the network storage device to transmit the corresponding data to the server device for media, and then transmits the corresponding data received from the network storage device from the server device for media to the network player

324. Lamkin discloses claim [2].

325. As an initial matter, this claim describes a standard caching technique, where data is retrieved from a backend device (network storage device), through an intermediary, proxy, or cache device (server device for media), and then to its final decisional for consumption by users/applications (network player). *See* §VIII.F above.

326. Lamkin describes that when the content requested by the user is stored in remote storage, the content is first transferred to the server ("causes the network storage device to transmit the corresponding data to the server device for media") before the server distributes the content to the client device ("then transmits the corresponding data received from the network storage device from the server device for media to the network player"). EX1004 (Lamkin), [0049]-[0052], [0078]-[0079], [0241]. In the below annotated Lamkin FIG. 1, local memory 132 and remote storage 146-147 are colored yellow ("network storage devices"), computer 122 is colored purple ("server device for media"), and players 124-130, 152, and 160-161 are colored blue ("network players"). As also shown in annotated FIG. 1

in the dotted red box, in some cases, Lamkin's local memory 132 (yellow, "network storage device") provides content stored in the local memory to the computer 122 (purple, "server device for media") before the computer 122 transmits the content to the player 130 (blue; "network player").

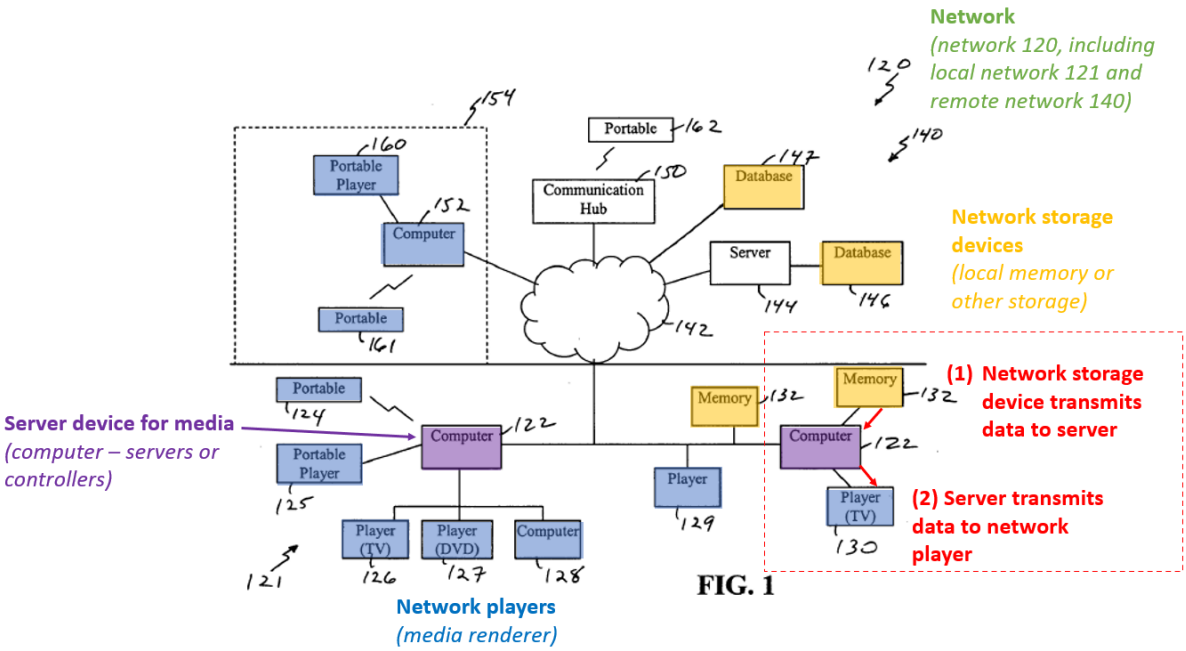


FIG. 1

EX1004 (Lamkin), FIG. 1 (annotated to show indirect data path)

327. Lamkin describes "[t]he **one or more servers or controllers 122** distribute, receive, control, update, **provide and/or coordinate access to content** [such that the] **client devices receive and/or access some content through the servers 122**." *Id.*, [0050]; *see also id.*, [0241] ("A media server stores, **retrieves and provides** media on request"). Lamkin's description of the server "coordinat[ing] access to content" so that "client devices receive… some content through the servers 122" indicates that the server acts as a proxy server in at least

some circumstances, requesting and receiving desired content from the local/remote storage before transmitting the content to the user. *Id.*, [0050]; *see* §VIII.D.1 above (explaining the function of a proxy server).

328. **In sum**, regarding claim [2], Lamkin discloses that "said digital contents data transmission processing unit [Lamkin's AV transport service of the computer/server 122] causes the network storage device [Lamkin's memory 132 or remote storage 146-147] to transmit [provide, per Lamkin] the corresponding data [Lamkin's content such as media files] to the server device for media [Lamkin's computer or server 122], and then transmits [transmits, per Lamkin] the corresponding data [Lamkin's content such as media files] received from the network storage device [Lamkin's memory 132 or remote storage 146-147] from the server device for media [Lamkin's computer or server 122] to the network player [Lamkin's players 124-130, 152, 160-161]."

329. For similar reasons, Lamkin discloses claim [8]. *See* §I above (Table of Claims).

2. Claim 3: The server device for media according to claim 1, wherein said digital contents data transmission processing unit transmits the corresponding data and information for identifying the network storage device to the network player, and causes the network storage device to directly transmit the corresponding data to the network player

330. Lamkin discloses and/or suggests claim [3].

331. As an initial matter, this claim describes an obvious idea—that to be able to retrieve data from some remote location, you have to identify that location. All network storage systems, whether client-server ones or more complex distributed storage systems, have to identify the location of each data item before transmitting it anywhere. Moreover, a POSITA would have understood that directly transmitting a piece of data, without going through an intermediary (*e.g.*, a proxy or cache device), is more efficient network-wise. *See* §VIII.D above.

332. Lamkin describes that during display of the content list by the CDS, the list of available contents and location of the contents are transmitted to the media renderer for display on the user interface ("transmits the corresponding data and information for identifying the network storage device to the network player"), and further that the server then causes the local/remote storage to transmit the selected data to the media renderer ("causes the network storage device to directly transmit the corresponding data to the network player"). EX1004 (Lamkin), [0286]; [0205] ("The content user interface can list or identify content available through the network, and in some embodiments, **further identify the device on which content is stored**"), [0113] ("The listing can be a centralized listing, one or more CDSs, one or more device listings, listings accessed over the remote network 142, and/or other such listings"), [0204] ("The content user interface displays at least some of the

available content accessible over the local and/or extended networks, and in some implementations provides an aggregated view of the content over the network 121").

333. Lamkin describes that digital content can be transferred to a user device from either the server or from the local/remote storage, and further that in some cases the content is transferred from local/remote storage through the server to the user device. EX1004 (Lamkin), [0286], [0241]. This description is consistent with the '101 patent's description of permitting direct (network storage device to network player) or indirect (network storage device to server device for media to network player) transmission of content. EX1001 ('101 patent), 7:50-65. In the below annotated Lamkin FIG. 1, local memory 132 and remote storage 146-147 are colored yellow ("network storage devices"), computer 122 is colored purple ("server device for media"), and players 124-130, 152, and 160-161 are colored blue ("network players"). As further shown in annotated FIG. 1 in the dotted red box, in some cases, the local memory 132 (yellow, "network storage device") provides content directly to the player 130 (blue, "network player").

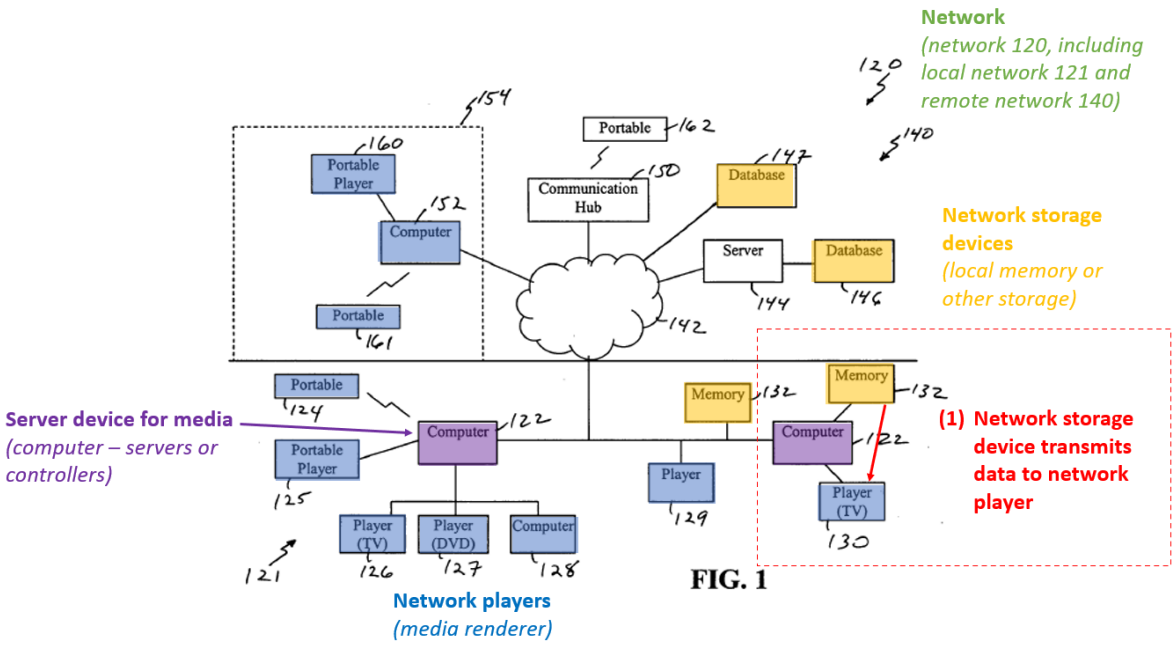


FIG. 1

EX1004 (Lamkin), FIG. 1 (annotated to show direct data path)

334. Although Lamkin's CDS provides the list of contents and their locations to the user, a POSITA would have understood that it would have been simple to program Lamkin's "AV transport service" instantiated as an AV transport algorithm to present the data along with location information to the media renderer (especially given that networked storage systems already have to know the location of each item, as I discuss in §VIII.D above). This alteration would have been well within the level of skill of a POSITA—merely returning an additional small piece of location information the server already has, along with the data itself—and would not have interfered with the function of Lamkin's AV transport service, CDS or server generally.

335. Alternatively, Lamkin's CDS and AV transport service are both parts of the server which function to provide content and location to a user and to prompt the transmission of content from the local/remote storage to the user. The '101 patent is not specific about particular algorithms or software that perform the claimed function and merely describes that "[t]he program according to the present invention causes a computer to function as each means of the above-described server device for media." To the extent that the '101 patent does not require a single algorithm or software component to perform all tasks of the claimed "digital contents data transmission processing unit" Lamkin's CDS and AV transport service meet this limitation as taught by the '101 patent. EX1001 ('101 patent), 3:5-9. Lamkin's CDS provides the list of contents and their locations to the user, while Lamkin's AV transport provides the contents themselves to the user device. EX1004 (Lamkin), [0113], [0204]-[0205], [0241], [0286].

336. **In sum**, regarding claim [3], Lamkin discloses and/or suggests that "said digital contents data transmission processing unit [Lamkin's AV transport service (or alternatively Lamkin's AV transport service and CDS) of Lamkin's computer/server 122] transmits [provides, per Lamkin] the corresponding data [Lamkin's content such as media files] and information for identifying the network storage device [Lamkin's universal locator indicating content is stored in Lamkin's memory 132 or remote storage 146-147] to the network player [Lamkin's players

124-130, 152, 160-161], and causes the network storage device [Lamkin's memory 132 or remote storage 146-147] to directly transmit [deliver, per Lamkin] the corresponding data [Lamkin's content such as media files] to the network player [Lamkin's players 124-130, 152, 160-161]."

337. For similar reasons, Lamkin discloses claim [9]. *See* §I above (Table of Claims).

3. Claim 4: The server device for media according to claim 1, further comprising a return control unit adapted to cause the digital contents corresponding to a predetermined condition among the digital contents which have been transferred to the network storage device to be returned from the network storage device to the internal storage device

338. Lamkin discloses claim [4].

339. As an initial matter, a POSITA would have understood that computers have internal storage devices, as I describe in §VIII.A above. Moreover, this claim describes nothing more than transmitting cached data back to its source or migrating data to make some room. *See* §VIII.D.1 above.

340. Lamkin describes transferring content between storage devices using the AV transport based on predicted use or in response to a user request for the content would thus result in content predicted to be used or selected for use ("the digital contents corresponding to a predetermined condition") being moved from local memory or remote storage to the server ("among the digital contents which have been transferred to the network storage device to be returned from the network

storage device to the internal storage device") based on the predetermined condition of being predicted to be accessed by the user or actually being selected for access by a user ("corresponding to a predetermined condition").

341. In the below annotated Lamkin FIG. 1, local memories 132 and remote storage 146-147 ("network storage devices") are colored yellow, client devices 124-130, 152, and 160-161 ("network players") are colored blue, server/computer 122 ("server device for media") is colored purple, and an internal storage device (colored red) is added to the server/computer 122 based on Lamkin's disclosure of storing content in the server/computer (*see* §XI.A.2 above). As also annotated in FIG. 1, the local memory 132 ("network storage device") transfers content to the internal storage device of the server/computer 122 ("server device for media"). EX1004, FIG. 1 (annotated below).

342. A POSITA would have understood that returning the content from the network storage device to the server device can occur as part of providing requested content to a user, or as part of Lamkin's scheme of organizing content so that content predicted to be used by a user is stored locally on the server device. EX1004 (Lamkin), [0046]. [0049]-[0052], [0064], [0078]-[0079], [0083], [0133], [0128], [0141]-[0146], [0220], [0231]-[0236], [0241].

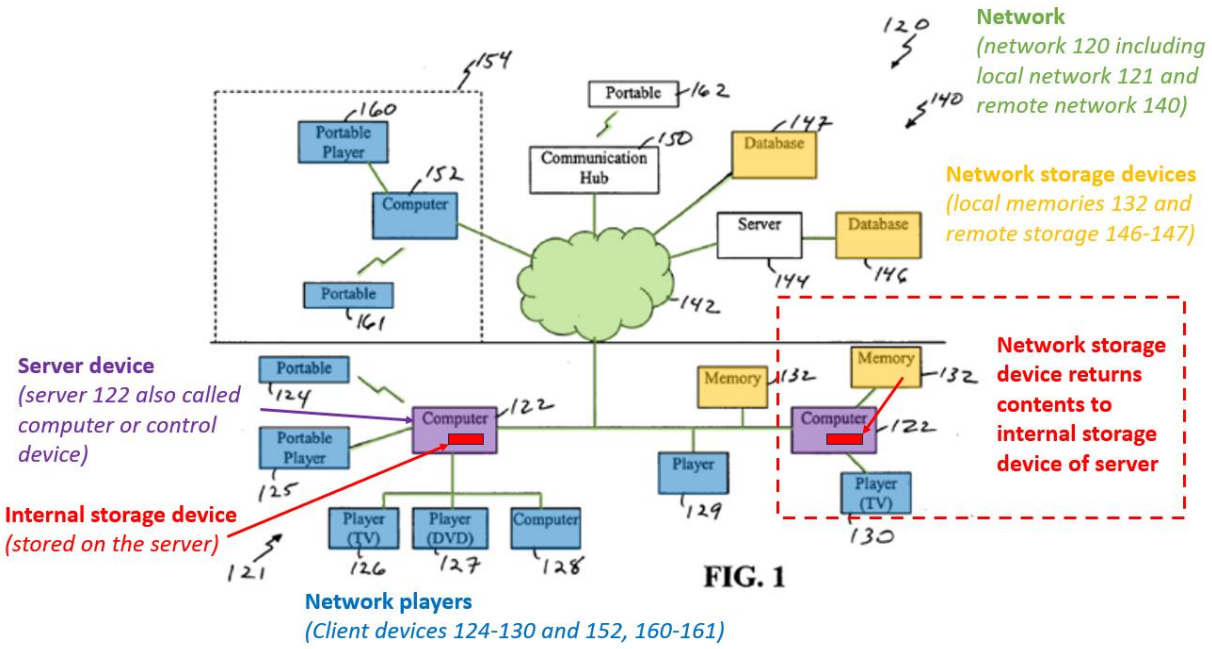


FIG. 1

EX1004, FIG. 1 (annotated)

343. As discussed above in §XI.A.11 above, Lamkin's "AV transport" performs the function of transmitting content between storage locations and network players. Therefore, the AV transport performs the same function as the claimed "return control unit"; it is instantiated as an AV transport algorithm, which a POSITA would have understood is computer-implemented software for performing a transport task. EX1004 (Lamkin), [0236] ("a control algorithm 2820 for an AV architecture"), [0049]-[0052]. A POSITA would have understood that an algorithm is a way to "cause[] a computer to function" in a particular way to achieve a task. EX1001 ('101 patent), 3:5-9. Indeed, an "algorithm" is "[a] finite sequence of steps for solving a logical or mathematical problem or performing a task." EX1017 (Microsoft Computer Dictionary), 5.

344. Accordingly, a POSITA would have appreciated that Lamkin's "AV transport" implemented as an algorithm is software implemented in a computer, just as the '101 patent's "return control unit" must be. Lamkin's AV transport uses the control algorithm to perform functions including transferring content between devices. EX1004 (Lamkin), [0236].

345. As I described in §XI.C.1 above, a POSITA would have understood Lamkin to disclose a method of transferring the requested content from local memory or remote storage to the server before the content is transferred to a client device; a POSITA would recognized this to be conventional in a tiered storage architecture. *See also* §VIII.D.1 above; EX1004 (Lamkin), [0049]-[0052], [0078]-[0079], [0083], [0128], [0141], [0220], [0231]-[0236], [0241].

346. Moreover, the '101 patent describes that content can be returned to the server device for media for a number of predetermined reasons, including "the digital contents which have recently been subjected to the self-playback among the digital contents which have been transferred to the network storage device 57, or the digital contents which have recently been played whether by self-playback or the network-playback." EX1001 ('101 patent), 8:5-10.

347. A POSITA would have understood that when Lamkin's system returns the requested content from local/remote storage to the server, it is returning content based on a predetermined reason; that is, based on the predetermined reason of being

selected for playback or other presentation to the user. This represents a common experience for a POSITA: once a media (*e.g.*, video) has been played once, often it need not consume resources in its current location and can be transferred or moved elsewhere. This is also consistent with the description of predetermined conditions in the '101 patent. *See e.g.*, *id*., 8:5-10 (describing that predetermined conditions can include "the digital contents which have recently been subjected to the self-playback" or "network playback").

348. Lamkin also describes that the content management service transfers content between storage devices based on predicted use, also performing the same functions as the claimed "return control unit." EX1004 (Lamkin), [0064], [0046]. For example, Lamkin describes that in some situations, content that is predicted to be used by a user is moved within the network so as to be readily available to anticipate user needs. EX1004 (Lamkin), [0079] ("Some implementations utilize the **local server storage to maintain the most used and/or most current content on the network**. This allows for rapid retrieval and ease of distribution to one or more client devices 124-130"), [0064] ("**Content can be distributed through the network 120** by some embodiments for many reasons, such as centralization of content, organization of content, **user or client device demands, anticipated or predicted use of the content and/or organization of content**, and/or for other content management"), [0133], [0141]-[0144], [0146] ("distribution can be based in

part on usage patterns and/or user profiles"), [0238]. Lamkin describes allocating content between the various available storage locations including the server and local/remote storage based on user needs, including, moving "the most used and/or most current content on the network" to the "local server storage" for ease and speed of retrieval by a user. *Id.*, [0079].

349. A POSITA would have understood that what Lamkin describes is consistent with a caching scheme with which a POSITA would have been familiar. *See* §VIII.F above; EX1004 (Lamkin), [0080] (describing hierarchy algorithm determines where content is stored), [0103] ("Upon detection of an event, the network 121 and/or server 122 initiates the distribution of content based on a predicted distribution scheme"), [0146]. Such schemes, as I describe in §VIII.F above, where conventionally implemented to automatically determine a storage location for content based on particular characteristics, properties or factors of the content.

350. **In sum**, regarding claim [4], Lamkin discloses that "a return control unit [Lamkin's AV transport service] adapted to cause the digital contents [Lamkin's audio/video content] corresponding to a predetermined condition [Lamkin's content selected for playback, or alternatively, content predicted to be used by a user] among the digital contents [from Lamkin's contents] which have been transferred [stored in. per Lamkin] to the network storage device [Lamkin's local memory 132 or

remote database storage 146-147] to be returned from the network storage device [Lamkin's local memory 132 or remote database storage 146-147] to the internal storage device [to the RAM of Lamkin's computer or server 122]."

351. For similar reasons, Lamkin discloses claim [10]. *See* §I above (Table of Claims).

4. Claim 5: The server device for media according to claim 1, wherein said list information transmission unit makes the list information to be transmitted to the network player include information for identifying whether each digital content is currently stored in the internal storage device or the network storage device in the display list of the network player

352. Lamkin discloses claim [5].

353. As an initial matter, this claim describes a file namespace unification scheme where data or files that reside in different locations are presented in a unified view. Including the location with a data item was well known, as I described in §VIII.D above. Moreover, file namespace unification was well known and I personally worked on such techniques, as described in my qualification in §III above.

354. Lamkin describes that the content user interface (populated by the CDS) displays to the user "the device on which content is stored." EX1004 (Lamkin), [0205]. Lamkin describes, for example, that "**some embodiments further provide for network navigation to allow users to more easily and readily access desired content [where] the local network server or controller 122**

identifies the devices 122, 124-130 and 132 of the local network and/or devices 152, 160-161 of the extended network when appropriate." *Id.*, [0199]. The network navigation may further include "a network mapping, for example, that can be **displayed on a user interface to allow users to more accurate[ly] perceive the local and/or extended network and the devices on the network**." *Id.*

355. Lamkin describes that the content user interface displays the location of the content in the list, as well as network navigation that allows a user to understand where listed content is located. EX1004 (Lamkin), [0205]. From Lamkin's disclosures, a POSITA would have understood that Lamkin's CDS ("list information unit") compiles the content list ("list information") and transmits the list to the media render ("to be transmitted to the network player") with the location of the listed content ("include information for identifying whether each digital content is currently stored") in the server 122 ("in the internal storage device") or in the other memory 132 or remote storage 146-147 ("or the network storage device") for display in the user interface ("in the display list of the network player"). *See e.g.*, EX1025 (Van Den Heuvel), [0045] (explaining that a "CDS additionally provides lookup/storage service allowing clients (e.g. UI devices) to locate and perhaps store individual content items that the media server is capable of providing.").

356. **In sum**, regarding claim [5], Lamkin discloses that "said list information transmission unit [Lamkin's content directory service] makes the list

information [Lamkin's listing of content in the aggregated database] to be transmitted [transmitted for display, per Lamkin] to the network player [Lamkin's players 124-130, 152, 160-161] include information for identifying [includes Lamkin's universal locator to provide location information] whether each digital content [Lamkin's content] is currently stored [is available, per Lamkin] in the internal storage device [RAM of Lamkin's computer 122] or the network storage device [Lamkin's memory 132 or remote storage 146-147] in the display list [Lamkin's user interface of player displaying directory of contents] of the network player [Lamkin's players 124-130, 152, 160-161]."

357. For similar reasons, Lamkin discloses claim [11]. *See* §I above (Table of Claims).

XII. GROUND 1B: LAMKIN IN VIEW OF FIECHTER RENDERS CLAIMS 1-12 OBVIOUS

358. Beyond the relevance of Lamkin's teachings alone, as described in Ground 1A (*see* §XI.A.6 above and §XI.B.1 above), the features recited in claim limitations [1b-iii] and [6b-iii], and corresponding features of claim limitations [7b-iii] and [12b-iii] also would have been obvious to a POSITA based on the combined teachings of Lamkin and Fiechter for at least the reasons set forth in the Petition, which I have read and fully endorse. *See* §I above (Table of Claims). To streamline the analysis in this declaration, I do not repeat all of the reasons stated in the Petition as to why a POSITA would have found claim elements [1b-iii] and [6b-

iii], and corresponding features of claim limitations [7b-iii] and [12b-iii] obvious based on the combination of Lamkin and Fiechter, but by way of example and to highlight certain points, I make the following additional observations.

359. The combination of Lamkin and Fiechter renders obvious all other elements of claims 1-12 for the same reasons described with respect to those elements in Ground 1A (*see* §XI above).

360. As I described in §XI.A.6 above, Lamkin describes that there are various circumstances where properties of particular content may mean that the content has higher risk of content loss during transferring of the content from the server to other storage (*i.e.*, the content would be "unrecoverable" in the case that a network failure occurs during transfer. In particular, Lamkin describes that content that has a very large file size and would require a high number of packets to be sent to transmit the entire file and content that is subject to particular kinds of copyright protection or other DRM considerations may fall into this category. EX1004 (Lamkin), 1004, [0074] ("In some instances, the network may limit the centralization of content when content is large"), [0127] ("The content management can provide users with access to content, synchronize content, archive content, establish collections of content, transfer files and/or licenses, and other such content management"), [0220] ("In some embodiments, the content management further defines, implements and/or interprets rules or authorization, for example, to

transcode, to transrate, access rights management, to go and acquire the rights to make a copy or to move content from a first device to a second device, to transfer a license and other such management.").

361. Lamkin describes two techniques for dealing with these types of content, the first being not transferring content that is too large or is subject to the kinds of copyright protection that limit its ability to be transferred. *See* §XI.A.6 above; *see also* EX1004 (Lamkin), [0074] ("When step 430 is performed and it is determined that the content exceeds the threshold, the process continues to step 450"), [0075] ("when the content is protected content, it is determined in step 434 whether the content can be transferred unprotected. The protection of some content may be limited to delivery to the network 120 and once received, available to be use across the network, while some content may have higher protection that limits the ability to transfer the content unprotected"), [0076] ("When the license cannot be transferred, the content is not centralized and the process 420 continues to step 450."). The second technique is to only transmit content—in particular, protected content that has limitations on transfer of the content—when the user has provided additional instructions or authorization to proceed. *See* §XI.B.1 above; *see also* EX1004 (Lamkin), [0067] ("some content may be content protected and thus **unable to be distributed or properly distributed without further authorization**"), [0316] ("agent interface… informs the user when additional information is needed").

362. Lamkin briefly describes the techniques for dealing with the transfer of content that is more likely to be "unrecoverable" if a network failure occurs. Fiechter, however, provides additional details about the implementation of such techniques and the network characteristics that may signal an increased risk of loss for a broader class of content; thus, Fiechter's techniques are more broadly applied in a system like Lamkin's. EX1006 (Fiechter), 17:50-59 ("mobile audio device 904 includes capability to detect and compensate for data transmission errors in wireless communication network 906," including detection that the device is in an area "where there is a high incidence of data loss due to propagation errors.").

363. Accordingly, in the combination, a system, such as that described by Lamkin, includes "includes capability to detect and compensate for data transmission errors [including] determining… a high incidence of data loss." EX1006 (Fiechter), 17:50-56. When high risk of data loss is detected, "one of several options may be taken including informing the user of the likelihood of errors or gaps in the information, asking the user if they would prefer to wait for the information until transmission of the data will be more reliable, or buffering a greater than normal amount of information to be able to continue undisrupted output until the connection is re-established." *Id.*, 17:64-18:3. Put more plainly, in the combination, a system such as Lamkin's can detect when a network failure that may compromise transfers over the network and result in data loss is more likely, and in

response to such detection either (1) not transfer content when there is risk of data loss (*e.g.*, waiting) or (2) asks the user to authorize proceeding with the transfer despite the risk (*e.g.*, asking the user to confirm to transfer). *Id.*, 17:64-18:3.

A. Claim [1b-iii]: wherein said transfer control unit does not transfer, from the internal storage device to the network storage device, the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device

364. The combination of Lamkin and Fiechter renders obvious claim limitation [1b-iii].

365. The combination of Lamkin and Fiechter provides a content management service ("transfer control unit"), per Lamkin, that anticipates when data loss associated with content to be transferred is more likely due to network failures or connectivity issues ("digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device"), per Fiechter, and prevents the transfer of the content from the server to the local/remote storage as a result ("does not transfer, from the internal storage device to the network storage device, the digital contents"). EX1004 (Lamkin), [0067], [0074]-[0076], [0082], [0105], [0182], [0245], [0253]-[0254], [0290]-[0293]; EX1006 (Fiechter), 17:62-18:3 ("wait … until transmission of the data will be more reliable"). A POSITA would have understood that a user selecting Fiechter's option of "wait[ing] … until transmission of the data will be more

reliable" would mean that the transfer of the data is not attempted during the period in which there is anticipated high likelihood of data loss.

366. **In sum**, regarding claim limitation [1b-iii], the combination of Lamkin and Fiechter renders obvious "said transfer control unit [Lamkin's content management services] does not transfer [Feichter does not transfer and waits for more reliable network], from the internal storage device [RAM of Lamkin's computer/server 122] to the network storage device [Lamkin's local memory 132 or remote storage 146-147], the digital contents [Lamkin's contents including audio/video content] that cannot be recovered [Fiechter's anticipated high incidence of data loss] if a network failure [Fiechter's communication network issues or failure] occurs during the transferring [Lamkin's during transfer] of the digital contents [Lamkin's contents including audio/video content] from the internal storage device [RAM of Lamkin's computer/server 122] to the network storage device [Lamkin's local memory 132 or remote storage 146-147]."

367. For similar reasons, the combination of Lamkin and Fiechter renders obvious the corresponding features of claim limitation [7b-iii]. *See* §I above (Table of Claims).

B. Claim [6b-iii]: wherein the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device is transferred after obtaining permission from a user

368. The combination of Lamkin and Fiechter renders obvious claim limitation [6b-iii].

369. In the combination of Lamkin and Fiechter, a system, per Lamkin, anticipates when data loss associated with content to be transferred is more likely due to network failures or connectivity issues ("the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device"), per Fiechter, and transfers contents after a user is prompted to provide instructions to proceed with the transfer ("is transferred after obtaining permission from a user"). EX1004 (Lamkin), [0067] ("some content may be content protected and thus unable to be distributed or properly distributed without **further authorization**"), [0253]-[0254] [0292]-[0293]; EX1006 (Fiechter), 17:62-18:3 ("informing the user of the likelihood of errors or gaps in the information [or] asking the user if they would prefer to wait for the information until transmission of the data will be more reliable"). A POSITA would have understood that "further authorization" (per Lamkin) or "asking the user" to make a selection (per Fiechter) involves asking a user for permission, an additional step.

370. **In sum**, regarding claim limitation [6b-iii], the combination of Lamkin and Fiechter renders obvious "the digital contents [Lamkin's contents including audio/video content] that cannot be recovered [Fiechter's anticipated high incidence of data loss] if a network failure [Fiechter's communication network issues or failure] occurs during the transferring [during transfer in Lamkin] of the digital contents [Lamkin's contents including audio/video content] from the internal storage device [RAM of Lamkin's computer/server 122] to the network storage device [Lamkin's local memory 132 or remote storage 146-147] is transferred [Lamkin's is moved] after obtaining permission from a user [Fiechter after asking the user if they would like to proceed or wait]."

371. For similar reasons, the combination of Lamkin and Fiechter renders obvious the corresponding features of claim limitation [12b-iii]. *See* §I above (Table of Claims).

C. Motivations for the Combination of Lamkin and Fiechter

372. A POSITA would have been motivated to combine Lamkin with Fiechter for at least the following six reasons.

373. **First**, a POSITA would have found it obvious to apply Fiechter's teachings to Lamkin because a POSITA would have understood that network failures and connectivity disruptions are common occurrences in most, if not all,

networks; thus, a system, such as that described by Lamkin, incorporates techniques for dealing with inevitable interruptions of transfer. *See* §VIII.B.2 above.

374. A POSITA, understanding the common issues raised by network failures, would have been motivated to provide mechanisms for predicting the occurrence of network failure-related issues and dealing with them, as Fiechter teaches. A POSITA would have been familiar with such mechanisms, including presenting a user with options for how to proceed. *See e.g.*, EX1007 (Marotta), [0481]-[0482], [0691]-[0710], FIGS. 20, 41.

375. **Second**, a POSITA would have understood that the potential techniques for dealing with the possibility of data loss are common-sense. Faced with increased risk of loss, either through increased likelihood of a failure or through properties of content that make the content more likely to be lost in the case of a failure, a POSITA would have had few options: either (i) to not transfer the content (or not transfer the content until a later time) or (ii) to decide to proceed with transferring the content despite the risk.

376. Understanding that the second option of proceeding with the transfer risks a potential loss of content, it makes sense to ask a user to authorize the action. A POSITA would have been motivated to implement additional user options to select between waiting or proceeding with the transfer, per Fiechter, in view of

Lamkin's description of user authorization of a transfer. EX1006 (Fiechter), 17:50-18:3.

377. **Third**, determining that data loss is likely or that a transfer is likely to fail, and taking precautions as a result, is a straightforward way to deal with uncertainties about conditions for a transfer, and may reduce the loss of data or the number of transfer attempts required to succeed in a transfer. *See* §VIII.B.1 above and §VIII.B.2 above (discussing networking transfers and transfer attempts). A POSITA would have sought to integrate Fiechter's teachings in a system like Lamkin's to achieve these advantages.

378. **Fourth**, a POSITA would have known that content loss is frustrating to a user, especially when the content is protected content that may have been purchased by the user. Thus, systems that predict and prevent data loss associated with network failures help greatly to reduce such frustrations. *See* §VIII.E.1 above. Additionally, network failures impact all types of computer systems, including stationary, home-based systems, and mobile systems, like many of the systems Fiechter describes. The techniques for mitigating the effect of potential network failures taught by Fiechter are equally applicable in home networks or mobile networks.

379. **Fifth**, a POSITA would have understood that in the combination, Lamkin's system continues to function as Lamkin describes (*see* §XI.A above),

implemented with details related to predicting and dealing with anticipated content loss related to network failure as Fiechter describes (*see* §X.3 above). A POSITA would have understood that the combination of a system such as that taught by Lamkin with teachings of Fiechter would result in the system with improved ability to anticipate and address potential loss of content during transfer, because the implementation of software to effect Fiechter's teachings would have had predictable results.

380. Moreover, as described in §XI.A.6 above, Lamkin's system already includes mechanisms for addressing network issues that impact content transfer, for example by not transmitting the content or by presenting information and options to a user to proceed with the transfer. *See* §XI.A.6 above. Fiechter's additional details about how to implement the halt in transferring content and what options to provide to a user does not alter the performance of the Lamkin system. *See* §XI.B.1 above.

381. **Sixth**, the combination of Lamkin and Fiechter would have been well within a POSITA's skill because mechanisms for monitoring network capabilities and failures as well as methods for responding to such failures by preventing a transfer or providing options to a user were well-known by the Critical Date, as Fiechter teaches. EX1006 (Fiechter), 17:62-18:3; *see also* §VIII.B.2 above. Additionally, implementing software to effect Fiechter's teachings to anticipate network issues that may impact transfer and result in content loss and to address

anticipated issues by preventing transfer or asking the user to provide an instruction to proceed would have been well within the level of skill of the POSITA. *See* §VII.A above.

382. Finally, a POSITA would have had a reasonable expectation of success in doing so for the reasons explained in this section, and because methods for responding to increased likelihood of network failures during data transfer were well known to a POSITA by the Critical Date. *See* §VIII.B above.

XIII. GROUND 1C: LAMKIN IN VIEW OF ITO RENDERS CLAIMS 1-5 AND 7-11 OBVIOUS

383. Beyond the relevance of Lamkin's teachings alone, as described in Ground 1A (*see* §XI.A.6 above), the features recited in claim limitations [1b-iii] and [7b-iii] also would have been obvious to a POSITA based on the combined teachings of Lamkin and Ito, for at least the reasons set forth in the Petition, which I have read and fully endorse. *See* §I above (Table of Claims). To streamline the analysis in this declaration, I do not repeat all of the reasons stated in the Petition as to why a POSITA would have found claim element [1b-iii], and corresponding features of claim limitation [7b-iii] obvious based on the combination of Lamkin and Ito, but by way of example and to highlight certain points, I make the following additional observations.

384. The combination of Lamkin and Ito renders obvious all other elements of claims 1-5 and 7-11 for the reasons described with respect to those elements in Ground 1A (*see* §XI above).

385. As I discuss in §XI.A.6 above, Lamkin describes that there are various circumstances where properties of particular content may mean that the content has higher risk of content loss during transferring of the content from the server to other storage (*i.e.*, the content would be "unrecoverable" in the case that a network failure occurs during transfer. In particular, Lamkin describes that content that has a very large file size and would require a high number of packets to be sent to transmit the entire file and content that is subject to particular kinds of copyright protection or other DRM considerations may fall into this category. EX1004 (Lamkin), [0074] ("In some instances, the network may limit the centralization of content when content is large"), [0220] ("In some embodiments, the content management further defines, implements and/or interprets rules or authorization, for example, to transcode, to transrate, access rights management, to go and acquire the rights to make a copy or to move content from a first device to a second device, to transfer a license and other such management."). Lamkin describes not transferring content that is too large or is subject to the kinds of copyright protection that limit its ability to be transferred. *See* §XI.A.6 above.

386. Lamkin briefly describes a technique for dealing with the transfer of protected content that is more likely to be "unrecoverable" if a network failure occurs. Ito, however, provides additional details about the implementation of such a technique when there is an intention to back up digital content including protected content. EX1012 (Ito), [0002] (copy control information "indicates that copying is allowed only once (Copy Once)" and as a result, "indicates that backup is prohibited, [such that] the transmission is terminated"), *see also generally* [0001]-[0008], [0252].

387. Accordingly, in the combination, a system, such as that described by Lamkin, includes copy control information related to digital content "indicates that copying is allowed only once (Copy Once)" and as a result, if the information associated with the content "indicates that backup is prohibited, [then] the transmission is terminated," per Ito. *Id.*, [0002], [0252]. Put more plainly, in the combination, a system such as Lamkin's, when attempting to store protected data that includes limitations on copying/transferring, does not transmit the protected data to the local memory or remote storage if copy control information related to the digital content indicates that the content cannot be transferred, as Ito teaches. *Id.*; *see also* EX1004 (Lamkin), [0067] ("some content may be content protected and thus unable to be distributed"), [0055], [0075]-[0076], [0114]-[0117], [0006], [0123], [0173], [0237], FIG. 8.

A. Claim [1b-iii]: wherein said transfer control unit does not transfer, from the internal storage device to the network storage device, the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device

388. The combination of Lamkin and Ito renders obvious claim limitation [1b-iii].

389. In the combination of Lamkin and Ito, where content is protected and cannot be backed up to guard against network failure ("digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device") the content transfer to the backup is terminated ("said transfer control unit does not transfer, from the internal storage device to the network storage device"). EX1004 (Lamkin), [0067], [0074]-[0076], [0082], [0105], [0182], [0245], [0253]-[0254], [0290]-[0293]; EX1012 (Ito), [0002] (copy control information "indicates that copying is allowed only once (Copy Once)" and as a result, "indicates that backup is prohibited, [such that] the transmission is terminated"), *see also generally* [0001]-[0008], [0252]-[0253].

390. **In sum**, regarding claim limitation [1b-iii], the combination of Lamkin and Ito render obvious "said transfer control unit [Lamkin's content management services] does not transfer [Ito terminates the transfer], from the internal storage device [RAM of Lamkin's computer/server 122] to the network storage device

[Lamkin's local memory 132 or remote storage 146-147], the digital contents [Lamkin's contents including audio/video content] that cannot be recovered [subject to "Copy Once" protections in Ito] if a network failure [Lamkin's loss of connectivity] occurs during the transferring [Lamkin's during transfer] of the digital contents [Lamkin's contents including audio/video content] from the internal storage device [RAM of Lamkin's computer/server 122] to the network storage device [Lamkin's local memory 132 or remote storage 146-147] to the network storage device."

391. For similar reasons, the combination of Lamkin and Ito renders obvious corresponding features of claim limitation [7b-iii]. *See* §I above (Table of Claims).

B. Motivations for the Combination of Lamkin and Ito

392. A POSITA would have been motivated to combine Lamkin with Ito for at least the following five reasons.

393. **First**, a POSITA would have found it obvious to apply Ito's teachings to Lamkin because, both Lamkin and Ito describe a system for backing up digital content in a network, including copyright-restricted content. *Compare* EX1012 (Ito), [0001] ("generating a backup of digital content while taking copyright protection of digital content into consideration"), *with* EX1004 (Lamkin), [0050] ("content can be received from the client devices and stored in the servers 122 and/or

storage 132"), [0107] ("in determining whether the content can be distributed, the network may determine whether the content is protected"), [0237], FIG. 4.

394. A POSITA would have recognized that Lamkin and Ito both describe storage architectures that include considerations for how to deal with transferring protected content. *See* §VIII.E above. Lamkin and Ito are both also concerned about network reliability during transfer and Ito provides additional details for how the system should act when transfer of protected content is anticipated. A POSITA would have looked to the additional details per Ito about terminating a transfer, and would have been motived to implement a system, per Lamkin, with the teachings of Ito, to prevent loss of protected content in a transfer. EX1012 (Ito), [0001]-[0008], [0252]; EX1004 (Lamkin), [0047]-[0052].

395. **Second,** a POSITA would have recognized that Ito's approach to dealing with the possibility of data loss for protected content is common-sense. Indeed, a POSITA would have understood Ito's determination of whether to proceed with the backup of the protected content anticipates and attempts to prevent loss of the protected content due to transfer. For example, Ito's use of copy control information to inform transfer processes attempts to prevent loss of license or rights to the content that may occur if the system fails to comply with copy control information. Additionally, preventing transfer of protected content may minimize

risk of loss of data during a network failure while the transfer is underway. *See* §VIII.B.2 above and §VIII.E above.

396. This principle of data protection was known to others. *See e.g.*, EX1016 (Kawasaki), [0009] ("if contents are transferred from the storage unit of a PC to the content reproducing apparatus such as a portable device as outlined above, then the contents not allowed to be reproduced by the content reproducing apparatus under copyright will also be transferred to the apparatus for unauthorized reproduction"), [0047]-[0049], [0055] ("the reproduction of content data is controlled on the basis of a source ID assigned uniquely to each of the content data providers involved").

397. Indeed, a POSITA would have understood how to deal with the possibility of data loss of protected content with limited transfer rights, for example, when copy-control information indicates copy only once; in that case, not completing the transfer made common sense. If the protected content will be compromised or lost to the user as a result of the transfer, the logical response is not to transfer the content from where it is stored. *See* §VIII.E.1 above.

398. **Third**, a POSITA would have understood that content loss is frustrating to a user especially when the content is protected content that may have been purchased by the user. Thus, systems that predict and prevent data loss associated with network failures help greatly to reduce such frustrations. *See* §VIII.E.1 above.

399. **Fourth**, a POSITA would have understood that in the combination, Lamkin's system continues to function as Lamkin describes (*see* §XI.A above), implemented with details related to predicting and dealing with anticipated content loss related to transfer rights of protected content as Ito describes (*see* §X.4 above). A POSITA would have understood that the combined system of Lamkin and Ito would yield an improved ability to anticipate the potential loss of protected content during transfer; this is because the implementation of software to effect Ito's teachings—for example to check the copy control information associated with content prior to initiating transfer—would have had predictable results.

400. **Fifth**, the combination of Lamkin and Ito would have been well within a POSITA's skill because mechanisms for anticipating content loss during transfer of protected content was well-known by the Critical Date, as Ito teaches. EX1012 (Ito), [0001]-[0008], [0252]-[0253]; *see also* §VIII.B above. Additionally, implementing software to effect Ito's teachings to anticipate issues related to transfer of protected content to mitigate content loss by preventing transfer would have been well within the level of skill of the POSITA. *See* §VII above.

401. Finally, a POSITA would have had a reasonable expectation of success in making the combination for the reasons explained in this section, and because methods for anticipating and preventing data loss during transfer of protected content were well known to a POSITA by the Critical Date. *See* §VIII.B above.

XIV. GROUND 2A: FRANKE RENDERS CLAIMS 1-12 OBVIOUS

402. In my opinion, a person of ordinary skill in the art would have found claims 1-12 of the '101 patent to have been obvious based on Franke. In my opinion, Franke renders each of the limitations of claims 1-12 obvious for at least the reasons set forth in the Petition, which I have read and fully endorse. To streamline the analysis in this declaration, I do not repeat all of the reasons stated in the Petition as to why a POSITA would have found claims 1-12 obvious based on Franke. Nonetheless, by way of example and to highlight certain points, I make the following additional observations.

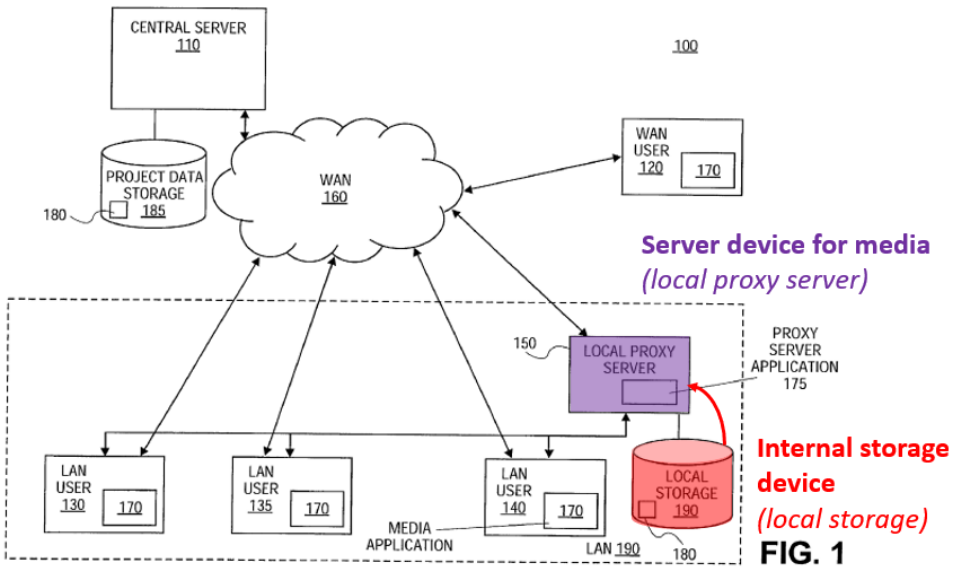
A. Claim 1

1. Claim [1pre]: A server device for media, the server device for media comprising

403. Franke discloses [1pre]. For example, Franke describes that "users can post or store [] data, e.g., ***media data***, on the local proxy ***server***" ("server device for media") to allow users to access the data locally. EX1005 (Franke), [Abstract].

404. Franke also describes that in addition to storing the data, the local proxy server "can also store the data from the local users on a central server." *Id.*, [Abstract]; *see also id.*, [0028], [0030] ("Local proxy server 150 is a client of central server 110 and acts as intermediary between central server 110 and LAN users 130-140"). Franke's annotated FIG. 1 below shows the local proxy server 150 (annotated in purple) coupled to local storage 190 (annotated in red). *See* EX1005 (Franke),

FIG. 1 (annotated below). Additionally, FIG. 1 shows LAN users 130-140 connected to local proxy server 150 in the LAN 190, as well as central server 110, project data storage 185, and WAN user 120 connected to WAN 160. *Id.*



EX1005 (Franke), FIG. 1 (annotated)

405. Franke's local proxy server is an "intermediary" between the user and the central server, because the users can deal directly with the local proxy server to save data to the central server and to access data stored on the central server (rather than dealing with the central server itself). *See id.*, [0030]; *see also* §VIII.B.1 above. For example, the local proxy server allows the local users to collaborate with remote users by saving contents to the local proxy server that will be shared to remote users by the local proxy server, and accessing contents from remote users through the local proxy server. *See id.*, [0004] ("A collaboration application allows multiple users to

work together on the same project from remote locations."), [0005] ("requiring local area users to download data from only a central server inefficiently uses network resources when the data may already exist locally") [0031] ("LAN users 130-140 and user 120 ("WAN user 120") can share data via WAN 160 and LAN 190"), [0042] ("LAN users 130-140 can store media data (e.g., media data 180) on local proxy server 150 … If LAN users 130-140 post media data or object data on local proxy server 150, local proxy server 150 can forward or store the project data from the users on central server 110."), [0047] ("Server application 175 can maintain storage of media data ("media data 180") or object data (not shown) in local storage 190 as one or more data files for access by LAN users 130-140."), *see also* [0041]-[0043].

406. **In sum**, regarding element [1pre], Franke discloses "a server device [Franke's local proxy server] for media [Franke's storing media data]."

407. For similar reasons, Franke discloses claim limitations [6pre], [7pre], and [12pre]. *See* §I above (Table of Claims).

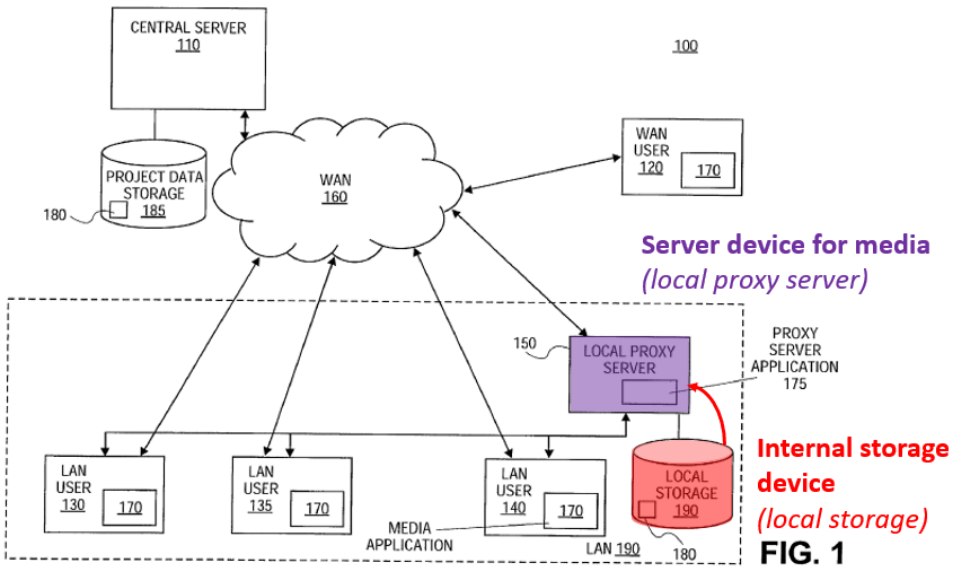
2. Claim [1a-i]: an internal storage device for storing digital contents

408. Franke discloses claim limitation [1a-i].

409. For example, Franke describes that the local proxy server ("server device for media") contains storage devices ("internal storage") for storing media data ("for storing digital contents"). EX1005 (Franke), [0045] ("**storage devices**

contained within server 150 may operate as a shared stor[age] device"), [0047] ("Server application 175 can maintain storage of media data ('media data 180') or object data (not shown) in local storage 190 as one or more data files for access by LAN users 130-140."), [0056]-[0059], FIG. 2.

410. Although Franke's FIG. 1 (annotated below) shows the local storage (colored red) connected to Franke's local proxy server (colored purple), Franke also describes that the local storage may be operated as a shared storage system "contained within" local proxy server. EX1005 (Franke), [0045], *see also generally* [0043]-[0045].



EX1005 (Franke), FIG. 1 (annotated)

411. For example, the local storage 190 can be "contained within local proxy server" and is thus an internal storage device of the local proxy server. *Id.; see also*

§VIII.A above (explaining that computers conventionally include internal storage or memory devices); EX1013 (Lambert), [0031] (describing that computer storage media includes "volatile and/or nonvolatile memory such as read only memory (ROM) 131 and random access memory (RAM)"), EX1014 (Dye), [0003] ("Since their introduction in 1981, the architecture of personal computer systems has remained substantially unchanged [and the] current state of the art in computer system architectures includes a central processing unit (CPU) which coupled to a memory controller interface that in turn couples to a system memory"), [0044]-[0046] (describing that computer system architectures include "programs and data … generally stored on the hard disk 120").

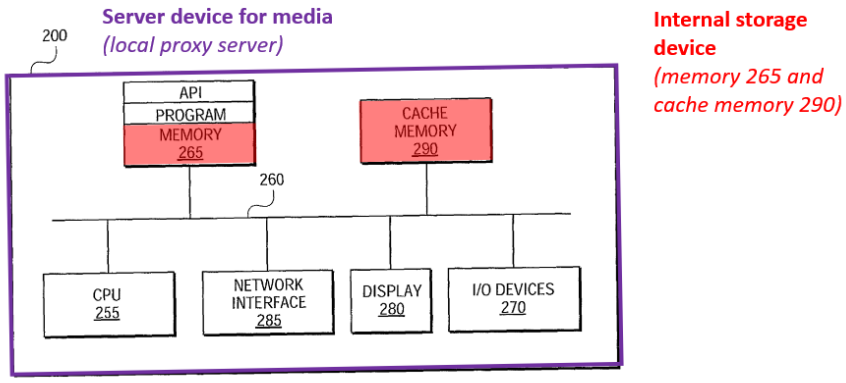


FIG. 2

EX1005 (Franke), FIG. 2 (annotated)

412. Franke's FIG. 2 (annotated above) further illustrates "an internal block diagram of an exemplary computer system 200" including "the internal components

of …. local proxy server 150" (surrounded by purple box above) that include memory 265 an cache memory 290 (colored red above). EX1005 (Franke), [0056] ("Computer system 200 may represent the internal components of LAN users 130-140, WAN user 120, central server 110, or local proxy server 150 as shown in FIG. 1"), [0057] ("Computer system 200 includes several components all interconnected via a system bus 260 [which is used] for accessing a memory 265 and a cache memory 290 and for transferring data among the components…. Examples of memory 265 and cache memory 290 include a random access memory (RAM), read-only memory (ROM), video memory, flash memory, or other appropriate memory devices."), *see also generally* [0056]-[0059].

413. **In sum**, regarding claim limitation [1a-i], Franke discloses "an internal storage device [Franke's local storage internal to server, for example RAM] for storing [Franke's storing for collaboration] digital contents [Franke's media/project data]."

414. For similar reasons, Franke discloses claim limitation [6a-i] and corresponding features of claim limitations [7a] and [12a]. *See* §I above (Table of Claims).

3. Claim [1a-ii]: wherein the server device for media responds to a data transmission request from a network player by stream-delivering corresponding data in corresponding digital contents from the internal storage device to the network player during connection to a network

415. Franke discloses claim limitation [1a-ii].

416. Franke describes that the local proxy server ("server device for media") can respond to user requests for content by transferring the content to the LAN users. EX1005 (Franke), [0028] ("Local area users can post or store data, e.g., media data, on the local proxy server … for the data to be accessed locally"), [0042] ("LAN users 130-140 can copy project data being stored on central server 110 **from local proxy server 150**"), [0047] ("Local proxy server 150 includes a server application 175, which can communicate with media application 170 to handle requests for media data or object data from LAN users 130-140"), [0064] ("LAN users 130 and 140 can thus download set 310 of posted media data from local proxy server 150"), *see also generally* [0042]-[0047].

417. Franke describes that in some conventional collaboration applications, users store content to a central server, and other users access that posted data on the central server by "request[ing] the posted data from the central server." *Id.*, [0004]. Franke describes that when there are multiple users in the same local network that collaborate on the content, these conventional systems can be inefficient. *Id*., [0005], [0035]. Instead, in Franke's system, a local proxy server handles "requests for media data or object data from LAN users" ("responds to a data transmission request from a network player"). *Id.*, [0047], [0053] ("LAN users 130-140 can also request project data from local proxy server 150").

418. Further, the request for media data may be a request for media data stored in the local proxy server on the local network ("corresponding data in corresponding digital contents from the internal storage device to the network player during connection to a network"). Franke describes that "because media data 180 or other data can be mirrored or stored on local proxy server 150, LAN users 130-140 are capable of accessing media data 180 within LAN 190." *Id.*, [0054].

419. A POSITA would have understood that multi-media streaming, such as Franke describes, was a well-known and conventional process by the Critical Date, and would have further understood that Franke discloses providing media data to the LAN users (and indeed throughout the LAN/WAN networks) by stream-delivering the content. *See* §VIII.E above; EX1005 (Franke), [0103] ("a streaming process can be implemented to download segments of the requested media data as soon as the segments become available on local proxy server 150."), [0008] ("a streaming process can be implemented"), [0109] ("a streaming process can be implemented").

420. **In sum**, regarding claim limitation [1a-ii], Franke discloses "the server device for media [Franke's local proxy server] responds [provides a download, per Franke] to a data transmission request [to a request for posted data, pre Franke] from a network player [from Franke's LAN users] by stream-delivering [by a streaming process, per Franke] corresponding data [Franke's media data or object data] in corresponding digital contents [Franke's project data] from the internal storage

device [Franke's local storage within local proxy server] to the network player [Franke's LAN users] during connection to a network [over Franke's LAN]."

421. For similar reasons, Franke discloses claim limitation [6a-ii] and corresponding features of claim limitations [7a] and [12a]. *See* §I above (Table of Claims).

4. Claim [1b-i]: a transfer control unit adapted to transfer and store part of held digital contents in the internal storage device to a network storage device

422. Franke discloses claim limitation [1b-i].

423. For example, Franke discloses a "[s]erver application 175" [which] can be software operating within local proxy server 150" ("a transfer control unit") that is adapted to perform the function of storing content from the local proxy server to the central server ("adapted to transfer and store part of held digital contents in the internal storage device to a network storage device"). EX1005 (Franke), [0047] ("Local proxy server 150 includes a server application 175, which can … store media data or object on central server 110 for the LAN users 130-140"), [0073], [0080], [0117], [Abstract].

424. As an initial matter, as described above, a POSITA would have understood that when computers transfer data (for example in transferring or downloading data), they cannot transfer it all at once, but instead do so in a piecemeal

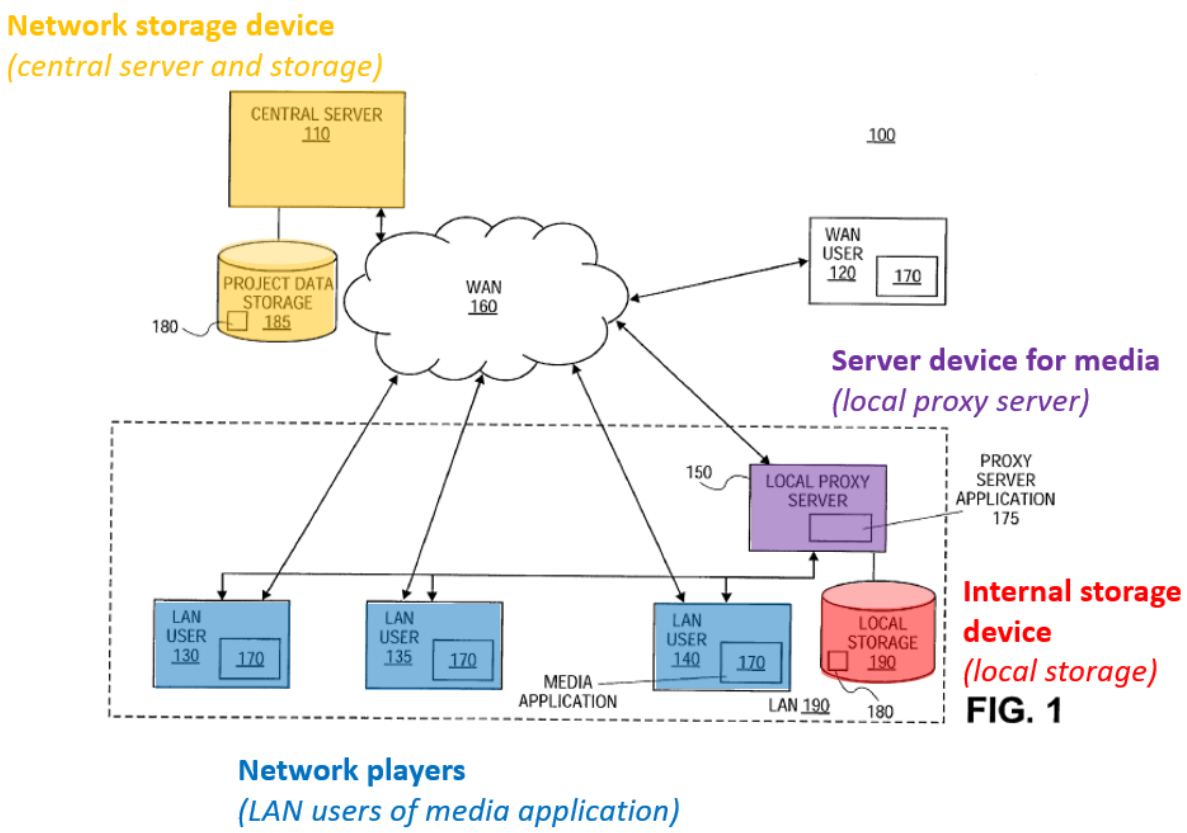
fashion. *See* §VIII.B above. Therefore, until all data has been transferred, only a part of it is transferred. *See id*.

425. Moreover, claim limitation [1b-iii], discussed in §XIV.A.6 below, specifically claims not transferring some data in particular situations; a POSITA would have further understood that if some data is not transferred, it means that other data most likely *was* transferred—that is, that "part" of the data has been transferred. *See* §XIV.A.6 below (claim limitation [1b-iii] reciting "and wherein said transfer control unit does not transfer, from the internal storage device to the network storage device, the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device"). Additionally, a POSITA would also understand that storing "part of" includes storing some portion of or all of the whole data.

426. Notably, the '101 patent does not identify particular algorithms or software for performing the functions of the "transfer control unit" and other claimed "units" and instead labels each 'unit' based on its assigned function and describes merely that a computer performs the functions based on a program or instructions. *See* EX1001 ('101 patent), 3:5-9. Franke's server application 175 performs each of the claimed functions of the 'units,' just as the '101 patent's 'computer' performs the functions. A POSITA would have understood that an application (such as Franke's server application 175) may include software and/or algorithms that

instruct a computer (in this case the local proxy server 150) to perform a variety of tasks. For example, it would have been obvious to a POSITA that software including instructions stored in a memory on a computer would function to prompt the computer to transfer and store data files. *See e.g.*, EX1014 (Dye), [0044]-[0046]; *see also* §VIII.A above.

427. Annotated FIG. 1 below shows Franke's local proxy server 150 (colored purple) coupled to local storage 190 (colored red). *See* EX1005 (Franke), FIG. 1 (annotated below). Additionally, FIG. 1 shows LAN users 130-140 (colored blue) connected to local proxy server 150 in the LAN 190, and shows local proxy server 150 further connected to central server 110 and project data storage 185 (colored yellow), and WAN user 120 connected to WAN 160. *Id.*



EX1005 (Franke), FIG. 1 (annotated)

428. The function of Franke's server application 175 (part of local proxy server 150 colored purple in FIG. 1 above) "to "forward or store the project data from the users on central server 110" (colored yellow) is consistent with the claimed function of the transfer control unit of transferring and storing media data to another storage device. *See* EX1005 (Franke), [0042]; *see also* EX1001 ('101 patent), 6:28-40. Franke describes that "central server 110 stores media data from LAN users 130-140 or WAN user 120 as media data 180 in media storage 185 [and] can **receive media data 180 from LAN users 130-140 through local proxy server 150**." *Id.*, [0037], [0047] ("Local proxy server 150 includes a server application 175, which

can … store media data or object on central server 110 for the LAN users 130-140"), [0008].

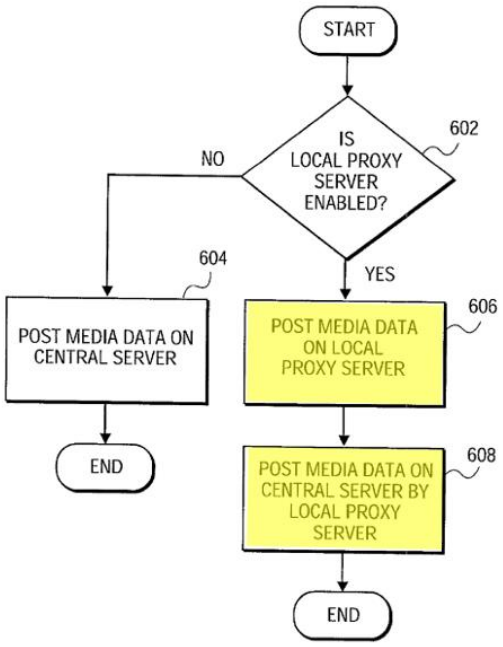


FIG. 6

EX1005 (Franke), FIG. 6 (annotated)

429. Additionally, Franke further describes the function of the server application in its FIG. 6 (annotated above), where media data is posted on the local proxy server (step 606) and is further posted on central server by the proxy server (608). EX1005 (Franke), FIG. 6 (annotated above), [0079]-[0080], [0081] ("Local proxy server 150 is thus responsible for storing or posting the media data to central server 150 for LAN users 130-140.")

430. **In sum**, regarding claim limitation [1b-i], Franke discloses "a transfer control unit [Franke's proxy server application] adapted to transfer [Franke's post] and store [Franke's store] part of held digital contents [Franke's media data] in the internal storage device [Franke's local storage within local proxy server] to a network storage device [Franke's central server]."

431. For similar reasons, Franke discloses claim limitations [6b-i], [7b-i], and [12b-i]. *See* §I above (Table of Claims).

5. Claim [1b-ii]: wherein the network storage device is connected to the network and is capable of storing data

432. Franke discloses claim limitation [1b-ii].

433. For example, Franke describes that the central server ("network storage device") is connected to a network and stores content ("is connected to the network and is capable of storing data"). EX1005 (Franke), [0008] ("users are interconnected via a first network and connected to a second network [where t]he second network includes a central server and the first network includes a selectively enabled local proxy server [and] Media data is stored on the local proxy server and stored on the on the central server by the local proxy server if the local proxy server is enabled"), *see also generally* [0033]-[0036], FIG. 1 (annotated below); *see* §VIII.B above.

434. Franke's annotated FIG. 1 below shows that the proxy server 150 (colored purple) is connected to the central server (colored yellow) by the network (colored green) that includes the WAN and LAN. *See* EX1005 (Franke), FIG. 1

(annotated below). The central server 130 is connected via the WAN and LAN to the local proxy server 150. *Id.* Franke describes that " **central server 110 can store**, manage, and distribute **media data and object data** in a project database to facilitate collaboration between LAN users 130-140 and WAN user 120 on the same project." *Id.*, [0033].

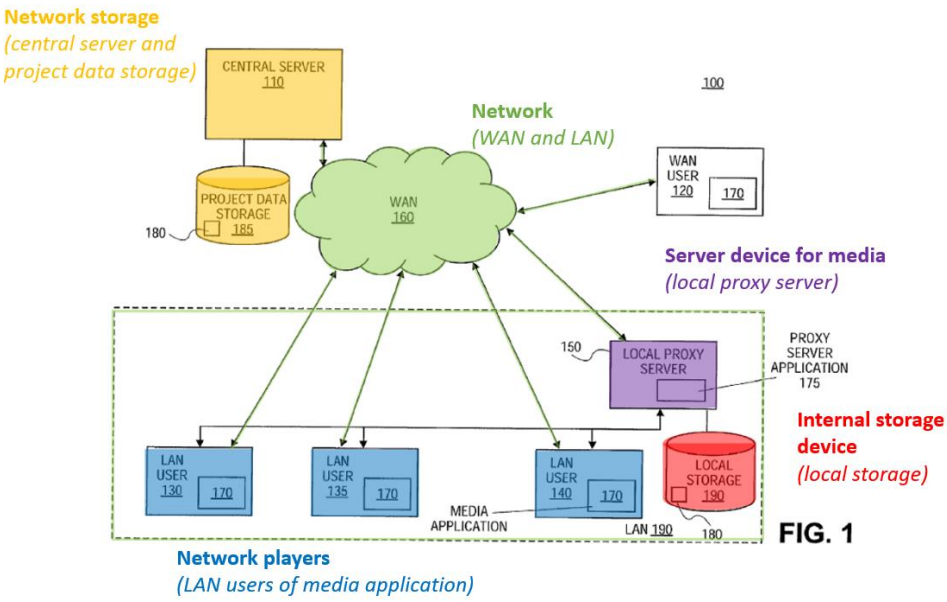


FIG. 1

EX1005 (Franke), FIG. 1 (annotated)

435. **In sum**, regarding claim limitation [1b-ii], Franke discloses that "the network storage device [Franke's central server] is connected to the network [Franke's WAN and LAN] and is capable of storing [Franke's stores] data [Franke's media and object data]."

436. For similar reasons, Franke discloses claim limitations [6b-ii], [7b-ii], and [12b-ii]. *See* §I above (Table of Claims).

6. Claim [1b-iii]: wherein said transfer control unit does not transfer, from the internal storage device to the network storage device, the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device

437. Franke discloses claim limitation [1b-iii].

438. Franke describes that when the central server is not enabled, the local proxy server does not transfer media content to the central server. EX1005 (Franke), [0009] ("media data is stored on the local proxy server and stored on the central server by the local proxy server if the local proxy server is enabled."), [0051] ("if enabled, local proxy server 150 stores project data … on central server 110"), *see also generally* [0078]-[0080].

439. Franke also describes that the status of storing data from the local proxy server to the central server is monitored and reflected in the presence of a completion signal when a transfer is complete, to prevent users from downloading incompletely stored data files. *Id.*, [0094] ("By using the completion signal, a user can avoid downloading an incomplete copy of media data"), [0109], [0096], [0103], [0108], [0036]. For example, Franke describes that the presence of "the requested media is on central server 110" is "indicated by a completion signal." EX1005 (Franke), [0109]. Franke's discussion of and use of a completion signal to indicate when a file stored on the central server is a complete file indicates a preference to ensure

that users do not interact with partially saved files; this represents a preference that provides additional efficiency to the system. *See id.*, [0005]-[0006], [0094].

440. A POSITA would have understood from Franke's description of the enablement of the servers dictating the mechanism for storage of data that when the central server is not enabled, the network connection between the proxy server (located on the LAN) and central server (located on the WAN) are in a state in which digital contents could not be recovered if a network failure occurs (*e.g.*, if the central server were not enabled or otherwise unavailable) ("digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device"). *Id.*, [0078], [0051], [0078]-[0080], [0115].

441. Others also have known of network failures and their effects. *See*, *e.g.*, EX1022 (Adrangi), 13:16-21 ("A network failure may have a number of potential causes, including, for example, unavailability of the source or destination site (*e.g.*, because servers are down), extreme network congestion, unavailability of network communication channels, and various types of software errors"); *see also* §VIII.B.2 above.

442. In such situations, Franke's proxy server (including server application) ("transfer control unit") would not transfer the contents to the central server ("does not transfer, from the internal storage device to the network storage device"), and

would instead wait to be able to transfer to the central server; thereafter, the completion signal would reflect the not-completed transfer. EX1005 (Franke), [0108] ("If the requested media data is not on central server 110, the user waits until the requested data is on central server 110 [as] a local proxy server may be in the process of storing or posting the media data to central server 110"), [0005] ("if the central server is experiencing a high volume of traffic, the central server will be slow in delivering requested data to LAN users [and l]ow bandwidth connections to the central server can exacerbate the situation"), [0115].

443. When a proxy server is not enabled, it is not accessible on the network, which is functionally similar to the result of a loss of connectivity on the network; *see also* EX1005 (Franke), [0094] (describing indication of incomplete transfers of media data from the local proxy server to the central server using completions signals), [0096], [0103], [0108], [0036].

444. This is also functionally equivalent to when a computer is down or lost power and is consistent with Patent Owner's infringement contentions mapping this limitation to not transferring recorded data when there is a power loss or loss of connectivity. *See* EX1102 (Infringement Contentions Appendix E-3), 15 (mapping this limitation to not transferring recorded data when there is a power loss or loss of connectivity).

445. **In sum**, regarding claim limitation [1b-iii], Franke discloses or renders obvious "said transfer control unit [Franke's proxy server application] does not transfer [Franke's post], from the internal storage device [Franke's local storage internal to local proxy server] to the network storage device [Franke's central server], the digital contents [Franke's media data] that cannot be recovered [would fail to be copied, per Franke] if a network failure [Franke's non-enabled server] occurs during the transferring [during posting, per Franke] of the digital contents [Franke's media data] from the internal storage device [Franke's local storage internal to local proxy server] to the network storage device [Franke's central server]."

446. For similar reasons, Franke discloses the corresponding feature of claim limitation [7b-iii]. *See* §I above (Table of Claims).

7. Claim [1c-i]: a list information transmission unit adapted to respond to a list presentation request for the held digital contents of the server device for media from the network player by transmitting list information to the network player

447. Franke discloses claim limitation [1c-i].

448. As an initial matter, a POSITA would have known that listing files and transmitting such list was well known in the art and commonplace. *See* §VIII.C.1 above.

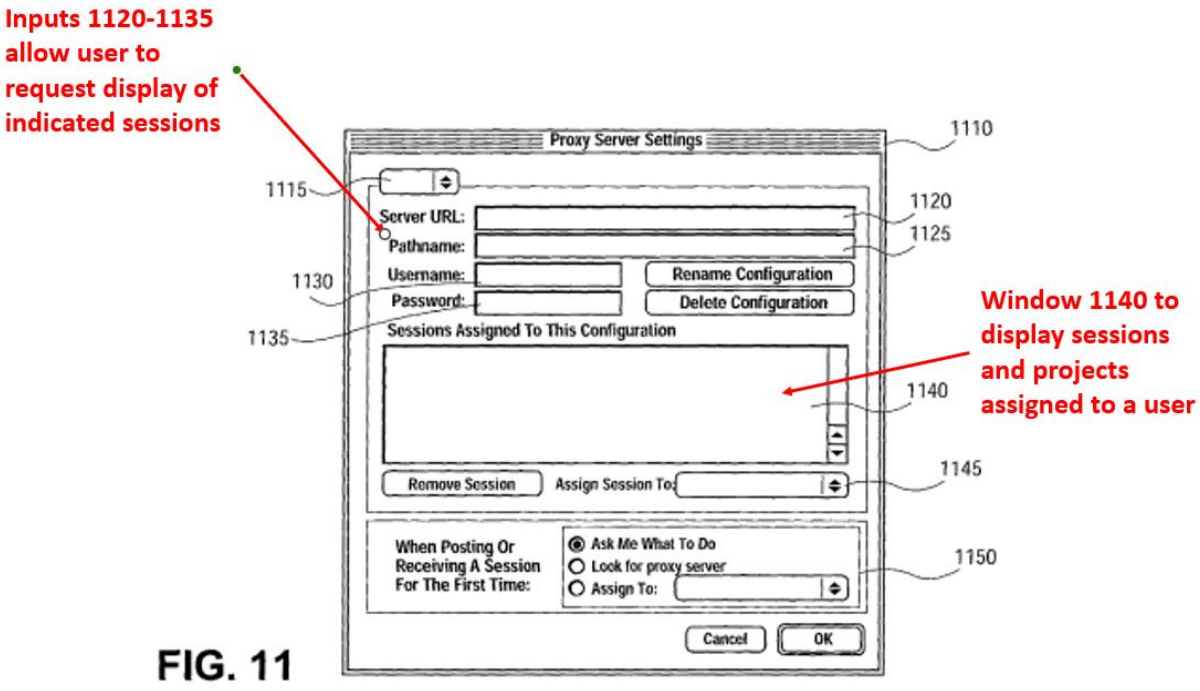
449. Franke discloses that server application 175 ("a list information transmission unit") performs the function of responding to a user request to display

available content and files by displaying a list of the available content on a user device ("adapted to respond to a list presentation request for the held digital contents of the server device for media from the network player by transmitting list information to the network player"). EX1005 (Franke), [0038] ("LAN users 130-140 and WAN user 120 can "log in" to central server 110 and navigate to a session regarding a project and collaborate on the project with other users using media application 170"), [0047] ("Proxy server application 175 can be used to identify, organize, and reference data files in local storage 190."), [0048] ("Proxy server application 175 can determine the location of data files being stored in local storage 190 use the same identifiers for data files generated by LAN users 130-140."), [0049] ("Local proxy server 150 may operate with applications to identify, organize, and reference posted media data 180 for LAN users 130-140 or other users (e.g., WAN user 120).").

450. As I described in §XIV.A.4 above, the '101 patent does not identify particular algorithms or software for performing the functions of the "transfer control unit" and other claimed "units." *See* EX1001 ('101 patent), EX1001 ('101 patent), 3:5-9 ("The program according to the present invention causes a computer to function as each means of the above-described server device for media"), 3:31-40. Instead, the '101 patent labels each 'unit' based on its assigned function and

describes merely that a computer performs the functions based on a program or instructions.

451. Franke's server application 175 performs each of the claimed functions of the 'units' (including the functions of the list information transmission unit) just as the '101 patent's 'computer' performs the functions. A POSITA would have understood that an application (such as Franke's server application 175) may include software and/or algorithms that instruct a computer (in this case the local proxy server 150) to perform a variety of tasks. For example, it would have been obvious to a POSITA that software including instructions stored in a memory on a computer would function to prompt the computer to respond to a user request to display available content and files by displaying a list of the available content. *See* §§VIII.A above, VIII.C above.



EX1005 (Franke), FIG. 11 (annotated)

452. For example, as annotated FIG. 11 above shows, Franke provides a user interface 1110 which "depicts a local proxy server… **dialog interface [including] a plurality of inputs 1115 through 1150** allowing a user to configure settings for posting or downloading media data" where "[a] **window 1140 can display** sessions or **projects assigned to the user**." *Id.*, [0112]-[0113]. The window in the user interface provides a list of content available for selection by the user. *Id.*, [0113] ("A window 1140 can display sessions or projects assigned to the user.").

453. **In sum**, regarding claim limitation [1c-i], Franke discloses and/or suggests "a list information transmission unit [Franke's server application] adapted to respond [display, per Franke] to a list presentation request [Franke's user input in

dialog interface for list of projects] for the held digital contents [Franke's media data] of the server device for media [Franke's local proxy server] from the network player [Franke's LAN User] by transmitting [Franke's displaying to the user] list information [Franke's list of projects] to the network player [Franke's LAN User]."

454. For similar reasons, Franke discloses claim limitations [6c-i], [7c-i], and [12c-i]. *See* §I above (Table of Claims).

8. Claim [1c-ii]: wherein the list information lists the digital contents left in the internal storage device and the digital contents transferred from the internal storage device to the network storage device and stored in the network storage device

455. Franke renders obvious claim limitation [1c-ii].

456. Franke describes that the user interface displays a list of projects ("list information lists the digital contents") that can be retrieved from either the local proxy server or central server, depending on where they are stored ("left in the internal storage device and the digital contents transferred from the internal storage device to the network storage device and stored in the network storage device"). EX1005 (Franke), [0042], [0049], [0112]-[0113], FIG. 11.

457. Franke describes that the proxy server application 175 provides information to users "if requested data files are being downloaded from central server 110" and that the proxy server application 175 can also use identifiers for data files to "determine the location of data files being stored in local storage 190." EX1005 (Franke), [0048], [0070] ("API 193 may determine whether posted media

data 180 is stored on local proxy server 150 and access it from local proxy server 150 if requested by media application 170 [or if] API 193 determines that posted media data 180 is not stored on local proxy server 150, API 193 may download posted media data 180 from central server 110").

458. From Franke's disclosure that the proxy server application 175 can identify and search for user requested data files stored in the proxy server ("digital contents left in the internal storage device") or data files located on the central server ("transferred from the internal storage device to the network storage device and stored in the network storage device"), a POSITA would have understood that the available sessions and projects displayed to the user in the user interface include data files stored in the central server and data files stored in the proxy server. *See* §XIV.A.7 above; EX1005 (Franke), [0048], [0070], FIG. 10.

459. **In sum**, regarding claim limitation [1c-ii], Franke discloses that "the list information [Franke's list of projects] lists [lists, per Franke] the digital contents [Franke's media data] left in [Franke's stored in] the internal storage device [Franke's local storage within local proxy server] and the digital contents [Franke's media data] transferred [posted, per Franke] from the internal storage device [Franke's local storage within local proxy server] to the network storage device [Franke's central server] and stored [posted and stored, per Franke] in the network storage device [Franke's central server]."

460. For similar reasons, Franke renders obvious claim limitations [6c-ii], [7c-ii], and [12c-ii]. *See* §I above (Table of Claims).

9. Claim [1c-iii]: wherein the list information maintains a tree structure of the digital contents in the internal storage device before transferring the digital contents to the network storage device

461. Franke renders obvious claim limitation [1c-iii].

462. As an initial matter, a POSITA would have known that file tree structures and hierarchical file systems were well known in the art and commonplace. *See* §VIII.C.1 above.

463. Franke maintains the list of contents ("the list information") stored in the local proxy server in a directory having a tree structure ("maintains a tree structure") by adding contents to the list before they are transferred to the central server, and continuing to monitor and record a status of the content after the transfer to the central server ("of the digital contents in the internal storage device before transferring the digital contents to the network storage device"). EX1005 (Franke), [0083] ("folders … to store common media data"), [0084] ("a standard filing system … is used to organize and store the media data"), [0084] ("an exemplary folder hierarchy").

464. A POSITA would have understood that such file system having a tree structure was conventional and well-known as a system for allowing users to access remote files and directories as if they were local. *See* §VIII.C.1 above. A POSITA

would have additionally understood that modern file systems are hierarchical and are organized in a "tree structure"—that is, a tree structure refers to a database that uses a hierarchical system to organize content using nested folders and subfolders to categorize like data. This was also known to others and was described in other publications. *See e.g.*, EX1009 (Kageyama), [0084] ("a tree structure containing a plurality of containers arranged in hierarchical levels… [t]he tree structure… has a "ROOT" container that forms a first level [below which] a plurality of containers are arranged").

465. As shown below in the screen-grab of Franke's paragraphs [0084]-[0092] reproduced below, Franke provides "an exemplary folder hierarchy" with folders that illustrate an example of a tree structure of the list information. EX1005 (Franke), [0084], *see also generally* [0084]-[0092]. The (intentional) indentation of the paragraphs in the below image illustrate the nested configuration of the folders that makes up the "tree" structure with a root directory, 'branches' formed from project folders, and smaller branches formed from the contents of those project folders, as a POSITA would have recognized. *See e.g.*, EX1009 (Kageyama), [0084] ("a tree structure containing a plurality of containers arranged in hierarchical levels"), FIGS. 4A-C, 6-7, 9, 12; *see* §VIII.C.1 above.

[**0085**] Parent Directory (root directory on the local proxy server)

[**0086**] Project1 (folder)

[**0087**] Aaron1-123456789123-datafile1.doc (file)

[**0088**] Mike1-894576890532-datafile1.doc (file)

[**0089**] Project2 (folder)

[**0090**] Aaron1-72384732874-picture.jpg (file)

[**0091**] Mike1-77773234234-mynotes.txt (file)

[**0092**] Project3 (folder) . . .

EX1005 (Franke), "exemplary folder hierarchy" of ¶¶[0085]-[0092]

466. As shown in Franke's FIGS. 6 and 7 (annotated side by side below), the organization of newly received media data into the folder hierarchy of the local proxy server ("the list information maintains a tree structure of the digital contents in the internal storage device") occurs when media is posted to the server to be displayed to the user regardless of where the media is located in the system ("before transferring the digital contents to the network storage device"). EX1005 (Franke), [0099] ("If the media data having the unique ID is stored on local proxy server 150, a check is made to determine if there is a completion signal for the media data (stage 810)"), FIGS. 6-7; *see also* EX1009 (Kageyama), [0006]-[0009].

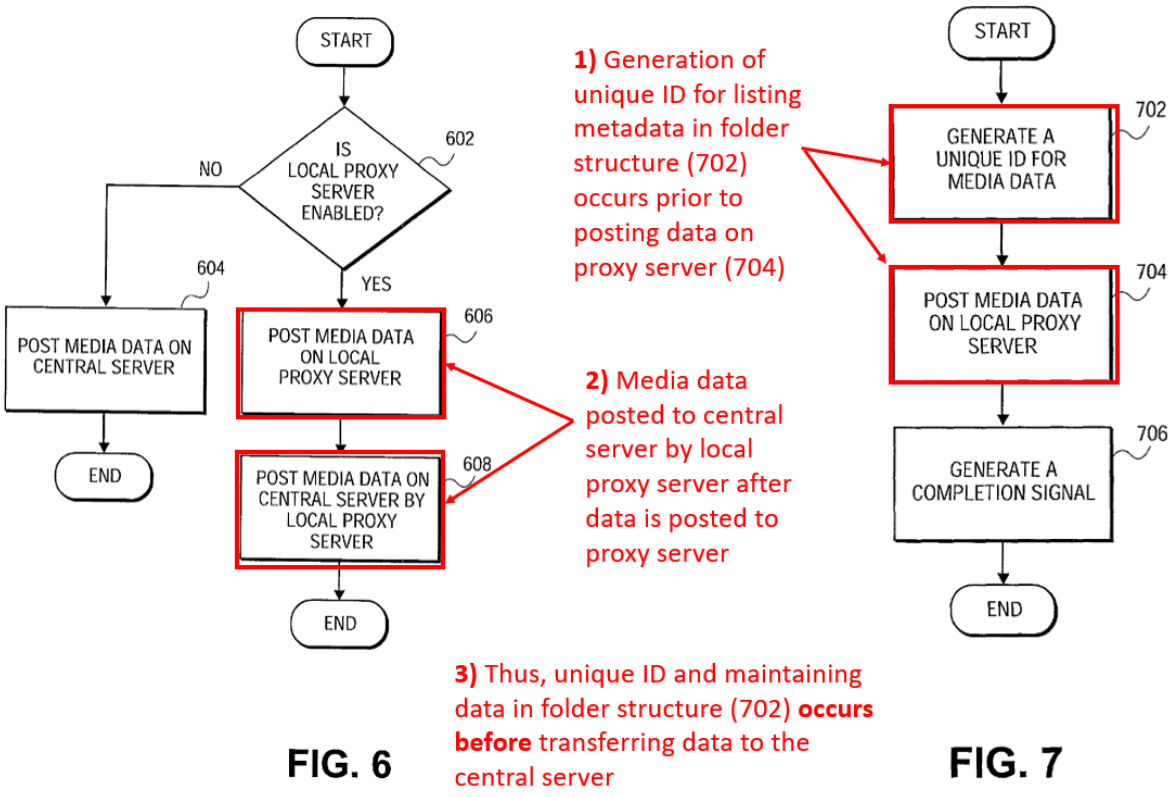


FIG. 6 FIG. 7

EX1005 (Franke), FIGS. 6 (left) and 7 (right) (annotated)

467. **In sum**, regarding claim limitation [1c-iii], Franke discloses and/or suggests "the list information [Franke's server application] maintains a tree structure [Franke's folder hierarchy] of the digital contents [Franke's media data] in the internal storage device [Franke's local storage in local proxy server] before transferring [Franke's posting] the digital contents [Franke's media data] to the network storage device [Franke's central server]."

468. For similar reasons, Franke renders obvious claim limitation [6c-iii] and corresponding features of claim limitations [7c-iii] and [12c-iii]. *See* §I above (Table of Claims).

10. Claim [1d]: a search unit adapted to respond to a data transmission request for the held digital contents from the network player by searching for a location where the held digital contents are currently stored

469. Franke renders obvious claim limitation [1d].

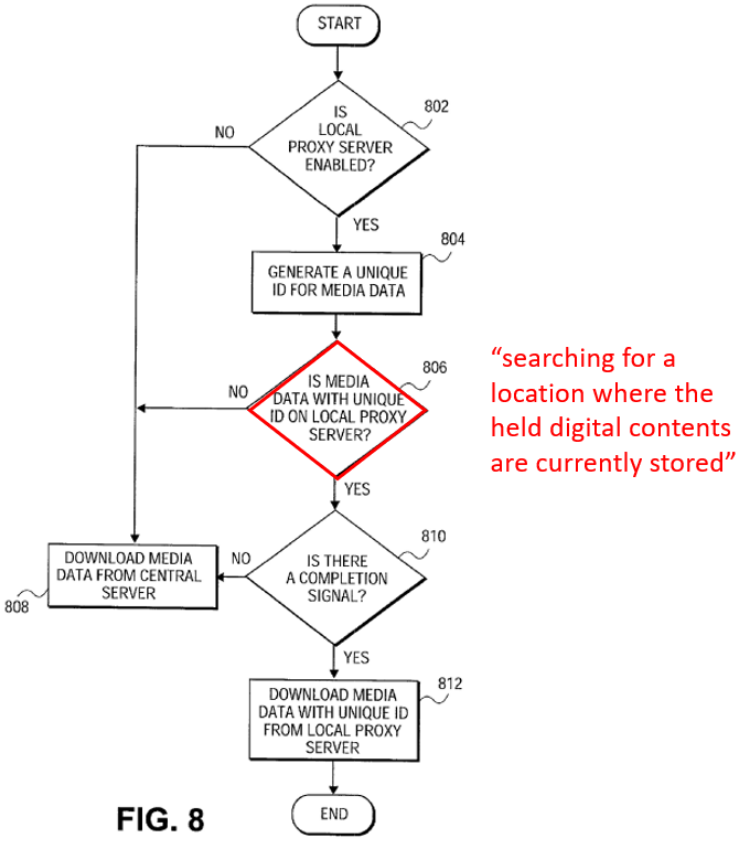
470. Franke describes that the proxy server application performs a check to determine the location of selected data files. EX1005 (Franke), [0059] ("Memory 265 may store instructions or code for implementing a program … and an application programming interface (API)"), [0099] ("A check is then made to determine if the media data having the unique ID is stored on local proxy server 150 (stage 806) [and if] no media data having the unique ID is stored on local proxy server 150, the media data is downloaded from central server 110 (stage 808)"), [0048], [0047], [0117], FIG. 2.

471. As I described in §XIV.A.4 above, the '101 patent does not identify particular algorithms or software for performing the functions of the claimed "units." *See* EX1001 ('101 patent), 3:5-9. Instead, the '101 patent labels each 'unit' based on its assigned function and describes merely that a computer performs the functions based on a program or instructions. Franke's server application 175 performs each of the claimed functions of the 'units' (including the functions of the search unit) just as the '101 patent's 'computer' performs the functions.

472. A POSITA would have understood that an application (such as Franke's server application 175) may include software and/or algorithms that

instruct a computer (in this case the local proxy server 150) to perform a variety of tasks. Indeed, an "algorithm" is "[a] finite sequence of steps for solving a logical or mathematical problem or performing a task." EX1017 (Microsoft Computer Dictionary), 5. For example, it would have been obvious to a POSITA that software including instructions stored in a memory on a computer would function to prompt the computer to respond to a user request to access media data on a user device by performing a search to determine a location of the requested content within storage devices. Indeed, such search methodology was well-known and conventional. *See* §VIII.C above; *see also e.g.,* EX1024 (Belknap), 6:64-7:4.

473. In performing a check (for example, step 806 in FIG. 8), Franke's server application performs the same function as the claimed "search unit" of determining a location where desired media data is stored (*i.e.*, "is media data on local proxy server"). As shown in annotated FIG. 8 below, if the result of step 806 is a determination that no, the desired media content is not located on the local proxy server, the media data is instead downloaded from the central server. *Id.,* [0099].



EX1005 (Franke), FIG. 8 (annotated)

474. It would have been obvious to a POSITA that the same determination of location of requested media data described by Franke with regard to the API 193 would also be a function of the proxy server application and/or API of the local proxy server based on Franke's similar description of the location determination function of the server application and API. EX1005 (Franke), [0099], [0059], [0070]. Others confirm this understanding. *See e.g.*, EX1024 (Belknap), 6:64-7:4 ("After receiving the request to retrieve … the media object, the object server 220 will identify whether the media object is located locally within the object store 222

or at a remote location within the media archive 140 or one of the media servers 130").

475. **In sum**, regarding claim limitation [1d], Franke discloses a search unit [Franke's server application] adapted to respond to a data transmission request [a request from Franke's LAN user to download media data, per Franke] for the held digital contents [Franke's media data] from the network player [Franke's LAN user] by searching [Franke's performing a check] for a location [to determine a device, per Franke] where the held digital contents [Franke's media data] are currently stored [Franke's stored]."

476. For similar reasons, Franke renders obvious claim limitations [6d], [7d], and [12d]. *See* §I above (Table of Claims).

11. Claim [1e]: a digital contents data transmission processing unit adapted to allow the corresponding data in held digital contents to be stream-delivered from the network storage device to the network player, if the result of search shows the network storage device

477. Franke discloses claim limitation [1e].

478. Franke describes that the server application ("a digital contents data transmission processing unit") transfers requested data files ("adapted to allow the corresponding data in held digital contents to be stream-delivered") from the central server to the LAN users if the content is stored in the central server ("from the network storage device to the network player, if the result of search shows the network storage device"). EX1005 (Franke), [0047] ("Proxy server application 175

may also be used to **transfer such data files to LAN users**"), [0054] ("if, e.g., media data 180 is not stored in local storage 190, local proxy server 150 can **request the data from central server 110**"), *see also* [0042]-[0048], [0053]-[0054], [0103], [0117].

479. As I described in §XIV.A.4 above, the '101 patent does not identify particular algorithms or software for performing the functions of the claimed "units." *See* EX1001 ('101 patent), 3:5-9. Instead, the '101 patent labels each 'unit' based on its assigned function and describes merely that a computer performs the functions based on a program or instructions. Franke's server application 175 performs each of the claimed functions of the 'units' (including the functions of the digital contents data transmission unit) just as the '101 patent's 'computer' performs the functions. EX1005 (Franke), [0047], [0054], [0042], [0048], [0053]-[0054], [0103], [0117].

480. A POSITA would have understood that an application (such as Franke's server application 175) may include software and/or algorithms that instruct a computer (in this case the local proxy server 150) to perform a variety of tasks. For example, it would have been obvious to a POSITA that software including instructions stored in a memory on a computer would function to prompt the computer to stream requested content from the storage device to the user device. *See* §§VIII.E-VIII.E above.

481. Franke describes that in its system, content is saved to the local proxy server by the local users, unless the local proxy server is not enabled, in which case the content is saved to the central server. *Id.*, [0079] ("If local proxy server 150 is enabled, the media data is stored or posted on local proxy server 150"), [0078] ("If local proxy server 150 is not enabled, media data is stored or posted on central server 110"), FIG. 8. As discussed above, the local proxy server also stores content to the central server and can mirror content stored on the central server by remote users. *Id.*, [0080] ("Local proxy server 150 then stores or posts the media data on central server 110"), FIG. 8 (annotated below).

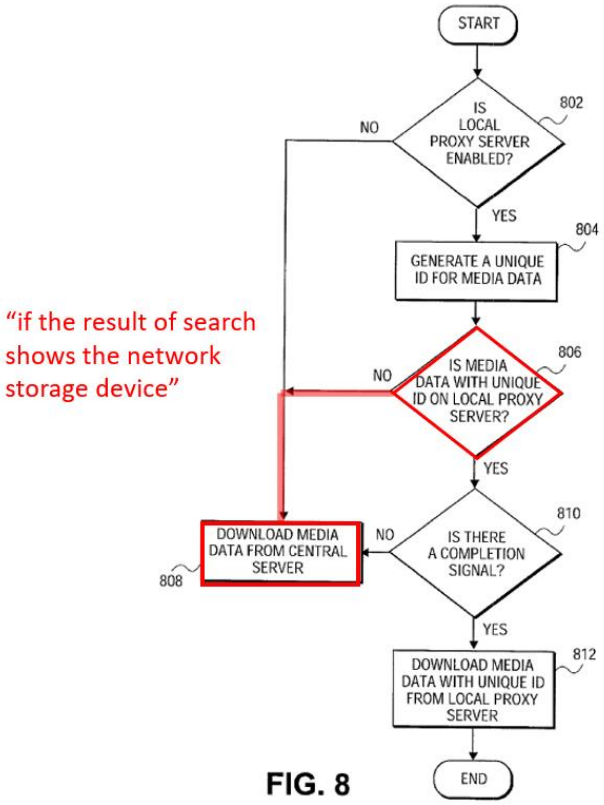
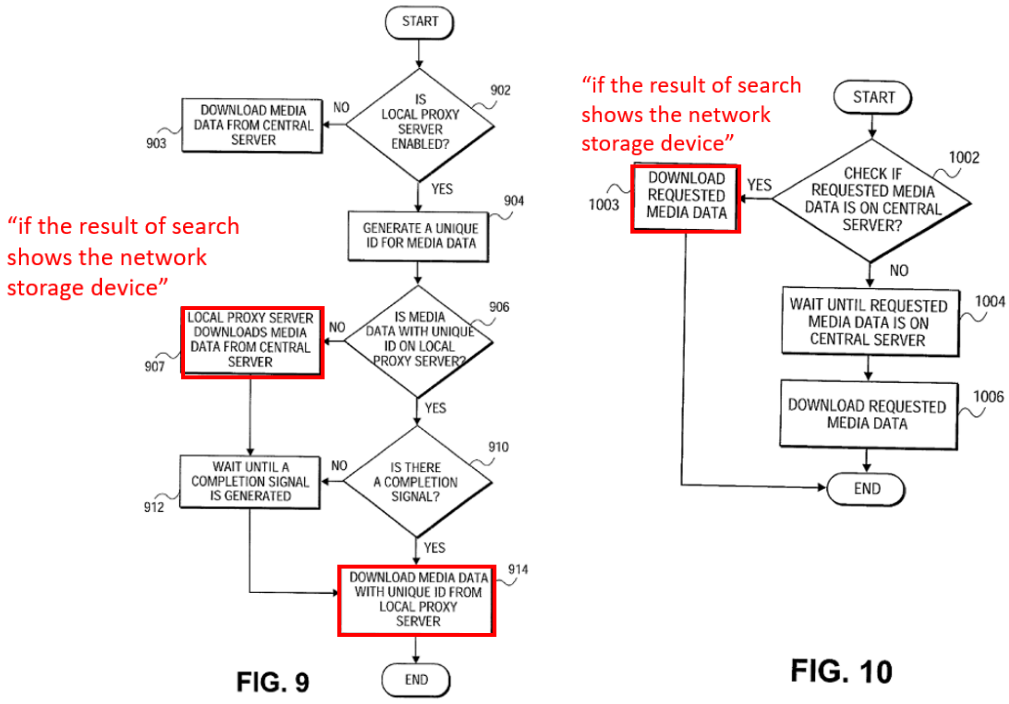


FIG. 8

EX1005 (Franke), FIG. 8 (annotated)

482. Alternatively, Franke describes that when the data is located on the central server (for example, the response to step 806 in annotated FIG. 8 above is "no"), the media data is downloaded from the central server to the user (see step 808 in FIG. 8). EX1005 (Franke), [0048] ("local proxy server 150, using proxy server application 175, can download project data, data files, or other types of data for LAN users 130-140 even if any of the users are not in a direct connection or session with a project maintained by central server 110"), [0053] ("LAN users 130-140 can also request project data from local proxy server 150."), [0099] ("the media data is downloaded from local proxy server 150"), *see also generally*, [0107]-[0109], FIGS. 8-10.

483. Franke also illustrates other situations in FIGS. 9 and 10 (annotated side by side below), where the content located on the central server is downloaded from the central server to the local proxy server for distribution to the local users (FIG. 9) or downloaded directly from the central server when it becomes available (FIG. 10). EX1005 (Franke), [0048] ("Proxy server application 175 provides download status information to LAN users 130-140 if requested data files are being downloaded from central server 110"), [0053] ("If local proxy server 150 does not contain the requested project data, local proxy server 150 can request it from central server 110 and then distribute it to LAN users 130-140"), [0099], [0107]-[0109], FIGS. 8-10.



EX1005 (Franke), FIGS. 9 (left) and 10 (right) (annotated)

484. A POSITA would have understood that multi-media streaming was a well-known and conventional process by the Critical Date, and would have further understood that Franke discloses providing media data to the LAN users by stream-delivering the content. *See* §VIII.E above; EX1005 (Franke), [0103] ("**a streaming process** can be implemented to download segments of the requested media data as soon as the segments become available on local proxy server 150"), [0109] ("**a streaming process** can be implemented to download segments of the requested media data as soon as the segments become available on central server 110").

485. **In sum**, regarding claim limitation [1e], Franke discloses "a digital contents data transmission processing unit [Franke's server application] adapted to

allow the corresponding data [Franke's requested media data] in held digital contents [Franke's media data] to be stream-delivered [by Franke's streaming process] from the network storage device [Franke's central server] to the network player Franke's [Franke's LAN user], if the result of search [if the check, per Franke] shows the network storage device [Franke's shows the media data is stored in the central server]."

486. For similar reasons, Franke discloses claim limitations [6e], [7e], and [12e]. *See* §I above (Table of Claims).

12. Claim [1f]: wherein the server device for media is a media player

487. Franke discloses claim limitation [1f].

488. Franke describes that the local proxy server can be a personal computer, which a POSITA would have understood is a media player. EX1005 (Franke), [0044], [0032]; *see* §VIII.A above.

489. As I described above, there are various characteristics common to computers, including having input/output devices like displays or speakers that enable a user to interact with the computer. *See* §VIII.A above. Franke's disclosure that "Local proxy server 150 is a computing system" such as "Apple® Power Macintoshes or Intel® Pentium-based **personal computers** running a version of the Windows® operating system" discloses that the local proxy server ("server device for media") is a media player, because a POSITA would have understood that such

personal computer devices including I/O devices allowing for media to be played directly from the device. EX1005 (Franke), [0044], [0032.

490. **In sum**, regarding claim limitation [1f], Franke discloses that "the server device for media [Franke's local proxy server] is a media player [Franke's personal computer able to directly play media]."

491. For similar reasons, Franke discloses claim limitations [6f], [7f], and [12f]. *See* §I above (Table of Claims).

B. Claim 6

492. As shown in §I above (Table of Claims), the claim 6 limitations [6pre]-[6b-ii] and [6c-i]-[6f] are the same as limitations [1pre]-[1b-ii] and [1c-i]-[1f]. Accordingly, Franke discloses the claim limitations of [6pre]-[6b-ii] and [6c-i]-[6f] for the same reasons explained above with regard to the corresponding elements of claim 1, and I do not repeat the analysis here. *See* §XIV.A above.

1. Element [6b-iii]: wherein the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device is transferred after obtaining permission from a user

493. Franke renders obvious claim limitation [6b-iii].

494. Franke describes that users are presented with options for how to proceed when there is an error in contacting a server to initiate a transfer ("digital contents … [are] transferred after obtaining permission from a user"), such as when a server is not enabled and content cannot be successfully transferred ("the digital

contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device"). EX1005 (Franke), [0112] ("User interface 1110 includes a plurality of inputs 1115 through 1150 allowing a user to configure settings for posting or downloading media data"), [0113] ("if the 'Ask Me What to D[o]' option is selected, a help type window is displayed informing the user of options that can be selected for user interface 1110."), [0115] ("user interface 1410 is provided if there is an error in communicating with local proxy server 150. For example, window 1415 can display a message such as 'Cannot communicate with Local Proxy Server. Edit your settings and select OK.'"), [0036] ("send status updates to [the] users"), [0103] ("completion signal is generated"), *see also generally* [0094]-[0096], [0108]-[0109].

495. In particular, when Franke's "ask me what to do" option is selected or enabled on a user device, "a help type window is displayed informing the user of options that can be selected for user interface 1110." EX1005 (Franke), [0113]. A POSITA would have recognized that a help window providing a user with selectable options would be useful when errors occur that disrupt function of the system, such as problems with transfers of content between the local proxy server, central server, or user devices, in view of Franke's discussion of network connection and enablement of the network servers impacting transfer. *Id.*, [0036], [0094]-[0096], [0103], [0108]-[0109].

496. A POSITA would have further understood that it was well-known and conventional to present a user with an option to provide instructions to the system if a failure had occurred or was likely to occur. *See e.g.,* EX1007 (Marotta), [0481]-[0482], [0691]-[0710], FIGS. 20, 41. *See also* §VIII.B.2 above.

497. Franke describes that there may be situations when the proxy server is not enabled, meaning that the proxy server is not accessible to other devices on the network, and as a result, content cannot be stored on the proxy server. EX1005 (Franke), [0009], [0051]. This is a similar result to that which occurs when network failure or poor connection prevent a device from being accessible in a network—the device simply cannot be contacted and transfer to or from the device is not possible. *See* §VIII.B.1 above.

498. Franke's description of non-enabled devices from which transfer is impossible giving rise to the claimed "digital contents that cannot be recovered if a network failure occurs" is consistent with the patent owner's infringement contentions, which map the limitation "digital contents that cannot be recovered if a network failure occurs" of claim element [1b-iii] to content that could not be transferred during a power loss or loss of connectivity. EX1102 (Infringement Contentions Appendix E-3), 15; *see also* EX1005 (Franke), [0094] (describing indication of incomplete transfers of media data from the local proxy server to the central server using completions signals), [0096], [0103], [0108], [0036].

499. It would have been obvious to a POSITA that if the network between the proxy server and central server were in a state in which digital contents could not be recovered if a network failure occurs (*e.g.*, if the central server is not available on the network) ("the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device") then the user is presented with options, including an "Ask me what to do" option and options for proceeding with the storage of the data to the central server ("is transferred after obtaining permission from a user"). EX1005 (Franke), [0108]-[0115], [0345] ("central server 110 handles authentication tasks"), [0049] ("password protection for media data 180"), [0112] ("User interface 1110 includes a plurality of inputs 1115 through 1150 allowing a user to configure settings for posting or downloading media data"), [0005].

500. **In sum**, regarding claim limitation [6b-iii], Franke discloses or renders obvious "the digital contents [Franke's media data] that cannot be recovered [would fail to be copied] if a network failure [Franke's non-enabled server] occurs during the transferring [Franke's during posting] of the digital contents [Franke's media data] from the internal storage device [Franke's local storage internal to local proxy server] to the network storage device [Franke's central server 110] is transferred [is posted, per Franke] after obtaining permission from a user [Franke's after receiving a user response in the user interface]."

501. For similar reasons, Franke renders obvious claim limitation [12b-iii]. *See* §I above (Table of Claims).

C. Dependent Claims 2-5 and 8-11

1. Claim 2: The server device for media according to claim 1, wherein said digital contents data transmission processing unit causes the network storage device to transmit the corresponding data to the server device for media, and then transmits the corresponding data received from the network storage device from the server device for media to the network player

502. Franke discloses claim [2].

503. As an initial matter, this claim describes a standard caching technique, where data is retrieved from a backend device (network storage device), through an intermediary, proxy, or cache device (server device for media), and then to its final decisional for consumption by users/applications (network player). *See* §VIII.F above.

504. Franke describes that the server application ("digital contents data transmission processing unit") downloads requested content from the central server to the local proxy server ("causes the network storage device to transmit the corresponding data to the server device for media") before transferring the content to the LAN users ("and then transmits the corresponding data received from the network storage device from the server device for media to the network player"). EX1005 (Franke), [0054] ("if, e.g., media data 180 is not stored in local storage 190, local proxy server 150 can request the data from central server 110 to make it

accessible to LAN users 130-140"), [0047], [0064], [0068]-[0070], FIG. 9; *see also* §XI.A.11 above.

505. This is illustrated in annotated FIG. 9 below, which shows that when requested media data is not located on the local proxy server (result of step 906 is "no"), the local proxy server downloads the requested media data from the central server (steps 907, 914, highlighted in red).

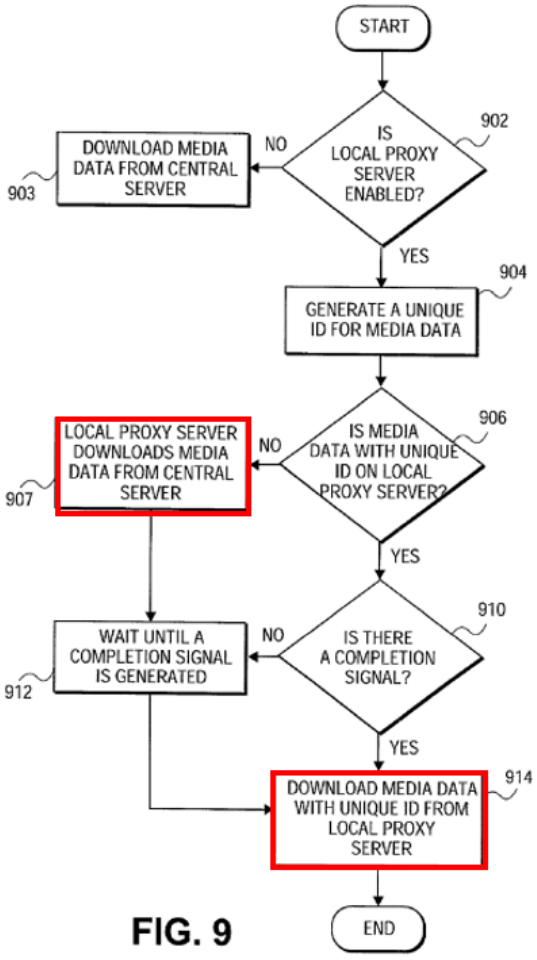


FIG. 9

EX1005 (Franke), FIG. 9 (annotated)

506. Franke describes that "if no media data having the unique ID is stored on local proxy server 150 and local proxy server 150 is enabled, the media data is copied from central server 110 by local proxy server 150 [such that the] local proxy server 150 downloads the requested media data with the unique ID from central server 150." EX1005 (Franke), [0102]. Because Franke's "[l]ocal proxy server 150 can act as a client to central server 110 and act as an intermediary between central server 110 and LAN users 130-140" the media data downloaded from the central server is then transmitted to the LAN users. EX1005 (Franke), [0072].

507. **In sum**, regarding claim [2], Franke discloses that "said digital contents data transmission processing unit [Franke's server application] causes the network storage device [Franke's central server] to transmit [Franke's download] the corresponding data [Franke's requested media data] to the server device for media [Franke's local proxy server], and then transmits [Franke provides a download] the corresponding data [Franke's requested media data] received from the network storage device [Franke's central server] from the server device for media [Franke's local proxy server] to the network player [Franke's LAN user]."

508. For similar reasons, Franke discloses claim [8]. *See* §I above (Table of Claims).

2. Claim 3: The server device for media according to claim 1, wherein said digital contents data transmission processing unit transmits the corresponding data and information for identifying the network storage device to the network player, and causes the network storage device to directly transmit the corresponding data to the network player

509. Franke discloses claim [3].

510. As an initial matter, this claim describes an obvious idea—that to be able to retrieve data from some remote location, you have to identify that location. All network storage systems, whether client-server ones or more complex distributed storage systems, have to identify the location of each data item before transmitting it anywhere. Moreover, a POSITA would have understood that directly transmitting a piece of data, without going through an intermediary (*e.g.*, a proxy or cache device), is more efficient network-wise. *See* §VIII.D above.

511. Franke describes that the server application 175 ("digital contents data transmission processing unit") identifies the location of requested content to the LAN user ("transmits the corresponding data and information for identifying the network storage device to the network player") and the requested content is transmitted directly to the LAN user from the central server ("causes the network storage device to directly transmit the corresponding data to the network player"). *See* §XIV.A.11 above; EX1005 (Franke), [0036] ("Central server 110 can also handle download requests of project data from LAN users"), [0070] ("API 193 may download posted media data 180 from central server 110"), [0053] ("To download

project data, LAN users 130-140 and WAN user 120 can request project data from central server 110 directly"), [0099] ("If no media data having the unique ID is stored on local proxy server 150, the media data is downloaded from central server 110 (stage 808)."), [0108] ("If the requested media data is on central server 110, the requested media data is downloaded from central server 110"), *see also generally* [0033]-[0036],.

512. As discussed in §XIV.A.7 above (claim limitations [1c-i]-[1c-ii]), Franke's local proxy server application provides the list of contents and their location information to a user. Additionally, Franke describes that when the local proxy server is not enabled, the LAN user's media application instead receives requested content directly from the central server. EX1005 (Franke), [0108], [0033] ("central server 110 can store, manage, and distribute media data and object data in a project database"), [0035] ("central server 110 can provide media data and object data "on demand" or upon request by a user"), [0068] ("To download posted media data, API 193 can copy posted media from central server 110"), [0053].

When media is located on the central server, requested media data is directly downloaded from the central server

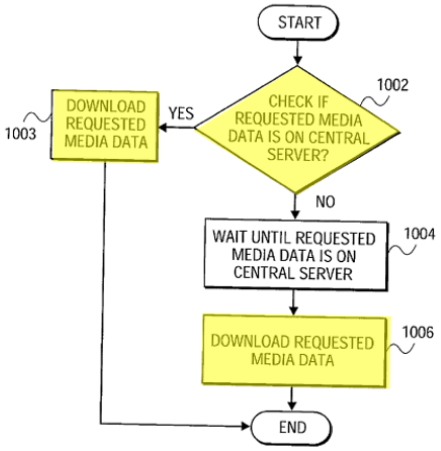


FIG. 10

EX1005 (Franke), FIG. 10 (annotated)

513. With regard to FIG. 10 (annotated above), Franke describes that in some cases, "[a] check is made to determine if the requested media data is on central server 110 (stage 1002) [and, if so], the requested media data is downloaded from central server 110 (stage 1003)." *Id.*, [0108]. Accordingly, Franke describes that, in some cases, when media data is located on the central server, the media data is transmitted directly from the central server to the user. *Id.*, FIG. 10.

514. **In sum**, regarding claim [3], Franke discloses that "said digital contents data transmission processing unit [Franke's server application] transmits [Franke's downloads] the corresponding data [Franke's list of projects] and information for identifying the network storage device [Franke's check result indicating requested media data is stored in the central server] to the network player [Franke's LAN user],

and causes the network storage device [Franke's central server] to directly transmit [Franke's download] the corresponding data [Franke's requested media] to the network player [Franke's LAN user]."

515. For similar reasons, Franke discloses claim [9]. *See* §I above (Table of Claims).

3. Claim 4: The server device for media according to claim 1, further comprising a return control unit adapted to cause the digital contents corresponding to a predetermined condition among the digital contents which have been transferred to the network storage device to be returned from the network storage device to the internal storage device

516. Franke discloses claim [4].

517. As an initial matter, a POSITA would have understood that computers have internal storage devices, as I describe in §VIII.A above. Moreover, this claim describes nothing more than transmitting cached data back to its source or migrating data to make some room. *See* §VIII.D.1 above.

518. As described in element [1e] (§XIV.A.11 above) and claim [2] (§XIV.C.1 above), Franke describes that requested content is transferred from the central server to the proxy server ("cause the digital contents… to be returned from the network storage device to the internal storage device") by the proxy server application ("a return control unit") in response to a user request for content ("the digital contents corresponding to a predetermined condition"), after storing the content in the central server ("the digital contents … among the digital contents

which have been transferred to the network storage device to be returned from the network storage device to the internal storage device"). EX1005 (Franke), [0081] ("media data 180 being stored on central server 110 can be mirrored on local proxy server 150 for access by LAN users 130-140 via LAN 190").

519. Franke's server application 175 performs the same function as the claimed "return control unit," and, as described above, is likely implemented using software to allow a computer (*e.g.,* local proxy server) to perform functions including transferring stored content between storage devices. *See* §VIII.A above. A POSITA would have appreciated that an application is software that "causes a computer to function" in a particular way to achieve a task. EX1001 ('101 patent), 3:5-9. Accordingly, a POSITA would have appreciated that Franke's "server application 175" is implemented as software in a computer to perform functions including transferring content between devices, just as the '101 patent's "return control unit" must be.

520. As I described in §VIII.D.1 above, a POSITA would have understood Franke's description of transferring requested content from the central server to the local proxy server before providing the requested content to the user to be conventional behavior of a proxy server. Additionally, a POSITA would have understood that Franke's reason for transferring requested content from the central server to the proxy server is consistent with the "predetermined reasons" given as

examples in the ’101 patent. For example, the ’101 patent describes that content can be returned to the server device for media for a number of predetermined reasons, including “the digital contents which have recently been subjected to the self-playback among the digital contents which have been transferred to the network storage device 57, or the digital contents which have recently been played whether by self-playback or the network-playback.” EX1001 (’101 patent), 8:5-10.

521. A POSITA would have understood that Franke’s system transmitting requested content from central server to the local proxy server is a return of content based on a predetermined reason; that is, based on the predetermined reason of being requested by the user. This represents a common experience for a POSITA: once a media (*e.g.*, video) has been played once, often it need not consume resources in its current location and can be transferred or moved elsewhere. This is also consistent with the description of predetermined conditions in the ’101 patent. *See e.g.*, *id*., 8:5-10 (describing that predetermined conditions can include “the digital contents which have recently been subjected to the self-playback” or “network playback”).

522. **In sum**, regarding claim [4], Franke discloses that “a return control unit [Franke’s server application] adapted to cause the digital contents [Franke’s media data] corresponding to a predetermined condition [Franke’s requested by a user] among the digital contents [from Franke’s media data] which have been transferred [Franke’s posted] to the network storage device [Franke’s central server] to be

returned [Franke's to be downloaded] from the network storage device [Franke's central server] to the internal storage device [Franke's local storage internal to local proxy server]".

523. For similar reasons, Franke discloses claim [10]. *See* §I above (Table of Claims).

4. Claim 5: The server device for media according to claim 1, wherein said list information transmission unit makes the list information to be transmitted to the network player include information for identifying whether each digital content is currently stored in the internal storage device or the network storage device in the display list of the network player

524. Franke renders obvious claim [5].

525. As an initial matter, this claim describes a file namespace unification scheme where data or files that reside in different locations are presented in a unified view. Including the location with a data item was well known, as I described in §VIII.D above. Moreover, file namespace unification was well known and I personally worked on such techniques, as described in my qualification in §III above.

526. Franke describes that the server application ("list information transmission unit") provides a user interface at a client device with a list of projects stored by the local proxy server ("makes the list information to be transmitted to the network player… in the display list of the network player") with a completion signal indicating whether the data has been completely stored in the local proxy server

("include information for identifying whether each digital content is currently stored in the internal storage device or the network storage device"). EX1005 (Franke), [0093] ("After the media data is completely posted on local proxy server 150, a completion signal is generated … [which] notifies other LAN users 130-140 that the media data has been completely posted or stored on the local proxy server."), [0094] (completion signal presented so that "a user can avoid downloading an incomplete copy of media data."), [0099] ("If there is a completion signal, the media data is downloaded from local proxy server 150 (stage 812)"), [0103] ("If media data having the unique ID is stored on local proxy server 150, a check is made to determine if there is a completion signal (stage 910)…. The completion signal indicates that the media data is completely stored on local proxy server 1150"), *see also generally* [0092]-[0096], [0103]-[0105].

527. As shown in annotated FIG. 8 below, Franke describes that presence of a completion signal indicates that the data is located in and can be downloaded from the proxy server, while an absence of the completion signal indicates the data is not completely stored in the proxy server and "the media data is downloaded from central server 110." EX1005 (Franke), [0099].

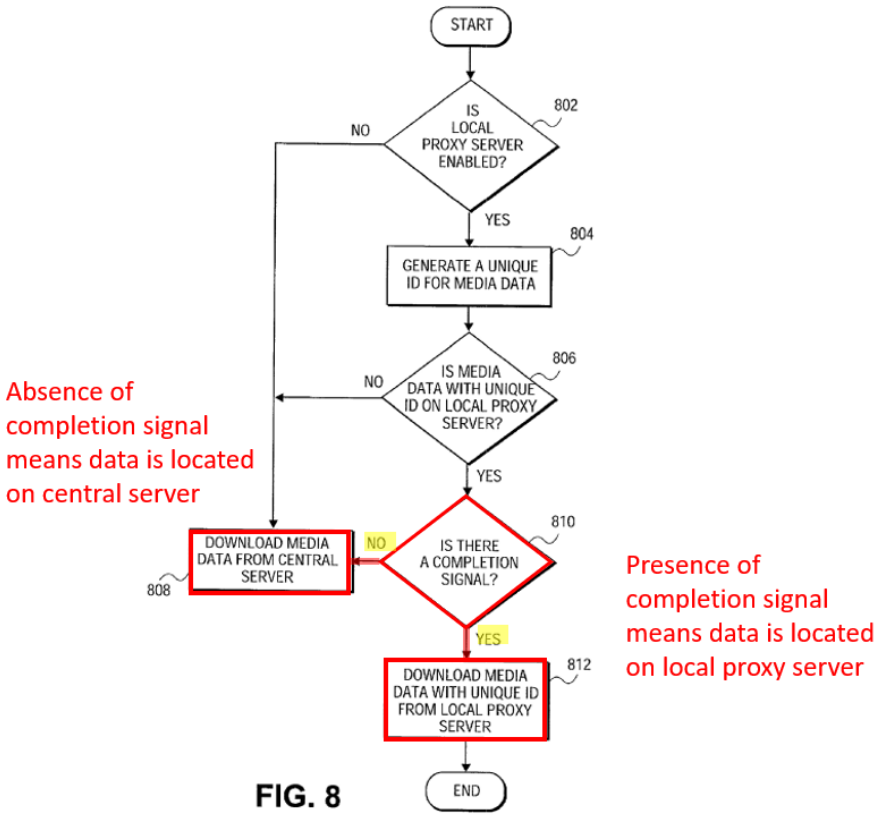


FIG. 8

EX1005 (Franke), FIG. 8 (annotated)

528. Accordingly, Franke's system uses the completion signal to indicate to a user whether particular content has been stored on the central server or if it is located only at the local proxy server. EX1005 (Franke), [0093] ("After the media data is completely posted on local proxy server 150, a completion signal is generated [which] notifies other LAN users 130-140 that the media data has been completely posted or stored on local proxy server 150 and is ready for downloading."), [0094] ("[t]he completion signal can be indicated, e.g., by **appending ".cmpit" to the unique ID for the media data**, **storing an indication in a database record**, or checking the media data size on local proxy server 150 with the required file size for

the media data [or by] **local proxy server 150 broadcasting a signal that the media data has been successfully posted**.").

529. **In sum**, regarding claim [5], Franke discloses that "said list information transmission unit [Franke's server application] makes the list information [Franke's list of projects] to be transmitted [Franke's displayed] to the network player [Franke's LAN users] include information for identifying [Franke's completion signal] whether each digital content [Franke's each listed media data] is currently stored [Franke's posted] in the internal storage device [Franke's local storage internal to local proxy server] or the network storage device [Franke's central server] in the display list [in Franke's project list in the user interface] of the network player [Franke's LAN user]."

530. For similar reasons, Franke renders obvious claim [11]. *See* §I above (Table of Claims).

XV. GROUND 2B: FRANKE IN VIEW OF FIECHTER RENDERS CLAIMS 1-12 OBVIOUS

531. Beyond the relevance of Franke's teachings alone, as described in Ground 2A (*see* §XIV.A.6 and §XIV.B above), the features recited in claim limitations [1b-iii] and [6b-iii], and corresponding features of claim limitations [7b-iii] and [12b-iii] also would have been obvious to a POSITA based on the combined teachings of Franke and Fiechter, for at least the reasons set forth in the Petition which I have read and fully endorse. *See* §I above (Table of Claims). To streamline

the analysis in this declaration, I do not repeat all of the reasons stated in the Petition as to why a POSITA would have found claim elements [1b-iii] and [6b-iii], and corresponding features of claim limitations [7b-iii] and [12b-iii] obvious based on the combination of Franke and Fiechter, but by way of example and to highlight certain points, I make the following additional observations.

532. The combination of Franke and Fiechter renders obvious all other elements of claims 1-12 for the reasons described with respect to those elements in Ground 2A (§XIV above).

533. As I described in §XIV.A.6 above, Franke describes that there are circumstances where one or both of the local proxy server and central server may be inaccessible on the network or not enabled, preventing full transfer of content to or from the server and/or prompting a user for instructions. *See* EX1005 (Franke), [0009] ("media data is stored on the local proxy server and stored on the central server by the local proxy server if the local proxy server is enabled."), [0036] ("if requested project data has not been completely stored or posted central server 110, central server 110 can send status updates to the users or devices connected via WAN 160"), [0051] ("if enabled, local proxy server 150 stores project data … on central server 110"), *see also generally* [0078]-[0080], [0094]-[0096], [0103]-[0109], [0115]. Franke briefly describes the techniques for dealing with non-enabled servers in the network system that would render content "unrecoverable" in that it

would not be able to be successfully transferred to its destination. Fiechter, however, provides additional details about the implementation of such techniques in a system like Franke's. EX1006 (Fiechter), 17:50-59 ("mobile audio device 904 includes capability to detect and compensate for data transmission errors in wireless communication network 906," including detection that the device is in an area "where there is a high incidence of data loss due to propagation errors.").

534. Accordingly, in the combination, in a system, such as that described by Franke, when high risk of data loss is detected due to the non-enablement of one or more servers or storage devices on the network, "one of several options may be taken including informing the user of the likelihood of errors or gaps in the information, asking the user if they would prefer to wait for the information until transmission of the data will be more reliable, or buffering a greater than normal amount of information to be able to continue undisrupted output until the connection is re-established." *Id.*, 17:64-18:3. Put more plainly, in the combination, a system such as Franke's, is able to detect when non-enablement of servers on the network that may compromise transfers over the network is more likely, and in response to the detection either (1) does not transfer content when there is risk of data loss (*e.g.*, waiting), or (2) asks the user to authorize proceeding with the transfer despite the risk (*e.g.*, asking the user to confirm to transfer). *Id.*, 17:64-18:3.

A. Claim [1b-iii]: wherein said transfer control unit does not transfer, from the internal storage device to the network storage device, the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device

535. The combination of Franke and Fiechter renders obvious claim limitation [1b-iii].

536. In the combination of Franke and Fiechter, a server application ("transfer control unit"), per Franke, that anticipates when data loss associated with content to be transferred is more likely due non-enablement of network servers ("digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device"), per Fiechter, and prevents the transfer of the content from the local proxy server to the central server as a result ("does not transfer, from the internal storage device to the network storage device, the digital contents"). EX1005 (Franke), [0005], [0078], [0051], [0078]-[0080], [0115]; EX1006 (Fiechter), 17:62-18:3 "wait … until transmission of the data will be more reliable"). A POSITA would have understood that a user selecting Fiechter's option of "wait[ing]… until transmission of the data will be more reliable" would mean that the transfer of the data is not attempted during the period in which there is anticipated high likelihood of data loss.

537. **In sum**, regarding claim limitation [1b-iii], the combination of Franke and Fiechter renders obvious "said transfer control unit [Franke's server application] does not transfer [Fiechter does not transfer and waits for more reliable network], from the internal storage device [Franke's local storage internal to local proxy server] to the network storage device [Franke's central server], the digital contents [Franke's media content] that cannot be recovered [Fiechter's high incidence of data loss] if a network failure [Fiechter's communication network issues or failure] occurs during the transferring [Franke's during posting] of the digital contents [Franke's media data] from the internal storage device [Franke's local storage internal to local proxy server] to the network storage device [Franke's central server]."

538. For similar reasons, the combination of Franke and Fiechter renders obvious corresponding features of claim limitation [7b-iii]. *See* §I above (Table of Claims).

B. Claim [6b-iii]: wherein the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device is transferred after obtaining permission from a user

539. The combination of Franke and Fiechter renders obvious claim limitation [6b-iii].

540. In the combination of Franke and Fiechter, a system, per Franke, anticipates when data loss associated with content to be transferred is more likely

due to non-enablement of network servers ("the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device") and transfers contents from the local proxy server to the central server only after a user is prompted to provide instructions to proceed with the transfer ("is transferred after obtaining permission from a user"). EX1005 (Franke), [0112] ("allowing a user to configure settings for posting or downloading media data"), [0113] ("interface 1110 presents an options window 1150 allowing a user to choose one of the following: 'Ask Me What to Do'"), [0115] ("user interface 1410 is provided if there is an error in communicating with local proxy server 150. For example, window 1415 can display a message such as 'Cannot communicate with Local Proxy Server. Edit your settings and select OK.'"), [0036] ("send status updates to [the] users"), [0103] ("completion signal is generated"), *see also generally* [0108]-[0115]; EX1006 (Fiechter), 17:62-18:3 ("high incidence of data loss due to propagation errors").

541. Put more plainly, in this combination, a system anticipates that there will be a high incidence of data loss, per Franke, and posts media data from the local proxy server to the central server only after providing a user with an option to select (*e.g.*, "Ask me what to do" per Franke, or "asking the user if they would prefer to wait for the information until transmission of the data will be more reliable" per Fiechter). EX1005 (Franke), [0113]; EX1006 (Fiechter), 17:62-18:3

542. **In sum**, regarding claim limitation [6b-iii], the combination of Franke and Fiechter renders obvious "the digital contents [Franke's media data] that cannot be recovered [Fiechter's high incidence of data loss] if a network failure [Fiechter's communication network issues or failure] occurs during the transferring [Franke's during posting] of the digital contents [Franke's media data] from the internal storage device [Franke's local storage internal to local proxy server] to the network storage device [Franke's central server] is transferred [Franke's is posted] after obtaining permission from a user [Fiechter's after the user indicates to proceed]."

543. For similar reasons, the combination of Franke and Fiechter renders obvious corresponding features of claim limitation [12b-iii]. *See* §I above (Table of Claims).

C. Motivations for the Combination Franke and Fiechter

544. A POSITA would have been motivated to combine Franke with Fiechter for at least the following six reasons.

545. **First**, a POSITA would have found it obvious to apply Fiechter's teachings to Franke because a POSITA would have understood that network failures and connectivity disruptions are common occurrences in most, if not all, networks; thus, a system, such as that described by Franke, incorporates techniques for dealing with inevitable interruptions of transfer. *See* §VIII.B.2 above.; *see also* EX1005 (Franke), [0005] ("if the central server is experiencing a high volume of traffic, the

central server will be slow in delivering requested data to LAN users [and low] bandwidth connections to the central server can exacerbate the situation"); EX1006 (Fiechter), 17:50-59.

546. Indeed, both Fiechter and Franke are concerned with a system that includes a device attempting to transfer media data over a network where communication is not always reliable and a receiving device may not be enabled—which a POSITA would have understood to be true of most networks. Fiechter offers additional details for how the system can respond when transfer of data is expected to be unreliable, and a POSITA would have been motivated to implement a system, per Franke, with the teachings of Fiechter in such situations. A POSITA, understanding the common issues raised by network failures, would have been motivated to provide mechanisms for predicting the occurrence of network failure-related issues and dealing with them, as Fiechter teaches.

547. **Second**, a POSITA would have understood the techniques discussed by Franke and Fiechter for dealing with the possibility of data loss are common-sense. Faced with increased risk of loss, either through increased likelihood of a failure or through properties of content that make the content more likely to be lost in the case of a failure, a POSITA would have had few options: either (i) to not transfer the content (or not transfer the content until a later time) or (ii) to decide to proceed with

transferring the content despite the risk. EX1005 (Franke), [0078], [0051], [0078]-[0080], [0115]; EX1006 (Fiechter), 17:50-18:3.

548. Understanding that the second option of proceeding with the transfer risks a potential loss of content, it makes sense to ask a user to authorize the action. A POSITA would have been familiar with such mechanisms, including presenting a user with options for how to proceed. *See e.g.,* EX1007 (Marotta), [0481]-[0482], [0691]-[0710], FIGS. 20, 41. Indeed, a POSITA would have been motivated to implement additional user options to select between waiting or proceeding with the transfer, per Fiechter, in view of Franke's description of providing information to the user. EX1006 (Fiechter), 17:50-18:3.

549. **Third**, determining that data loss is likely or that a transfer is likely to fail, and taking precautions—such as presenting a user with an option of waiting or proceeding with the transfer despite the risks—is a straightforward way to deal with uncertainties about conditions for a transfer, and may reduce the loss of data or the number of transfer attempts required to succeed in a transfer. *See* §§VIII.B.1 above–VIII.B.2 above (discussing networking transfers and transfer attempts); EX1006 (Fiechter), 17:62-18:3. A POSITA would have sought to integrate Fiechter's teachings in a system like Franke's to achieve these advantages.

550. **Fourth**, a POSITA would have known that content loss is frustrating to a user, especially when the content is protected content that may have been

purchased by the user. Thus, systems that predict and prevent data loss associated with network failures help greatly to reduce such frustrations. *See* §VIII.E.1 above; EX1005 (Franke), [0005], [0028], [0032], [0036], [0053]-[0054], [0058], [0115]; EX1006 (Fiechter), 2:9-14, 3:10-14, FIGS. 2, 8-9. Additionally, network failures impact all types of computer systems, including stationary, home-based systems, and mobile systems, like many of the systems Fiechter describes. The techniques for mitigating the effect of potential network failures taught by Fiechter are equally applicable in home networks or mobile networks.

551. **Fifth**, a POSITA would have understood that in the combination, Franke's system continues to function as Franke describes (*see* §XIV.A above), implemented with details related to dealing with anticipated content loss or transfer failure related to non-enabled network servers as Fiechter describes (*see* §X.3 above). A POSITA would have understood that the combination of a system such as that taught by Franke with teachings of Fiechter would result in the system with improved ability to anticipate and address potential loss of content during transfer, because the implementation of software to effect Fiechter's teachings would have had predictable results.

552. Moreover, as described in §§XIV.A.6 and XIV.B.1 above, Franke's system already includes mechanisms for addressing situations where content cannot be transferred or is not available when requested by a user not transmitting the

content or by presenting information, like a completion signal, and options to a user as to how to proceed, and Fiechter's additional details about how to implement the halt in transferring content and what options to provide to a user does not alter the performance of the Franke system. *See e.g.*, EX1005 (Franke), [0036], [0094]-[0096], [0103]-[0108].

553. **Sixth**, the combination of Franke and Fiechter would have been well within a POSITA's skill because mechanisms for monitoring network capabilities and connection, and responding to failures to fully transfer contents by preventing a transfer or providing options to a user were well-known by the Critical Date, as Fiechter teaches. EX1006 (Fiechter), 17:62-18:3; *see also* §VIII.B.2 above. Additionally, implementing software to effect Fiechter's teachings to anticipate network issues that may impact transfer and result in content loss and to address anticipated issues by preventing transfer or asking the user to provide an instruction to proceed would have been well within the level of skill of the POSITA. *See* §VII.A above.

554. Finally, a POSITA would have had a reasonable expectation of success in doing so for the reasons explained in this section, and because methods for responding to increased likelihood of network failures during data transfer were well known to a POSITA by the Critical Date. *See* §VIII.B above.

XVI. GROUND 2C: FRANKE IN VIEW OF ITO RENDERS CLAIMS 1-5 AND 7-11 OBVIOUS

555. Beyond the relevance of Franke's teachings alone, as described in Ground 2A (*see* §XIV.A.6 above), the features recited in claim limitations [1b-iii] and corresponding features recited in claim limitation [7b-iii] also would have been obvious to a POSITA based on the combined teachings of Franke and Ito, for at least the reasons set forth in the Petition which I have read and fully endorse. *See* §I above (Table of Claims). To streamline the analysis in this declaration, I do not repeat all of the reasons stated in the Petition as to why a POSITA would have found claim element [1b-iii], and corresponding features of claim limitation [7b-iii] obvious based on the combination of Franke and Ito, but by way of example and to highlight certain points, I make the following additional observations.

556.

557. The combination of Franke and Ito renders obvious all other elements of claims 1-5 and 7-11 for the reasons described with respect to those elements in Ground 2A (*see* §XIV above).

558. As I discuss in §XIV.A.6 above Franke describes that there are circumstances where one or both of the local proxy server and central server may be inaccessible on the network or not enabled, preventing transfer of content to or from the server and/or prompting a user for instructions. *See* EX1005 (Franke), [0009] ("media data is stored on the local proxy server and stored on the central server by

the local proxy server if the local proxy server is enabled."), [0036] ("if requested project data has not been completely stored or posted central server 110, central server 110 can send status updates to the users or devices connected via WAN 160"), [0051] ("if enabled, local proxy server 150 stores project data … on central server 110"), *see also generally* [0078]-[0080], [0094]-[0096], [0103]-[0109], [0115].

559. Franke briefly provides techniques for dealing with non-enabled servers in the network system that would render content "unrecoverable" in that it would not be able to be successfully transferred to its destination. Ito, however, provides additional details about the implementation of these techniques in a system like Franke's. EX1012 (Ito), [0002] (copy control information "indicates that copying is allowed only once (Copy Once)" and as a result, "indicates that backup is prohibited, [such that] the transmission is terminated"), [0008] (describing that in some cases, content is lost in some cases, content is lost "due to mishandling or failure" including "a failure of the recording and playback device or the network to which the recording and playback device is connected"); *see also* [0001]-[0008], [0252]-[0253].

560. Accordingly, in the combination, in a system, such as that described by Franke, when high risk of data loss is detected due to the non-enablement of one or more servers or storage devices on the network, the system anticipates that "backup is prohibited, [and] the transmission is terminated." *Id.*, [0252]; *see also id.*, [0252]

("If the backup process indicates permission to generate backup information corresponding to the encrypted content to be backed up, the encrypted content is sent to the backup device"). Put more plainly, in the combination, a system such as Franke's can detect when non-enablement of servers on the network that may compromise transfers over the network is likely; the system can then decide to not transfer content to backup on the central server. *Id.*, [0252].

A. Claim [1b-iii]: wherein said transfer control unit does not transfer, from the internal storage device to the network storage device, the digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device

561. The combination of Franke-Ito renders obvious claim limitation [1b-iii].

562. Franke describes a system that monitors a connection status of the proxy and central servers and the transfer status of content being stored in the central server by the proxy server, preventing transfer of contents when the servers are not enabled to receive content. *See* EX1005 (Franke), [0009] ("media data is stored on the local proxy server and stored on the central server by the local proxy server if the local proxy server is enabled."), [0036] ("if requested project data has not been completely stored or posted central server 110, central server 110 can send status updates to the users or devices connected via WAN 160"), [0051] ("if enabled, local proxy server 150 stores project data … on central server 110").

563. In combination with Ito, where content cannot be backed up to the central server to guard against network failure ("digital contents that cannot be recovered if a network failure occurs during the transferring of the digital contents from the internal storage device to the network storage device") the content transfer to the backup is terminated ("said transfer control unit does not transfer, from the internal storage device to the network storage device"). *See* §XVI above.

564. For example, in the Franke-Ito combination, where network connection between the proxy server and central server is not available (or content is otherwise not able to be transferred), the Franke-Ito system does not transfer content from the proxy server to the central server so as to prevent loss of the content. EX1005 (Franke), [0005], [0036], [0051], [0078]-[0080], [0094]-[0096], [0103]-[0109], [0112]-[0115]; EX1012 (Ito), [0001]-[0008], [0252]-[0253].

565. **In sum**, regarding claim limitation [1b-iii], the combination of Franke and Ito renders obvious "said transfer control unit [Franke's proxy server application] does not transfer [Ito terminates the transfer], from the internal storage device [Franke's local storage internal to local proxy server] to the network storage device [Franke's central server], the digital contents [Franke's media data] that cannot be recovered [subject to "Copy Once" protections in Ito] if a network failure occurs [Franke's non-enabled server] during the transferring [Franke's during posting] of the digital contents [Franke's media data from the internal storage device

[Franke's local storage internal to local proxy server] to the network storage device [Franke's central server]."

566. For similar reasons, the combination of Franke and Ito renders obvious corresponding features of claim limitation [7b-iii]. See §I above (Table of Claims).

B. Motivation for the Combination Franke and Ito

567. A POSITA would have been motivated to combine Franke with Ito for at least the following five reasons.

568. **First**, a POSITA would have been motivated to implement a system, such as Franke describes, with the additional details that Ito teaches related to the termination of transfer of content that cannot be successfully backed up in order to prevent data loss and based on similarities in Franke and Ito's discussion of network connectivity issues. EX1012 (Ito), [0001]-[0008], [0252]-[0253]; EX1005 (Franke), [0008], [0030].

569. **Second**, a POSITA would have understood that when data cannot be successfully transferred to storage (for example, copy control information indicates copy only once or the storage device is not available on the network), not completing the transfer is common sense. If the protected content will be compromised or lost to the user as a result of the transfer, the logical response is not to transfer the content from where it is stored. For example, Ito's use of copy control information to inform transfer processes attempts to prevent loss of license or rights to the content that may

occur if the system fails to comply with copy control information. Additionally, preventing transfer of protected content may minimize risk of loss of data during a network failure while the transfer is underway. A POSITA would have been motivated to implement a policy of not transferring content likely to be lost if transfer is attempted, per Ito, to prevent content loss. *See* §VIII.B.2 above; §VIII.E.1 above; *see also* EX1005 (Franke), [0078], [0051], [0078]-[0080], [0115]; EX1012 (Ito), [0001]-[0008], [0252].

570. **Third**, a POSITA would have understood that content loss is frustrating to a user, and systems that predict and prevent data loss associated with network failures help greatly to reduce such frustrations. *See* §VIII.E.1 above; EX1005 (Franke), [Abstract], [0028], [0094]; EX1012 (Ito), [0252]-[0253].

571. **Fourth**, a POSITA would have understood that in the combination of Franke and Ito, Franke's system continues to function as Franke describes (*see* §XIV.A above), implemented with details related to predicting and dealing with anticipated content loss related to transfer of content as Ito describes (*see* §X.4 above). A POSITA would have understood that the combination of a system such as that taught by Franke with teachings of Ito would result in the system with improved ability to anticipate potential loss of content during transfer, because the implementation of software to effect Ito's teachings—for example to check the copy-control information associated with content prior to initiating transfer or

determine the likelihood that the server is not enabled or otherwise unavailable—would have had predictable results without altering performance of Franke's system.

572. **Fifth**, the combination of Franke and Ito would have been well within a POSITA's skill because mechanisms for anticipating content loss during transfer were well-known by the Critical Date, as Ito teaches. EX1012 (Ito), [0001]-[0008], [0252]-[0253]; *see also* §VIII.B above. Additionally, implementing software to effect Ito's teachings to anticipate issues related to content transfer to mitigate content loss by preventing transfer would have been well within the level of skill of the POSITA. *See* §VII above.

573. Finally, a POSITA would have had a reasonable expectation of success in making the combination for the reasons explained in this section, and because methods for anticipating and preventing data loss during content transfer were well known to a POSITA by the Critical Date. *See* §VIII.B above.

XVII. ADDITIONAL REMARKS/CONCLUSION

574. I currently hold the opinions set expressed in this declaration. But my analysis may continue, and I may acquire additional information and/or attain supplemental insights that may result in added observations.

APPENDIX A

EREZ ZADOK, CURRICULUM VITAE

Erez Zadok
Computer Science Department
349 New Computer Science
Stony Brook University
Stony Brook, NY 11794-2424

Phone: +1 631 632 8461 (office)
Fax: +1 631 632 8243
Timezone: US/Eastern
Email: ezk@cs.stonybrook.edu
Web: https://www.cs.sunysb.edu/~ezk

RESEARCH INTERESTS

Operating systems with a special focus on file systems, storage, clouds, big data, hardware/architecture, encryption, security, benchmarking, performance analysis and optimization, energy efficiency, and system administration.

EDUCATION

May 2001 Ph.D., Computer Science, Columbia University, New York, NY,
FiST: A System for Stackable File-System Code Generation.

Sep 1997 M.Phil., Computer Science, Columbia University, New York, NY

Oct 1994 M.S., Computer Science, Columbia University, New York, NY
Discovery and Hot Replacement of Replicated Read-Only File Systems, with Application to Mobile Computing

May 1991 B.S., Computer Science, Columbia University, New York, NY

May 1982 Certified Technician, Electrical Engineering, Holtz College, Israel

PROFESSIONAL EXPERIENCE

Aug 2024–present	Graduate Program Director, Computer Science Department, Stony Brook University
Jul 2024–present	Senator, University Faculty Senate, representative of the Computer Science Department, Stony Brook University
Aug 2017–Jul 2024	Graduate Academic Adviser, Computer Science Department, Stony Brook University
Jan 2016–present	Professor, Computer Science Department, Stony Brook University
Jan 2007–2015	Associate Professor, Computer Science Department, Stony Brook University
Jan 2001–Jan 2007	Assistant Professor, Computer Science Department, Stony Brook University
2013–present	Director, Smart Energy Technologies (SET) Faculty Cluster, Stony Brook University
2025–present	President; Zadok Consulting LI Inc., DBA "Zadoks Consulting Services"
2013–present	Managing Member; Zadoks Consulting, LLC.
2009–2010	Consultant; CTERA Networks, Inc.
2009–2019	Consultant; Packet General Networks, Inc.
1991–2000	Graduate Research Assistant, Computer Science Department, Columbia University
1999–2000	Director of Software Development, HydraWEB Technologies, Inc.
1994–1998	Project Leader, HydraWEB Technologies, Inc.

1990–1998	Consultant, SOS Corporation
1997	Manager of Computing Facilities, Computer Science Department, Columbia University
1991–1998	Technical Staff Member, Computer Science Department, Columbia University
1989–1991	Assistant Lab Manager, Academic Information Systems, Columbia University
1987–1989	Student Consultant, Academic Information Systems, Columbia University
1984–1986	National Army Service, Israeli Air Force, Israel
1982–1984	Programmer, Commodore Israel, Tel-Aviv, Israel
1981–1984	Computer Lab Manager, Holtz College, Tel-Aviv, Israel

PERSONAL

Born December 4, 1964, Tel-Aviv, Israel.

Married, one child.

Citizenships: U.S.A and Israel

Fluent in English and Hebrew

Member: ACM, ACM SIGOPS, IEEE, IEEE Computer Society, USENIX

Affiliate: Storage Systems Research Center (SSRC), Jack Baskin School of Engineering, University of California, Santa Cruz, California.

Member: The I/O Traces, Tools and Analysis (IOTTA) Technical Work Group (TWG), part of the the Storage Networking Industry Association (SNIA).

FUNDING

Oct 2022–present	*CNS Core: Large: Systems and Verifiable Metrics for Sustainable Data Centers*. NSF. $1,500,000 (SBU share $928,403), 4 years. Co-PI with four other SBU faculty, collaborative with Binghamton University and Penn State University.
Oct 2022–present	*Collaborative Research: CyberTraining: Implementation: Medium: FOUNT: Scaffolded, Hands-On Learning for a Data-Centric Future*. NSF. $996,548 (SBU share $174,905), 3 years. Sole PI at SBU. Collaborative with U. Chicago (lead), NYU, Northern Illinois U., and UC San Diego (6 collaborators total).
2022	*CNS Core: Medium: Optimizing Storage Caches via Adaptive and Reconfigurable Tiering*. NSF Research Experiences for Undergraduates (REU) supplement. $16,000. Lead-PI with one other SBU faculty, collaborative with Emory University.
2022	*SCC-IRG Track 1: Smart Aging: Connecting Communities Using Low-Cost and Secure Sensing Technologies*. NSF Research Experiences for Undergraduates (REU) supplement. $24,000. Co-PI with four other SBU faculty.
2022	*CNS Core: Medium: Optimizing Storage Caches via Adaptive and Reconfigurable Tiering*. NSF Research Experiences for Undergraduates (REU) supplement. $16,000. Lead-PI with one other SBU faculty, collaborative with Emory University.

2022	*CNS Core: III: Medium: Collaborative Research: Optimizing and Understanding Large Parameter Spaces in Storage Systems*. NSF Research Experiences for Undergraduates (REU) supplement. $16,000. Lead PI with one other.
Oct 2021–present	*CNS Core: Medium: Optimizing Storage Caches via Adaptive and Reconfigurable Tiering*. NSF. $800,000 (SBU share $533,333), 3 years. Lead-PI with one other SBU faculty, collaborative with Emory University.
Oct 2021–present	*CNS Core: Medium: Secure, Reliable, and Efficient Long-Term Storage*. NSF. $1,198,842 (SBU share $717,303), 4 years. Lead-PI with one other SBU faculty, collaborative with UC Santa Cruz.
2021–2022	*Kubernetes/Containerizations on GPU Clusters for AI*. SUNY-IBM AI Collaborative Research Alliance. $100,000. Co-PI with two others.
2020–2021	*Automated Cross-Validation of TLS 1.3 Implementations*. Facebook Faculty Research Award. $50,000. Lead PI with two others.
Oct 2020–present	*SCC-IRG Track 1: Smart Aging: Connecting Communities Using Low-Cost and Secure Sensing Technologies*. NSF Smart and Connected Communities. $1,700,126, 4 years. Co-PI with four other SBU faculty.
Sep 2020–present	*Recruitment and Retention of Women Undergraduates*. Center for Inclusive Computing Diversity Initiative, by Northeastern University and Pivotal Ventures. $295,742, 4 years. Co-PI with four other SBU faculty.
2020	*FMitF: Track I: NLP-Assisted Formal Verification of the NFS Distributed File System Protocol*. NSF Research Experiences for Undergraduates (REU) supplement. $16,000. Lead PI with two others.
2020	*CNS Core: III: Medium: Collaborative Research: Optimizing and Understanding Large Parameter Spaces in Storage Systems*. NSF Research Experiences for Undergraduates (REU) supplement. $8,000. Lead PI with one other.
Oct 2019–present	*CNS Core: III: Medium: Collaborative Research: Optimizing and Understanding Large Parameter Spaces in Storage Systems* NSF. $1,088,017 (SBU share $823,142), 4 years. Lead-PI with one other SBU faculty, collaborative with Harvey Mudd College.
Oct 2019–present	*FMitF: Track I: NLP-Assisted Formal Verification of the NFS Distributed File System Protocol* NSF. $748,300, 3 years. Lead-PI with two other SBU faculty.
2019	*CI-SUSTAIN: National File System Trace Repository*. NSF Research Experiences for Undergraduates (REU) supplement. $8,000. Sole PI.
2019	*Study of a Novel Non-Wearable Respiration and Heart Rate Sensor in Cardiopulmonary Exercise Testing*. Stony Brook College of Engineering and Applied Sciences, SEED grant. $15,000, Co-PI with five others.
2019	*Realizing the Full Performance and Parallelization Potential of Modern Storage Architectures for Big Data Applications*. Stony Brook Research Foundation, SEED grant. $60,000, Co-PI with Anshul Gandhi.
2018	*Storage/Deduplication research*. Dell-EMC Corporation. $25,000, Single PI.

Jun 2017–present	*CI-SUSTAIN: National File System Trace Repository*. NSF. $129,867 (SBU share), 3 years. Co-PI with lead institution Harvey Mudd College.
2017–2022	*I/UCRC Phase II: Center for Visual and Decision Informatics (CVDI) Site at SUNY Stony Brook*. NSF. $400,000, 4 years. Co-PI with A. Kaufman, K. Mueller, H. Schwartz, and D. Samaras.
2017	Dell-EMC Corporation. *Storage/Deduplication research*, $25,000, Single PI.
2016–2021	*NRT-DESE: Interdisciplinary Graduate Training to Understand and Inform Decision Processes Using Advanced Spatial Data Analysis and Visualization (STRIDE)*. NSF. $2,993,930, 5 years. Senior personnel.
2016–2019	*Early Detection of User-impersonating Attackers using Multilayer Tripwires*, U.S. Office of Naval Research (ONR). $586,215, 3 years. Co-PI with Nick Nikiforakis.
Jun 2016–2019	*EAGER: Elastic Multi-layer Memcached Tiers* NSF. $257,165, 2 years. Co-PI with Anshul Gandhi.
2016	EMC Corporation. *Storage/Deduplication research*, $25,000, Single PI.
2015	EMC Corporation. *Storage/Deduplication research*, $25,000, Single PI.
Feb 2015	*Student Travel Support for the 13th USENIX Conference on File and Storage Technologies (FAST 2015)*. NSF. $20,000, 1 year. Sole PI.
Sep 2014–2017	*Adaptive Runtime Verification and Recovery for Mission-Critical Software*. U.S. Air Force Office of Scientific Research (AFOSR). $620,861, 3 years. Co-PI with Scott A. Smolka and Scott D. Stoller. (Collaboration with NASA JPL.)
Jan 2014–2016	*Smarter Electric Grid Research, Innovation, Development, Demonstration, Deployment Center (SGRID3)*. Brookhaven Science Associates LLC (BNL), $236,397, 1 year. Lead-PI with one other Stony Brook Co-PI.
2014	EMC Corporation. *Storage/Deduplication research*, $25,000, Sole PI.
Jun 2013–2017	*CSR: Medium: Collaborative Research: Workload-Aware Storage Architectures for Optimal Performance and Energy Efficiency*. NSF. $513,900 (SBU share, total budget $1,000,000), 3 years. Lead-PI with one other Stony Brook Co-PI, and two more institutions (Harvard U. and Harvey Mudd College).
Jan 2013–2017	*BIGDATA: Small: DCM: Collaborative Research: An efficient, versatile, scalable, and portable storage system for scientific data containers*. NSF. $444,267 (SBU share, total budget $746,290), 3 years. Lead-PI with two other Stony Brook Co-PIs, and two more institutions (Brandeis U. and Louisiana State U.).
Sep 2013–2017	*CRI-CI-ADDO-EN: National File System Trace Repository*. NSF. $37,018 (SBU share, total budget $167,817), 3 years. Co-PI with lead institution Harvey Mudd College.
2013	Western Digital Research award. *Shingled Magnetic Recording Disks Benchmarking*, $50,000, Single PI.
Sep 2012 – 2016	*NFS4Sec: An Extensible Security Layer for Network Storage*. NSF. $486,783, 3 years. Lead-PI with one other Co-PI.

2012-2013	*Server-Class Performance vs. Energy Optimizations*. Government of Israel (GoI), Mission to the USA. $47,152, 1 year. Lead PI with one other Co-PI.
2011	NetApp Research award. *Dedup Workload Modeling, Synthetic Datasets, and Scalable Benchmarking*, $40,000, Single PI.
2010	NetApp Research award. *A Study of Network Storage Benefits using FLASH Hardware with Indexing Workloads*, $40,000, Single PI.
Nov 2010 – 2016	*Long Island Smart Energy Corridor*. Department of Energy (DOE), LIPA, and New York State. Collaboration between Stony Brook University, SUNY Farmingdale, and LIPA. $2,822,638, Co-PI.
Sep 2009 – Aug 2013	*Collaborative Research: Performance- and Energy-Aware HEC Storage Stacks*. NSF. $652,000, 3 years. Co-PI with Geoff Kuenning (Harvey Mudd College)
Sep 2009 – Aug 2013	*Collaborative Research: Secure Provenance in High End Computing Systems*. NSF. $564,972, 3 years. Co-PI with Radu Sion. Collaborative project with Patrick McDaniel (Penn State U.) and Marianne Winslett (UIUC).
Apr 2009 – Nov 2012	*Survivable Software*. U.S. Air Force Office of Scientific Research (AFOSR). $881,691, 39 months. Co-PI with Scott A. Smolka, Radu Grosu, Scott D. Stoller, and Klaus Havelund (NASA JPL).
Feb 2010	*Student Travel Support for the First USENIX Workshop on Sustainable Information Technology (SustainIT 2010)*. NSF. $10,000, 1 year. Lead PI.
2009	Network Appliance Research award. *Power use in Storage Servers.* $30,000 Single PI.
2008	*The Impact of Storage Software and Aging on Power Consumption*, **IBM Faculty award** (IBM T.J. Watson Labs). $20,000, one year. Single PI.
2008	Network Appliance Equipment gift. *A Study of User File Access Patterns.* $91,083 Single PI.
Sep 2007 – 2015	*Center for Information Protection: A Multi-University Industry/University Collaborative Research Center*. NSF. $250,147, 5 years. Co-PI with R. Sekar (PI), Tzi-Cker Chiueh, Scott Stoller, and Radu Sion.
Sep 2006 – Aug 2009	*CT-ISG: N3S: Networked Secure Searchable Storage with Privacy and Correctness Assurances*. NSF. $300,000, 3 years. Co-PI with Radu Sion.
Aug 2006 – Aug 2010	*File System Tracing, Replaying, Profiling, and Analysis on HEC Systems*. NSF. $760,252, 3 years. Lead PI with Klaus Mueller (Stony Brook) and Ethan Miller (UC Santa Cruz).
Jul 2006	*End-to-End File Server Security*, **IBM Faculty award** (IBM Haifa Research Labs). $20,000, one year. Single PI.
Jun 2006 – Aug 2010	*CSR—PDOS: Support for Atomic Sequences of File System Operations*. NSF. $561,727, 3 years. Lead PI with Margo Seltzer (Harvard University).
Jan 2006 – Dec 2006	*Secure File Systems*, NY State "Millennium" award, $204,528, one year. Co-PI with R. Sekar (PI), Tzi-Cker Chiueh, CR Ramakrishnan, Radu Sion, and Scott D. Stoller.

Jul 2005 – Aug 2010	*CSR—AES: Runtime-Monitoring and Model Checking for High-Confidence Systems Software*. NSF. $830,000, 4 years. Lead PI with Radu Grosu, Y. Annie Liu, Scott Smolka, and Scott D. Stoller.
Sep 2005 – Aug 2004	*I/UCRC: A Plan for Developing a Multi-University Industry/University Collaborative Research Center on Cyber Security*. NSF. $9,987, one year. Co-PI with R. Sekar, Radu Sion Scott D. Stoller, and Tzi-Cker Chiueh.
Sep 2004 – Aug 2009	*Federal Cyber Service: Scholarship for Service (SFS)*. NSF. $2,459,061, 4 years. Co-PI with R. Sekar, Scott D. Stoller, I. V. Ramakrishnan, and Tzi-Cker Chiueh.
Sep 2003 – Aug 2008	*A Layered Approach to Securing Network File Systems*. NSF Trusted Computing Program (TC). $400,000, 3 years. Single PI.
Sep 2003 – Aug 2005	*Collaborative Research: Capacity Expansion in Information Assurance*. NSF Collaborative Research Proposal (CAP). $199,883, 2 years. Co-PI with R. Sekar, Scott D. Stoller, and I. V. Ramakrishnan.
Sep 2002 – Aug 2007	**CAREER**: *An In-Kernel Runtime Execution Environment for User-Level Programs.* NSF Next Generation Software Program (NGS). $400,000, 5 years. Single PI.
Jan 2003 – Dec 2003	*Assessing the Technological Basis for Enterprise Protection*, NIJ (CyberScience Lab) CSL. $90,563, one year. Co-PI with R. Sekar, Tzi-cker Chiueh, and Scott D. Stoller.
Sep 2002 – May 2003	New York State Strategic Partnership for Industrial Resurgence (SPIR), with Packet General Networks. *S3: Secure Shared Storage.* $94,581. Single PI.
May 2002 – Aug 2002	New York State Strategic Partnership for Industrial Resurgence (SPIR), with Packet General Networks. *A Secure and Scalable Network Appliance.* $55,676. Single PI.
Apr 2003	Microsoft Tablet PC Seed Award. $4,169. Single PI.
2002	HP/Intel IA-64/IPF Second Generation Equipment gift. *Linux Application Performance and File System Security.* $131,529. Single PI.
2001	HP/Intel IA-64/IPF Equipment gift. *Linux Network Scalability and File System Reliability.* $22,490. Single PI.
2001	Red Hat University software award.

AWARDS AND HONORS

2023	Best paper award. A. Merenstein, V. Tarasov, A. Anwar, S. Guthridge, and E. Zadok. F3: Serving Files Efficiently in Serverless Computing. In Proceedings of the 16th ACM International Systems and Storage Conference (SYSTOR '23).
2023	Elected as IEEE Senior Member
2021–2022	Chancellor's Award for Excellence in Scholarship and Creative Activities, State University of New York (SUNY).
2022	Provost's Excellent Mentor award, Stony Brook University.

2021 Elected as ACM Distinguished Member for "Outstanding Scientific Contributions to Computing."

2021 Dean's "Millionaire" award (given lead PIs who secure more than one-million dollars of new funding in a given year), College of Engineering and Applied Sciences, Stony Brook University.

2020 Inducted as member of the Stony Brook University Chapter of the National Academy of Inventors (NAI).

2020 Best paper award. M. Wajahat, A. Yele, T. Estro, A. Gandhi, and E. Zadok. Distribution fitting and performance modeling for storage traces. In 27th IEEE International Symposium on the Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2019).

2019 Dean's "Millionaire" award (given lead PIs who secure more than one-million dollars of new funding in a given year), College of Engineering and Applied Sciences, Stony Brook University.

Sep 2011 Best Paper Award. S. D. Stoller and E. Bartocci and J. Seyster and R. Grosu and K. Havelund and S. A. Smolka and E. Zadok. Runtime verification with state estimation. In proceedings of the 2nd International Conference on Runtime Verification (RV'11).

Dec 2009 LISTnet's "Top 20 techies of Long Island" award.

Aug 2008 Service Award (for 2006–2008), Stony Brook University, Computer Science Department

2008 IBM Faculty award (IBM T.J. Watson Labs)

2007–2008 Chancellor's Award for Excellence in Teaching, State University of New York (SUNY).

2007–2008 President's Award for Excellence in Teaching, Stony Brook University.

Aug 2006 Research Excellence Award (for 2005–2006), Stony Brook University, Computer Science Department

Jul 2006 IBM Faculty award (IBM Haifa Research Labs)

Nov 2005 Best Short Paper Award. N. Joukov, A. Kashyap, G. Sivathanu, E. Zadok. Kefence: An Electric Fence for Kernel Buffers. In proceedings of the first ACM International Workshop on Storage Security and Survivability (StorageSS 2005), "The Paradigm Shift to Info-Centric Protection," held in conjunction with the 12th ACM Conference on Computer and Communications Security (CCS 2005).

May 2005 Best Paper Award. N. Joukov, A. Rai, and E. Zadok. Increasing Distributed Storage Survivability with a Stackable RAID-like File System. In proceedings of the 2005 IEEE/ACM Workshop on Cluster Security, in conjunction with the Fifth IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid).

Aug 2004 Graduate Teaching and Research Award (for 2003–2004), Stony Brook University, Computer Science Department

May 2003 Promising Inventor Award, The State University of New York, Research Foundation

Sep 2002 NSF CAREER award, National Science Foundation

Jun 2001 Best Student Paper. M. G. Schultz, E. Eskin, E. Zadok, M. Bhattacharyya, and S. J. Stolfo. MEF: Malicious Email Filter — A UNIX Mail Filter that Detects Malicious Windows Executables. In *Proceedings of the Annual USENIX Technical Conference, FreeNIX Track*, pages 245–252, JUNE 2001.

Jun 2000 Usenix Student Stipend award

Jun 1999 Usenix Student Stipend award

1991 Dean's List

1991 NCR Best Undergraduate Research Award

1989–1990 Gussman Scholar

1988–1989 National Dean's List

1988–1989 Scheuer Foundation Presidential Scholar

1988 Dean's List

PUBLICATIONS

Books

E. Zadok. *Linux NFS and Automounter Administration*. Sybex, Inc., May 2001.

Journal and Magazine Articles

Mário Antunes, Tyler Estro, Pranav Bhandari, Anshul Gandhi, Geoff Kuenning, Yifei Liu, Carl Waldspurger, Avani Wildani, and Erez Zadok. Kneeliverse: A universal knee-detection library for performance curves. *SoftwareX*, 30:102161, 2025.

Tyler Estro, Mário Antunes, Pranav Bhandari, Anshul Gandhi, Geoff Kuenning, Yifei Liu, Carl Waldspurger, Avani Wildani, and Erez Zadok. Accelerating multi-tier storage cache simulations using knee detection. *Performance Evaluation*, 164:102410, May 2024.

S. Hussain, P. McDaniel, A. Gandhi, K. Ghose, K. Gopalan, D. Lee, Y. Liu, Z. Liu, S. Mu, and E. Zadok. Verifiable sustainability in data centers. *IEEE Security & Privacy*, (01):2–15, March 2024.

Anshul Gandhi, Dongyoon Lee, Zhenhua Liu, Shuai Mu, Erez Zadok, Kanad Ghose, Kartik Gopalan, Yu David Liu, Syed Rafiul Hussain, and Patrick Mcdaniel. Metrics for sustainability in data centers. *SIGENERGY Energy Inform. Rev.*, 3(3):40–46, oct 2023.

Ibrahim "Umit" Akgun, Ali Selman Aydin, Andrew Burford, Michael McNeill, Michael Arkhangelskiy, and Erez Zadok. Improving storage systems using machine learning. *ACM Transactions on Storage (TOS)*, 19(1):1–30, Jan 2023.

Erez Zadok and Ada Gavrilovska. Running virtual pc (cpc) meetings. *;login: The USENIX Magazine*, 45(2):54–56, July 2020.

Muhammad Wajahat, Aditya Yele, Tyler Estro, Anshul Gandhi, and Erez Zadok. Analyzing the distribution fit for storage workload and internet traffic traces. *Performance Evaluation*, pages 102–121, 2020.

George Amvrosiadis, Ali R. Butt, Vasily Tarasov, Erez Zadok, and Ming Zhao. Selected results of the workshop on data storage research 2025. *;login: The USENIX Magazine*, 44(4):24–28, December 2019.

Bharath Kumar Reddy Vangoor, Prafful Agarwal, Manu Mathew, Arun Ramachandran, Swaminathan Sivaraman, Vasily Tarasov, and Erez Zadok. Performance and resource utilization of FUSE user-space file systems. *ACM Transactions on Storage (TOS)*, 15(2), May 2019.

Zhen "Jason" Sun, Geoff Kuenning, Sonam Mandal, Philip Shilane, Vasily Tarasov, Nong Xiao, and Erez Zadok. Cluster and single-node analysis of long-term deduplication patterns. *ACM Transactions on Storage (TOS)*, 14(2), May 2018.

Ming Chen, Geetika Bangera, Dean Hildebrand Farhaan Jalia, Geoff Kuenning, Henry Nelson, and Erez Zadok. vNFS: Maximizing NFS performance with compounds and vectorized I/O. *ACM Transactions on Storage (TOS)*, 13(7), September 2017.

Vasily Tarasov, Erez Zadok, and Spencer Shepler. Filebench: A flexible framework for file system benchmarking. *;login: The USENIX Magazine*, 41(1):6–12, March 2016.

Z. Li, M. Chen, A. Mukker, and E. Zadok. On the trade-offs among performance, energy, and endurance in a versatile hybrid drive. *ACM Transactions on Storage (TOS)*, 11(3), July 2015.

Ming Chen, Dean Hildebrand, Geoff Kuenning, Soujanya Shankaranarayana, Bharat Singh, and Erez Zadok. Is NFSv4.1 ready for prime time? *;login: The USENIX Magazine*, 40(3):6–12, June 2015.

Zhiyuan Zhang, Kevin T. McDonnell, Erez Zadok, and Klaus Mueller. Visual correlation analysis of numerical and categorical data on the correlation map. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 21(2), February 2015.

Michael A. Bender, Martin Farach-Colton, Rob Johnson, Russell Kraner, Bradley C. Kuszmaul, Dzejla Medjedovic, Pablo Montes, Pradeep Shetty, Richard P. Spillane, and Erez Zadok. Don't thrash: How to cache your hash on flash. 5(11):1627–1637, August 2012.

J. Seyster, K. Dixit, X. Huang, R. Grosu, K. Havelund, S. A. Smolka, S. D. Stoller, and E. Zadok. InterAspect: Aspect-oriented instrumentation with GCC. *Formal Methods in System Design*, August 2012.

X. Huang, J. Seyster, S. Callanan, K. Dixit, R. Grosu, S. A. Smolka, S. D. Stoller, and E. Zadok. Software monitoring with controllable overhead. *International Journal on Software Tools for Technology Transfer (STTT)*, 14(3):327–347, 2012.

Priya Sehgal, Vasily Tarasov, and Erez Zadok. Optimizing energy and performance for server-class file system workloads. *ACM Transactions on Storage (TOS)*, 6(3), September 2010.

E. Zadok, V. Tarasov, and P. Sehgal. The case for specialized file systems, or, fighting file system obesity. *;login: The USENIX Magazine*, 35(1):38–40, February 2010.

M. T. Dougherty, M. J. Folk, E. Zadok, H. J. Bernstein, F. C. Bernstein, K. W. Eliceiri, W. Benger, and C. Best. Unifying biological image formats with hdf5. *Communications of the ACM (CACM)*, 52(10):42–47, October 2009.

A. Traeger and E. Zadok. Notes on a nine year study of file system and storage benchmarking. *Byte and Switch*, July 2009. www.byteandswitch.com/storage/storage-management/notes-on-a-nine-year-study-of-file-system-and-storage-benchmarking.php.

A. Traeger, E. Zadok, E. L. Miller, and D. D. E. Long. Findings from the first annual storage and file systems benchmarking workshop. *;login: The USENIX Magazine*, 33(5):113–117, October 2008.

Avishay Traeger, Erez Zadok, Nikolai Joukov, and Charles P. Wright. A nine year study of file system and storage benchmarking. *ACM Transactions on Storage (TOS)*, 4(2):25–80, May 2008.

C. P. Wright, R. Spillane, G. Sivathanu, and E. Zadok. Extending ACID semantics to the file system. *ACM Transactions on Storage (TOS)*, 3(2):1–42, June 2007.

Erez Zadok, Rakesh Iyer, Nikolai Joukov, Gopalan Sivathanu, and Charles P. Wright. On incremental file system development. *ACM Transactions on Storage (TOS)*, 2(2):161–196, 2006.

C. P. Wright, J. Dave, P. Gupta, H. Krishnan, D. P. Quigley, E. Zadok, and M. N. Zubair. Versatility and unix semantics in namespace unification. *ACM Transactions on Storage (TOS)*, 2(1):1–32, February 2006.

C. P. Wright and E. Zadok. Unionfs: Bringing file systems together. *Linux Journal*, 2004(128):24–29, December 2004.

E. Zadok. Using the Amd automounter. *Linux Journal*, 10(114):52–55, October 2003.

E. Zadok. Writing stackable file systems. *Linux Journal*, 05(109):22–25, May 2003.

W. Lee, W. Fan, M. Miller, S. Stolfo, and E. Zadok. Toward cost-sensitive modeling for intrusion detection and response. *Journal of Computer Security*, 10(1–2):5–22, January 2002.

Refereed Conference and Workshop Papers

Dongsheng Luo, Raju Rangaswami, Amir Rahmati, and Erez Zadok. GitTemporalAI: Leveraging temporal knowledge graphs and LLMs for multi-agent repository intelligence. 2025. The First MARW: Multi-Agent AI in the Real World Workshop at AAAI 2025.

Mohammad Saqib Hasan, Sayontan Ghosh, Dhruv Verma, Geoff Kuenning, Erez Zadok, Scott Smolka, and Niranjan Balasubramanian. Handling open-vocabulary constructs in formalizing specifications: Retrieval augmented parsing with expert knowledge. In *Proceedings of the First Conference on Language Modeling (COLM 2024)*, Philadelphia, PA, October 2024.

Christopher Smith, Maliha Tabassum, Soumya Chowdary Daruru, Gaurav Kulhare, Arvin Wang, Ethan Miller, and Erez Zadok. Secure archival is hard... really hard. In *Proceedings of the 16th ACM Workshop on Hot Topics in Storage and File Systems (HotStorage '24)*, Santa Clara, CA, July 2024. ACM.

Alex Merenstein, Xinran Wang, Vasily Tarasov, Prajjawal Agarwal, Scott Guthridge, Kapil Thakkar, Katherine Wu, Ali Anwar, and Erez Zadok. Balancing costs and durability for serverless data. In *Proceedings of the 38th International IEEE Symposium on Mass Storage Systems and Technologies (MSST '24)*, Santa Clara, California, June 2024. IEEE.

Debajyoti Halder, Manas Acharya, Aniket Malsane, Anshul Gandhi, and Erez Zadok. Empirical evaluation of ML models for per-job power prediction. In *Proceedings of the 7th ICPE Workshop on Hot Topics in Cloud Computing Performance (HotCloudPerf 2024)*, pages 181–188, London, UK, May 2024. ACM.

Yifei Liu, Manish Adkar, Gerard Holzmann, Geoff Kuenning, Pei Liu, Scott Smolka, Wei Su, and Erez Zadok. Metis: File system model checking via versatile input and state exploration. In *Proceedings of the 22nd USENIX Conference on File and Storage Technologies (FAST '24)*, pages 123–140, Santa Clara, CA, February 2024. USENIX Association. Received all 3 Artifact-Evaluation badges.

Tyler Estro, Mário Antunes, Pranav Bhandari, Anshul Gandhi, Geoff Kuenning, Yifei Liu, Carl Waldspurger, Avani Wildani, and Erez Zadok. Guiding simulations of multi-tier storage caches using knee detection. In *31st Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS '23)*, Stony Brook, NY, October 2023. IEEE Computer Society.

Peter Desnoyers, Andy Rudoff, Ian Adams, Tyler Estro, Anshul Gandhi, Geoff Kuenning, Mike Mesnier, Carl Waldspurger, Avani Wildani, and Erez Zadok. Persistent memory research in the

post-optane era. In *1st Workshop on Disruptive Memory Systems (DIMES 2023)*, October 2023. https://dl.acm.org/doi/10.1145/3609308.3625268.

Yifei Liu, Gautam Ahuja, Geoff Kuenning, Scott Smolka, and Erez Zadok. Input and output coverage needed in file system testing. In *Proceedings of the 15th ACM Workshop on Hot Topics in Storage and File Systems (HotStorage '23)*, Boston, MA, July 2023. ACM.

Alex Merenstein, Vasily Tarasov, Ali Anwar, Scott Guthridge, and Erez Zadok. F3: Serving files efficiently in serverless computing. In *Proceedings of the 16th ACM International Systems and Storage Conference (SYSTOR '23)*, Haifa, Israel, June 2023. ACM. Won Best Paper Award.

Ibrahim "Umit" Akgun, Santiago Vargas, Michael Arkhangelskiy, Andrew Burford, Michael McNeill, Aruna Balasubramanian, Anshul Gandhi, and Erez Zadok. Predicting network buffer capacity for BBR fairness. In *36th Conference on Neural Information Processing Systems (NeurIPS 2022) Workshop on ML for Systems*, December 2022.

Anjul Tyagi, Tyler Estro, Geoff Kuenning, Erez Zadok, and Klaus Mueller. PC-Expo: A metrics-based interactive axes reordering method for parallel coordinate displays. *IEEE Transactions on Visualization and Computer Graphics*, October 2022.

Anshul Gandhi, Kanad Ghose, Kartik Gopalan, Syed Hussain, Dongyoon Lee, David Liu, Zhenhua Liu, Patrick McDaniel, Shuai Mu, and Erez Zadok. Metrics for sustainability in data centers. In *Proceedings of the 1st Workshop on Sustainable Computer Systems Design and Implementation (HotCarbon'22)*, San Diego, CA, USA, July 2022. USENIX.

Sayontan Ghosh, Amanpreet Singh, Meyer "Alex" Merenstein, Wei Su, Scott A. Smolka, Erez Zadok, and Niranjan Balasubramanian. SpecNFS: A challenge dataset towards extracting formal models from natural language specifications. In *Proceedings of the 13th Language Resources and Evaluation Conference (LREC'22)*, pages 2166–2176, Marseille, France, June 2022.

Wei Su, Yifei Liu, Gomathi Ganesan, Gerard Holzmann, Scott Smolka, Erez Zadok, and Geoff Kuenning. Model-checking support for file system development. In *Proceedings of the 13th ACM Workshop on Hot Topics in Storage and File Systems (HotStorage '21)*, pages 103–110, Virtual, July 2021. ACM.

Ibrahim 'Umit' Akgun, Ali Selman Aydin, Aadil Shaikh, Lukas Velikov, and Erez Zadok. A machine learning framework to improve storage system performance. In *Proceedings of the 13th ACM Workshop on Hot Topics in Storage and File Systems (HotStorage '21)*, pages 94–102, Virtual, July 2021. ACM.

Alex Merenstein, Vasily Tarasov, Ali Anwar, Deepavali Bhagwat, Julie Lee, Lukas Rupprecht, Dimitris Skourtis, Yang Yang, and Erez Zadok. CNSBench: A cloud native storage benchmark native storage. In *Proceedings of the 19th USENIX Conference on File and Storage Technologies (FAST '21)*, Virtual, February 2021. USENIX Association.

Alex Merenstein, Vasily Tarasov, Ali Anwar, Deepavali Bhagwat, Lukas Rupprecht, Dimitris Skourtis, and Erez Zadok. The case for benchmarking control operations in cloud native storage. In *Proceedings of the 12th USENIX Workshop on Hot Topics in Storage (HotStorage '20)*, Boston, MA, July 2020. USENIX.

Tyler Estro, Pranav Bhandari, Avani Wildani, and Erez Zadok. Desperately seeking ... optimal multi-tier cache configurations. In *Proceedings of the 12th USENIX Workshop on Hot Topics in Storage (HotStorage '20)*, Boston, MA, July 2020. USENIX.

Wei Su, Akshay Aurora, Ming Chen, and Erez Zadok. Supporting transactions for bulk NFSv4 compounds. In *Proceedings of the 13th ACM International Systems and Storage Conference (SYSTOR '20)*, Haifa, Israel, June 2020. ACM.

Ibrahim Umit Akgun, Geoff Kuenning, and Erez Zadok. Re-animator: Versatile high-fidelity storage-system tracing and replaying. In *Proceedings of the 13th ACM International Systems and Storage Conference (SYSTOR '20)*, pages 61–74, Haifa, Israel, June 2020. ACM.

Elinor Schoenfeld, Fan Ye, Bing Zhou, Zongxing Xie, Erez Zadok, Xi Cheng, Jacqueline B. Mondros, Patricia Bruckenthal, and Jignesh Patel. Smart aging: Utilizing low-cost, non-wearable and secure sensing technologies to support the health and wellbeing of community dwelling older adults. In *Proceedings of the HIMSS 20 Preconference Session on Aging and Tech Forum: Addressing the Silver Tsunami*, Orlando, FL, March 2020.

Ibrahim Umit Akgun, Ali Selman Aydin, and Erez Zadok. KMLib: Towards machine learning for operating systems. In *Proceedings of the 2020 On-Device Intelligence Workshop, co-located with the MLSys Conference*, February 2020. Refereed abstract+poster.

Zhen Cao, Geoff Kuenning, and Erez Zadok. Carver: Finding important parameters for storage system tuning. In *Proceedings of the 18th USENIX Conference on File and Storage Technologies (FAST '20)*, Santa Clara, CA, February 2020. USENIX Association.

Muhammad Wajahat, Aditya Yele, Tyler Estro, Anshul Gandhi, and Erez Zadok. Distribution fitting and performance modeling for storage traces. In *27th IEEE International Symposium on the Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2019)*, Rennes, France, October 2019. IEEE Computer Society. (**Won best student paper award**).

Anjul Tyagi, Zhen Cao, Tyler Estro, Klaus Mueller, and Erez Zadok. ICE: Interactive configuration explorer for high dimensional categorical parameter spaces. In *IEEE Conference on Visual Analytics Science and Technology (VAST 2019)*, October 2019.

Zhen Cao, Geoff Kuenning, Klaus Mueller, Anjul Tyagi, and Erez Zadok. Graphs are not enough: Using interactive visual analytics in storage research. In *Proceedings of the 11th USENIX Workshop on Hot Topics in Storage (HotStorage '19)*, Renton, WA, July 2019. USENIX.

Ming Chen and Erez Zadok. Kurma: Secure geo-distributed multi-cloud storage gateways. In *Proceedings of the 12th ACM International Systems and Storage Conference (SYSTOR '19)*, Haifa, Israel, June 2019. ACM.

Zhen Cao, Vasily Tarasov, Sachin Tiwari, and Erez Zadok. Towards better understanding of black-box auto-tuning: A comparative analysis for storage systems. In *Proceedings of the Annual USENIX Technical Conference*, Boston, MA, July 2018. USENIX Association. Data set at `http://download.filesystems.org/auto-tune/ATC-2018-auto-tune-data.sql.gz`.

S. Archak, S. Dixit, R.P. Spillane, and E. Zadok. Multi-tier caching, May 2018. US Patent 9,959,279.

Erez Zadok, Dean Hildebrand, Geoff Kuenning, and Keith Smith. POSIX is dead! Long live... errr... what exactly? In *Proceedings of the 9th USENIX Workshop on Hot Topics in Storage (HotStorage '17)*, Santa Clara, CA, July 2017. USENIX. WACI track.

Ming Chen, Dean Hildebrand, Henry Nelson, Jasmit Saluja, Ashok Subramony, and Erez Zadok. vNFS: Maximizing NFS performance with compounds and vectorized I/O. In *Proceedings of the 15th USENIX Conference on File and Storage Technologies (FAST '17)*, pages 301–314, Santa Clara, CA, February-March 2017. USENIX Association. **(Nominated for best paper award)**.

Bharath Kumar Reddy Vangoor, Vasily Tarasov, and Erez Zadok. To FUSE or not to FUSE: Performance of user-space file systems. In *Proceedings of the 15th USENIX Conference on File and Storage Technologies (FAST '17)*, pages 59–72, Santa Clara, CA, February-March 2017. USENIX Association.

Zhen Cao, Vasily Tarasov, Hari Raman, Dean Hildebrand, and Erez Zadok. On the performance variation in modern storage stacks. In *Proceedings of the 15th USENIX Conference on File and Storage Technologies (FAST '17)*, pages 329–343, Santa Clara, CA, February-March 2017. USENIX Association.

Ming Chen, Arun Vasudevan, Kelong Wang, and Erez Zadok. SeMiNAS: A secure middleware for wide-area network-attached storage. In *Proceedings of the 9th ACM International Systems and Storage Conference (ACM SYSTOR '16)*, Haifa, Israel, June 2016. ACM.

S. Archak, S. Dixit, R.P. Spillane, and E. Zadok. Multi-tier caching, May 2016. US Patent 9,355,109.

Zhen Sun, Geoff Kuenning, Sonam Mandal, Philip Shilane, Vasily Tarasov, Nong Xiao, and Erez Zadok. A long-term user-centric analysis of deduplication patterns. In *Proceedings of the 32nd International IEEE Symposium on Mass Storage Systems and Technologies (MSST '16)*, pages 1–7, Santa Clara, California, May 2016. IEEE.

Sonam Mandal, Geoff Kuenning, Dongju Ok, Varun Shastry, Pilip Shilane, Sun Zhen, Vasily Tarasov, and Erez Zadok. Using hints to improve inline block-layer deduplication. In *Proceedings of the 14th USENIX Conference on File and Storage Technologies (FAST '16)*, pages 315–322, Santa Clara, CA, February 2016. USENIX Association.

Erez Zadok, Aashray Arora, Zhen Cao, Akhilesh Chaganti, Arvind Chaudhary, and Sonam Mandal. Parametric optimization of storage systems. In *HotStorage '15: Proceedings of the 7th USENIX Workshop on Hot Topics in Storage*, Santa Clara, CA, July 2015. USENIX, USENIX.

Vasily Tarasov, Abhishek Gupta, Kumar Sourav, Sagar Trehan, and Erez Zadok. Terra incognita: On the practicality of user-space file systems. In *HotStorage '15: Proceedings of the 7th USENIX Workshop on Hot Topics in Storage*, Santa Clara, CA, July 2015. USENIX, USENIX.

Ming Chen, Dean Hildebrand, Geoff Kuenning, Soujanya Shankaranarayana, Bharat Singh, and Erez Zadok. Newer is sometimes better: An evaluation of NFSv4.1. In *Proceedings of the 2015 ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS 2015)*, Portland, OR, June 2015. ACM.

M. Chen, D. Hildebrand, G. Kuenning, S. Shankaranarayana, V. Tarasov, A. Vasudevan, E. Zadok, and K. Zakirova. Linux NFSv4.1 performance under a microscope. In *Proceedings of USENIX LISA*. USENIX Association, November 2014. Extended Abstract.

Vasily Tarasov, Deepak Jain, Geoff Kuenning, Sonam Mandal, Karthikeyani Palanisami, Philip Shilane, Sagar Trehan, and Erez Zadok. Dmdedup: Device mapper target for data deduplication. In *Proceedings of the Linux Symposium*, pages 83–95, Ottawa, Canada, July 2014.

Z. Li, A. Mukker, and E. Zadok. On the importance of evaluating storage systems' $costs. In *Proceedings of the 6th USENIX Conference on Hot Topics in Storage and File Systems*, HotStorage'14, 2014.

Vasily Tarasov, Deepak Jain, Dean Hildebrand, Renu Tewari, Geoff Kuenning, and Erez Zadok. Improving I/O performance using virtual disk introspection. In *Proceedings of the USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage)*, June 2013.

Vasily Tarasov, Dean Hildebrand, Geoff Kuenning, and Erez Zadok. Virtual machine workloads: The case for new benchmarks for NAS. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST '13)*, San Jose, CA, February 2013. USENIX Association.

Pradeep Shetty, Richard Spillane, Ravikant Malpani, Binesh Andrews, Justin Seyster, and Erez Zadok. Building workload-independent storage with VT-trees. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST '13)*, pages 17–30, San Jose, CA, February 2013. USENIX Association.

Z. Li, A. Desai, C. Bhatt, and E. Zadok. vATM: vSphere adaptive task management. In *In Proceedings of the Seventh International Workshop on Feedback Computing (FC'12)*, September 2012.

David S. H. Rosenthal, Daniel C. Rosenthal, Ethan L. Miller, Ian F. Adams, Mark W. Storer, and Erez Zadok. The economics of long-term digital storage. In *The Memory of the World in the Digital age: Digitization and Preservation*. United Nations Educational, Scientific and Cultural Organization (UNESCO), September 2012.

E. Bartocci, R. Grosu, A. Karmarkar, S. A. Smolka, S. D. Stoller, E. Zadok, and J. Seyster. Adaptive runtime verification. In *Proc. 3rd International Conference on Runtime Verification (RV'12)*, Istanbul, Turkey, September 2012.

Michael A. Bender, Martin Farach-Colton, Rob Johnson, Russell Kraner, Bradley C. Kuszmaul, Dzejla Medjedovic, Pablo Montes, Pradeep Shetty, Richard P. Spillane, and Erez Zadok. Don't thrash: How to cache your hash on flash. In *Proceedings of the 38th International Conference on Very Large Data Bases (VLDB)*, Istanbul, Turkey, August 2012. Morgan Kaufmann.

Vasily Tarasov, Gyumin Sim, Anna Povzner, and Erez Zadok. Efficient I/O scheduling with accurately estimated disk drive latencies. In *In Proceedings of the 8th Annual Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT'12)*, pages 36–45, Pisa, Italy, 2012. The George Washington University.

V. Tarasov, A. Mudrankit, W. Buik, P. Shilane, G. Kuenning, and E. Zadok. Generating realistic datasets for deduplication analysis. In *Proceedings of the Annual USENIX Technical Conference*, Boston, MA, June 2012. USENIX Association.

Z. Li, K. M. Greenan, A. W. Leung, and E. Zadok. Power consumption in enterprise-scale backup storage systems. In *Proceedings of the Tenth USENIX Conference on File and Storage Technologies (FAST '12)*, San Jose, CA, February 2012. USENIX Association.

V. Tarasov, K. S. Kumar, J. Ma, D. Hildebrand, A. Povzner, G. Kuenning, and E. Zadok. Extracting flexible, replayable models from large block traces. In *Proceedings of the Tenth USENIX Conference on File and Storage Technologies (FAST '12)*, San Jose, CA, February 2012. USENIX Association.

M.G. Schultz, E. Eskin, E. Zadok, M. Bhattacharyya, and S.S. J. Systems and methods for detection of new malicious executables, July 2011. US Patent 7,979,907.

Richard P. Spillane, Pradeep J. Shetty, Erez Zadok, Shrikar Archak, and Sagar Dixit. An efficient multi-tier tablet server storage architecture. In *Proceedings of the 2nd ACM Symposium on Cloud Computing (SOCC'11)*, Cascais, Portugal, October 2011.

J. Seyster, P. Radhakrishnan, S. Katoch, A. Duggal, S. D. Stoller, and E. Zadok. Redflag: A framework for analysis of kernel-level concurrency. In *Proc. of the 11th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'11)*, Melbourne, Australia, October 2011.

S. D. Stoller, E. Bartocci, J. Seyster, R. Grosu, K. Havelund, S. A. Smolka, and E. Zadok. Runtime verification with state estimation. In *Proc. 2nd International Conference on Runtime Verification (RV'11)*, San Francisco, CA, September 2011. (**Won best paper award**).

Z. Li, R. Grosu, K. Muppalla, S. A. Smolka, S. D. Stoller, and E. Zadok. Model discovery for energy-aware computing systems: An experimental evaluation. In *Proceedings of the 1st Workshop on Energy Consumption and Reliability of Storage Systems (ERSS'11)*, Orlando, FL, July 2011.

Michel A. Bender, Martin Farach-Colton, Rob Johnson, Bradley C. Kuszmaul, Dzejla Medjedovic, Pablo Montes, Pradeep Shetty, Richard P. Spillane, and Erez Zadok. Don't thrash: How to cache your hash on flash. In *HotStorage '11: Proceedings of the 3rd USENIX Workshop on Hot Topics in Storage*, Portland, OR, June 2011. USENIX Association.

Z. Li, R. Grosu, P. Sehgal, S. A. Smolka, S. D. Stoller, and E. Zadok. On the energy consumption and performance of systems software. In *Proceedings of the 4th Israeli Experimental Systems Conference (ACM SYSTOR '11)*, Haifa, Israel, May/June 2011. ACM.

Vasily Tarasov, Saumitra Bhanage, Erez Zadok, and Margo Seltzer. Benchmarking file system benchmarking: It *IS* rocket science. In *Proceedings of HotOS XIII:The 13th USENIX Workshop on Hot Topics in Operating Systems*, Napa, CA, May 2011.

J. Seyster, K. Dixit, X. Huang, R. Grosu, K. Havelund, S. A. Smolka, S. D. Stoller, and E. Zadok. Aspect-oriented instrumentation with GCC. In *Proc. of the 1st International Conference on Runtime Verification (RV 2010)*, Lecture Notes in Computer Science. Springer, November 2010.

R. Spillane, S. Dixit, S. Archak, S. Bhanage, and E. Zadok. Exporting kernel page caching for efficient user-level I/O. In *Proceedings of the International IEEE Symposium on Mass Storage Systems and Technologies (MSST)*, Incline Village, Nevada, May 2010. IEEE.

Priya Sehgal, Vasily Tarasov, and Erez Zadok. Evaluating performance and energy in file system server workloads. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST '10)*, pages 253–266, San Jose, CA, February 2010. USENIX Association.

Patrick McDaniel, Kevin Butler, Stephen Mclaughlin, Radu Sion, Erez Zadok, and Marianne Winslett. Towards a secure and efficient system for end-to-end provenance. In *Proceedings of the Second USENIX workshop on the Theory and Practice of Provenance (TAPP '10)*, San Jose, CA, February 2010. USENIX Association.

M.G. Schultz, E. Eskin, E. Zadok, M. Bhattacharyya, and S.J. Salvatore. System and methods for detection of new malicious executables, February 2009. US Patent 7,487,544.

D. J. Dean, S. Callanan, and E. Zadok. The visual development of GCC plug-ins. In *Proceedings of the 2009 GCC Developers' Summit*, Montreal, Canada, June 2009.

C. Yalamanchili, G. Sivathanu, K. Vijayasankar, and E. Zadok. DHIS: Discriminating hierarchical storage. In *Proceedings of the Second ACM Israeli Experimental Systems Conference (SYSTOR '09)*, Haifa, Israel, May 2009. ACM.

Rachita Kothiyal, Vasily Tarasov, Priya Sehgal, and Erez Zadok. Energy and performance evaluation of lossless file data compression on server systems. In *Proceedings of the Second ACM Israeli Experimental Systems Conference (SYSTOR '09)*, Haifa, Israel, May 2009. ACM.

R. Spillane, R. Sears, C. Yalamanchili, S. Gaikwad, M. Chinni, and E. Zadok. Story Book: An efficient extensible provenance framework. In *Proceedings of the First USENIX workshop on the Theory and Practice of Provenance (TAPP '09)*, San Francisco, CA, February 2009. USENIX Association.

R. P. Spillane, S. Gaikwad, E. Zadok, C. P. Wright, and M. Chinni. Enabling transactional file access via lightweight kernel extensions. In *Proceedings of the Seventh USENIX Conference on File and Storage Technologies (FAST '09)*, pages 29–42, San Francisco, CA, February 2009. USENIX Association.

S. Sundararaman, G. Sivathanu, and E. Zadok. Selective versioning in a secure disk system. In *Proceedings of the 17th USENIX Security Symposium*, pages 259–274, San Jose, CA, July-August 2008. USENIX Association.

A. Traeger, I. Deras, and E. Zadok. DARC: Dynamic analysis of root causes of latency distributions. In *Proceedings of the 2008 International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS 2008)*, pages 277–288, Annapolis, MD, June 2008. ACM.

D. P. Quigley, K. Albrecht, and E. Zadok. PLEASE: Policy language for easy administration of SELinux. In *Proceedings of the Second annual Computer Security Conference (CSC 2008)*, Myrtle Beach, SC, April 2008.

S. Callanan, D. J. Dean, M. Gorbovitski, R. Grosu, J. Seyster, S. A. Smolka, S. D. Stoller, and E. Zadok. Software monitoring with bounded overhead. In *Proceedings of the 2008 NSF Next Generation Software Workshop, in conjunction with the 2008 International Parallel and Distributed Processing Symposium (IPDPS 2008)*, Miami, FL, April 2008.

E. Zadok. The layers are coming, the layers are coming. In *The 2008 Linux Storage and Filesystem Workshop (LSF'08)*, San Jose, CA, February 2008. USENIX Association.

K. Vijayasankar, G. Sivathanu, S. Sundararaman, and E. Zadok. Exploiting type-awareness in a self-recovering disk. In *Proceedings of the Third ACM Workshop on Storage Security and Survivability (StorageSS 2007)*, pages 25–30, Alexandria, VA, October 2007. ACM.

A. Traeger, K. Thangavelu, and E. Zadok. Round-trip privacy with NFSv4. In *Proceedings of the Third ACM Workshop on Storage Security and Survivability (StorageSS 2007)*, pages 1–7, Alexandria, VA, October 2007. ACM.

N. Joukov, A. M. Krishnakumar, C. Patti, A. Rai, S. Satnur, A. Traeger, and E. Zadok. RAIF: Redundant array of independent filesystems. In *Proceedings of 24th IEEE Conference on Mass Storage Systems and Technologies (MSST 2007)*, pages 199–212, San Diego, CA, September 2007. IEEE.

S. Callanan, D. J. Dean, and E. Zadok. Extending GCC with modular GIMPLE optimizations. In *Proceedings of the 2007 GCC Developers' Summit*, pages 31–37, Ottawa, Canada, July 2007.

Richard P. Spillane, Charles P. Wright, Gopalan Sivathanu, and Erez Zadok. Rapid file system development using ptrace. In *Proceedings of the Workshop on Experimental Computer Science (ExpCS), in conjunction with ACM FCRC*, page 22, San Diego, CA, June 2007.

J. Sipek, Y. Pericleous, and E. Zadok. Kernel support for stackable file systems. In *Proceedings of the 2007 Ottawa Linux Symposium (OLS 2007)*, volume 2, pages 223–227, Ottawa, Canada, June 2007.

A. D. A. Martin and E. Zadok. Autofs - an automounting file system for freebsd 6.x. In *Proceedings of the 2007 BSDCan Technical BSD Conference*, Ottawa, ON, Canada, May 2007.

S. Callanan, R. Grosu, J. Seyster, S. A. Smolka, and E. Zadok. Model predictive control for memory profiling. In *Proceedings of the 2007 NSF Next Generation Software Workshop, in conjunction with the 2007 International Parallel and Distributed Processing Symposium (IPDPS 2007)*, Long beach, CA, March 2007.

J. Sipek and E. Zadok. Making Linux stacking-friendly. In *The 2007 Linux Storage and Filesystem Workshop (LSF'07)*, San Jose, CA, February 2007. USENIX Association.

G. Sivathanu, S. Sundararaman, and E. Zadok. Type-safe disks. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2006)*, pages 15–28, Seattle, WA, November 2006. ACM SIGOPS.

Nikolai Joukov, Ashivay Traeger, Rakesh Iyer, Charles P. Wright, and Erez Zadok. Operating system profiling via latency analysis. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2006)*, pages 89–102, Seattle, WA, November 2006. ACM SIGOPS.

N. Joukov, H. Papaxenopoulos, and E. Zadok. Secure deletion myths, issues, and solutions. In *Proceedings of the Second ACM Workshop on Storage Security and Survivability (StorageSS 2006)*, pages 61–66, Alexandria, VA, October 2006. ACM.

A. Traeger, N. Joukov, J. Sipek, and E. Zadok. Using free web storage for data backup. In *Proceedings of the Second ACM Workshop on Storage Security and Survivability (StorageSS 2006)*, pages 73–77, Alexandria, VA, October 2006. ACM.

D. P. Quigley, J. Sipek, C. P. Wright, and E. Zadok. UnionFS: User- and community-oriented development of a unification filesystem. In *Proceedings of the 2006 Linux Symposium*, volume 2, pages 349–362, Ottawa, Canada, July 2006.

S. Callanan, R. Grosu, X. Huang, S. A. Smolka, and E. Zadok. Compiler-assisted software verification using plug-ins. In *Proceedings of the 2006 NSF Next Generation Software Workshop, in conjunction with the 2006 International Parallel and Distributed Processing Symposium (IPDPS 2006)*, Rhodes Island, Greece, April 2006. DOI 10.1109/IPDPS.2006.1639579.

S. Callanan, R. Grosu, A. Rai, S. A. Smolka, M. R. True, and E. Zadok. Runtime verification for high-confidence systems: A Monte Carlo approach. In *Proceedings of the Second Workshop on Model Based Testing (MBT 2006), held in conjunction with the European Joint Conferences on Theory and Practice of Software (ETAPS 2006)*, pages 41–53, Vienna, Austria, March 2006.

Nikolai Joukov, Timothy Wong, and Erez Zadok. Accurate and efficient replaying of file system traces. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST '05)*, pages 337–350, San Francisco, CA, December 2005. USENIX Association.

N. Joukov and E. Zadok. Adding secure deletion to your favorite file system. In *Proceedings of the third international IEEE Security In Storage Workshop (SISW 2005)*, pages 63–70, San Francisco, CA, December 2005. IEEE Computer Society.

N. Joukov, A. Kashyap, G. Sivathanu, and E. Zadok. Kefence: An electric fence for kernel buffers. In *Proceedings of the First ACM Workshop on Storage Security and Survivability (StorageSS 2005)*, pages 37–43, FairFax, VA, November 2005. ACM. (**Won best short paper award**).

G. Sivathanu, C. P. Wright, and E. Zadok. Ensuring data integrity in storage: Techniques and applications. In *Proceedings of the First ACM Workshop on Storage Security and Survivability (StorageSS 2005)*, pages 26–36, FairFax, VA, November 2005. ACM.

N. Joukov, A. Rai, and E. Zadok. Increasing distributed storage survivability with a stackable raid-like file system. In *Proceedings of the 2005 IEEE/ACM Workshop on Cluster Security, in conjunction with the Fifth IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2005)*, pages 82–89, Cardiff, UK, May 2005. IEEE. (**Won best paper award**).

C. P. Wright, N. Joukov, D. Kulkarni, Y. Miretskiy, and E. Zadok. Auto-pilot: A platform for system software benchmarking. In *Proceedings of the Annual USENIX Technical Conference, FREENIX Track*, pages 175–187, Anaheim, CA, April 2005. USENIX Association.

E. Zadok, S. Callanan, A. Rai, G. Sivathanu, and A. Traeger. Efficient and safe execution of user-level code in the kernel. In *Proceedings of the 2005 NSF Next Generation Software Workshop, in conjunction with the 2005 International Parallel and Distributed Processing Symposium (IPDPS 2005)*, page 221.1, Denver, CO, April 2005.

A. Kashyap, S. Patil, G. Sivathanu, and E. Zadok. I3FS: An in-kernel integrity checker and intrusion detection file system. In *Proceedings of the 18th USENIX Large Installation System Administration Conference (LISA 2004)*, pages 69–79, Atlanta, GA, November 2004. USENIX Association.

Y. Miretskiy, A. Das, C. P. Wright, and E. Zadok. Avfs: An on-access anti-virus file system. In *Proceedings of the 13th USENIX Security Symposium (Security 2004)*, pages 73–88, San Diego, CA, August 2004. USENIX Association.

K. Muniswamy-Reddy, C. P. Wright, A. Himmer, and E. Zadok. A versatile and user-oriented versioning file system. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST '04)*, pages 115–128, San Francisco, CA, March/April 2004. USENIX Association.

Akshat Aranya, Charles P. Wright, and Erez Zadok. Tracefs: A file system to trace them all. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST '04)*, pages 129–143, San Francisco, CA, March/April 2004. USENIX Association.

E. Zadok, J. Osborn, A. Shater, C. P. Wright, K. Muniswamy-Reddy, and J. Nieh. Reducing storage management costs via informed user-based policies. In *Proceedings of the 12th NASA Goddard, 21st IEEE Conference on Mass Storage Systems and Technologies (MSST 2004)*, pages 193–197, College Park, MD, April 2004. IEEE.

C. P. Wright, J. Dave, and E. Zadok. Cryptographic file systems performance: What you don't know can hurt you. In *Proceedings of the Second IEEE International Security In Storage Workshop (SISW 2003)*, pages 47–61, Washington, DC, October 2003. IEEE Computer Society.

C. P. Wright, M. Martino, and E. Zadok. NCryptfs: A secure and convenient cryptographic file system. In *Proceedings of the Annual USENIX Technical Conference*, pages 197–210, San Antonio, TX, June 2003. USENIX Association.

Amit Purohit, Charles P. Wright, Joseph Spadavecchia, and Erez Zadok. Cosy: Develop in user-land, run in kernel-mode. In *Proceedings of the 2003 ACM Workshop on Hot Topics in Operating Systems (HotOS IX)*, pages 109–114, Lihue, Hawaii, May 2003. USENIX Association.

R. Grosu, E. Zadok, S. A. Smolka, R. Cleaveland, and Y. A. Liu. High-confidence operating systems. In *Proceedings of the Tenth SIGOPS European Workshop: "Can we really depend on an OS?"*, pages 205–208, Saint-Emilion, France, September 2002.

J. Spadavecchia and E. Zadok. Enhancing NFS cross-administrative domain access. In *Proceedings of the Annual USENIX Technical Conference, FREENIX Track*, pages 181–194, Monterey, CA, June 2002. USENIX Association.

E. Zadok. Overhauling AMD for the '00s: A case study of GNU autotools. In *Proceedings of the Annual USENIX Technical Conference, FREENIX Track*, pages 287–297, Monterey, CA, June 2002. USENIX Association.

E. Zadok, J. M. Anderson, I. Bădulescu, and J. Nieh. Fast indexing: Support for size-changing algorithms in stackable file systems. In *Proceedings of the Annual USENIX Technical Conference (ATC)*, pages 289–304, Boston, MA, June 2001. USENIX Association.

E. Zadok and J. Nieh. FiST: A language for stackable file systems. In *Proceedings of the Annual USENIX Technical Conference*, pages 55–70, San Diego, CA, June 2000. USENIX Association.

E. Zadok, I. Bădulescu, and A. Shender. Extending file systems using stackable templates. In *Proceedings of the Annual USENIX Technical Conference*, pages 57–70, Monterey, CA, June 1999. USENIX Association.

E. Zadok and I. Bădulescu. A stackable file system interface for Linux. In *LinuxExpo Conference Proceedings*, pages 141–151, Raleigh, NC, May 1999.

E. Zadok and A. Dupuy. HLFSD: Delivering Email to your $HOME. In *Proceedings of the Seventh USENIX Systems Administration Conference (LISA VII)*, pages 243–254, Monterey, CA, November 1993. USENIX Association.

E. Zadok and D. Duchamp. Discovery and hot replacement of replicated read-only file systems, with application to mobile computing. In *Proceedings of the Summer USENIX Technical Conference*, pages 69–85, Cincinnati, OH, June 1992. USENIX Association.

M. G. Schultz, E. Eskin, E. Zadok, M. Bhattacharyya, and S. J. Stolfo. MEF: Malicious Email Filter — A UNIX Mail Filter that Detects Malicious Windows Executables. In *Proceedings of the Annual USENIX Technical Conference, FREENIX Track*, pages 245–252, Boston, MA, June 2001. (**Won best student paper award**).

M. G. Schultz, E. Eskin, E. Zadok, and S. J. Stolfo. Data mining methods for detection of new malicious executables. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 38–49, Oakland, CA, May 2001.

W. Lee, W. Fan, M. Miller, S. Stolfo, and E. Zadok. Toward cost-sensitive modeling for intrusion detection and response. In *Workshop on Intrusion Detection and Prevention, Seventh ACM Conference on Computer Security (CCS)*, pages 1–22, Athens, Greece, November 2000.

Other Publications

Maliha Tabassum. Integrity protections for secure long-term storage. Technical Report FSL-24-05, Computer Science Department, Stony Brook University, December 2024.

Yifei Liu. Towards efficient, scalable, and versatile file system model checking. Technical Report FSL-24-04, Computer Science Department, Stony Brook University, November 2024.

Christopher Smith. Long-term secure archival using proactively secret-shared datastores. Technical Report FSL-24-03, Computer Science Department, Stony Brook University, September 2024.

Alex Merenstein. *Techniques for Storage Performance Measurement and Data Management in Container-Native and Serverless Environments*. PhD thesis, May 2024.

Christopher Smith, Maliha Tabassum, Soumya Chowdary Daruru, Gaurav Kulhare, Arvin Wang, Ethan Miller, and Erez Zadok. Plugging the leaks in secure archival systems. Non-archival Poster presentation at Stony Brook University Department of Computer Science Graduate Research Day, March 2024.

Christopher Smith, Maliha Tabassum, Soumya Chowdary Daruru, Gaurav Kulhare, Arvin Wang, Ethan Miller, and Erez Zadok. Plugging the leaks in secure archival systems. Work in Progress

presentation at the 22nd USENIX Conference on File and Storage Technologies (FAST'24), February 2024.

Christopher Smith, Maliha Tabassum, Soumya Chowdary Daruru, Gaurav Kulhare, Arvin Wang, Ethan Miller, and Erez Zadok. Plugging the leaks in secure archival systems. Non-archival Poster presentation in the 22nd USENIX Conference on File and Storage Technologies (FAST'24), February 2024.

Alex Merenstein. Moving beyond host based virtualization: New techniques for performance measurement and data management in cloud native environments. Technical Report FSL-24-01, Computer Science Department, Stony Brook University, January 2024.

Mário Antunes, Tyler Estro, Pranav Bhandari, Anshul Gandhi, Geoff Kuenning, Yifei Liu, Carl Waldspurger, Avani Wildani, and Erez Zadok. (multi)knee/elbow point detection library, December 2023.

Syed Rafiul Hussain, Patrick McDaniel, Anshul Gandhi, Kanad Ghose, Kartik Gopalan, Dongyoon Lee, Yu David Liu, Zhenhua Liu, Shuai Mu, and Erez Zadok. Verifiable sustainability in data centers, July 2023. https://arxiv.org/abs/2307.11993.

Tyler Estro, Mário Antunes, Pranav Bhandari, Anshul Gandhi, Geoff Kuenning, Yifei Liu, Carl Waldspurger, Avani Wildani, and Erez Zadok. Guiding simulations of multi-tier storage caches using knee detection. Technical Report FSL-23-01, Computer Science Department, Stony Brook University, July 2023.

Elinor Schoenfeld, Tracy Trimboli, Givenchy Ayisi-Boahene, Patricia Bruckenthal, Shelley Horwitz, Erez Zadok, and Fan Ye. Smart aging: Engaging older adults to guide sensor development to support aging in place. In *Issues and experiences of integrating patient and community stakeholder perspectives in aging research*, volume 7, pages 583–583, Tampa, FL, Dec 2023.

Elinor Schoenfeld, Tracy Trimboli, Kaylyn Finnerty, Givenchy Ayisi-Boahene, Patricia Bruckenthal, Shelley Horwitz, Erez Zadok, and Fan Ye. Engaging older adults to guide low-cost non-wearable sensor technology development to age in place: Survey findings. In *Gerontology Society of America Annual Meeting (GSA 2023)*, volume 7, pages 692–692, Tampa, FL, Dec 2023.

Yifei Liu, Gerard Holzmann, Geoff Kuenning, Scott Smolka, and Erez Zadok. The case for model checking emerging file systems. Non-archival poster presentation in the 17th USENIX Symposium on Operating Systems Design and Implementation (OSDI '23), July 2023.

Tyler Estro, Kamalnath Polakam, and Erez Zadok. Using bash notebooks for systems education (user meeting 23). `https://www.chameleoncloud.org/experiment/share/82fbba82-1b34-4562-9866-b7adaec94b19`.

Ibrahim "Umit" Akgun. *Using Machine Learning to Improve Operating Systems' I/O Subsystems*. PhD thesis, December 2022.

Ibrahim "Umit" Akgun. Using machine learning to improve operating systems' I/O subsystems. Technical Report FSL-22-02, Computer Science Department, Stony Brook University, June 2022.

Sayontan Ghosh, Amanpreet Singh, Alex Merenstein, Wei Su, Scott Smolka, Erez Zadok, and Niranjan Balasubramanian. SpecNFS: A challenge dataset towards extracting formal models from natural language specifications. Non-archival poster presentation in the Workshop on Natural Language Processing for Programming in the ACL'21 conference, August 2021.

Tyler Estro. Accelerating multi-tier cache evaluations with intelligent MRC point selection. Technical Report FSL-21-01, Computer Science Department, Stony Brook University, January 2021.

Wei Su. Supporting transactions for bulk NFSv4 compounds. Technical Report FSL-20-02, Computer Science Department, Stony Brook University, December 2020.

Alex Merenstein. CNSBench: A cloud native storage benchmark. Technical Report FSL-20-01, Computer Science Department, Stony Brook University, December 2020.

Ibrahim "Umit" Akgun. Re-animator: Versatile high-fidelity system-call tracing and replaying. Technical Report FSL-19-02, Computer Science Department, Stony Brook University, May 2019.

George Amvrosiadis, Ali R. Butt, Vasily Tarasov, Erez Zadok, Ming Zhao, Irfan Ahmad, Remzi H. Arpaci-Dusseau, Feng Chen, Yiran Chen, Yong Chen, Yue Cheng, Vijay Chidambaram, Dilma Da Silva, Angela Demke-Brown, Peter Desnoyers, Jason Flinn, Xubin He, Song Jiang, Geoff Kuenning, Min Li, Carlos Maltzahn, Ethan L. Miller, Kathryn Mohror, Raju Rangaswami, Narasimha Reddy, David Rosenthal, Ali Saman Tosun, Nisha Talagala, Peter Varman, Sudharshan Vazhkudai, Avani Waldani, Xiaodong Zhang, Yiying Zhang, and Mai Zheng. Data storage research vision 2025: Report on NSF visioning workshop held may 30–june 1, 2018. Technical report, National Science Foundation, February 2019. `https://dl.acm.org/citation.cfm?id=3316807`.

Zhen Cao. *A Practical Auto-Tuning Framework for Storage Systems*. PhD thesis, Stony Brook University, January 2019.

Zhen Cao. A practical, real-time auto-tuning framework for storage systems. Technical Report FSL-18-01, Computer Science Department, Stony Brook University, April 2018.

M. Chen. *Kurma: Efficient and Secure Multi-Cloud Storage Gateways for Network-Attached Storage*. PhD thesis, Computer Science Department, Stony Brook University, April 2017. Technical Report FSL-17-01.

Bharath Kumar Reddy Vangoor. To FUSE or not to FUSE? analysis and performance characterization of the FUSE user-space file system framework. Master's thesis, Stony Brook University, December 2016. Technical Report FSL-16-02, `http://www.fsl.cs.sunysb.edu/docs/fuse/bharath-msthesis.pdf`.

Zhen Cao. Parametric optimization of storage systems. Technical Report FSL-16-01, Computer Science Department, Stony Brook University, January 2016.

Sonam Mandal. Design and implementation of an open-source deduplication platform for research. Technical Report FSL-15-03, Computer Science Department, Stony Brook University, December 2015.

Ming Chen. Kurma: Geo-distributed secure middlewares for cloud-backed network attached storage. Technical Report FSL-15-02, Computer Science Department, Stony Brook University, November 2015.

Arun Olappamanna Vasudevan. Finding the right balance: Security vs. performance with network storage systems. Master's thesis, Stony Brook University, May 2015. Technical Report FSL-15-01, `http://www.fsl.cs.sunysb.edu/docs/arun-msthesis/arun-msthesis.pdf`.

Benixon Arul Dhas, Erez Zadok, James Borden, and Jim Malina. Evaluation of Nilfs2 for shingled magnetic recording (SMR) disks. Technical Report FSL-14-03, Stony Brook University, September 2014.

M. Chen, D. Hildebrand, G. Kuenning, S. Shankaranarayana, V. Tarasov, A. Vasudevan, E. Zadok, and K. Zakirova. Linux NFSv4.1 performance under a microscope. Technical Report FSL-14-02, Stony Brook University, August 2014.

Z. Li. *GreenDM: A Versatile Tiering Hybrid Drive for the Trade-Off Evaluation of Performance, Energy, and Endurance*. PhD thesis, Computer Science Department, Stony Brook University, May 2014.

M. Palmur, Z. Li, and E. Zadok. Cpuidle from user space. Technical Report FSL-13-05, Computer Science Department, Stony Brook University, December 2013.

V. Tarasov. *Multi-dimensional Workload Analysis and Synthesis for Modern Storage Systems*. PhD thesis, Computer Science Department, Stony Brook University, December 2013.

Zhichao Li. System-aware resource scheduling for performance, energy, and reliability in tiered storage systems. Technical Report FSL-13-03, Computer Science Department, Stony Brook University, May 2013.

V. Tarasov. Multi-dimensional workload analysis and synthesis for modern storage systems. Technical Report FSL-13-02, Computer Science Department, Stony Brook University, April 2013.

M. Chen and E. Zadok. Greendm: A versatile hybrid drive for energy and performance. Technical Report FSL-13-01, Computer Science Department, Stony Brook University, April 2013.

J. Seyster. *Runtime Verification of Kernel-Level Concurrency Using Compiler-Based Instrumentation*. PhD thesis, Computer Science Department, Stony Brook University, December 2012. Technical Report FSL-12-05, `www.fsl.cs.sunysb.edu/docs/jseyster-dissertation/redflag.pdf`.

Pradeep J. Shetty, Richard P. Spillane, Ravikant R. Malpani, Binesh Andrews, Justin Seyster, and Erez Zadok. Unifying file systems and databases efficiently. Technical Report FSL-12-06, Stony Brook University Computer Science, September 2012.

A. Mudrankit. A context aware block layer: The case for block layer deduplication. Master's thesis, Stony Brook University, May 2012. Technical Report FSL-12-04.

P. J. Shetty. From tuples to files: a fast transactional system store and file system. Master's thesis, Stony Brook University, May 2012. Technical Report FSL-12-03.

S. Archak, S. Dixit, R.P. Spillane, and E. Zadok. Multi-tier caching, March 2012. US Patent App. 13/159,039.

R. P. Spillane. *Efficient, Scalable, and Versatile Application and System Transaction Management for Direct Storage Layers*. PhD thesis, Computer Science Department, Stony Brook University, Jan 2012. Technical Report FSL-12-02.

J. Seyster. Runtime verification of kernel-level concurrency using compiler-based instrumentation. Technical Report FSL-12-01, Computer Science Department, Stony Brook University, January 2012. `www.fsl.cs.sunysb.edu/docs/jseyster-proposal/redflag.pdf`.

A. Duggal. Stopping data races using Redflag. Master's thesis, Stony Brook University, May 2010. Technical Report FSL-10-02, `http://www.fsl.cs.sunysb.edu/docs/abhinav-thesis/abhinav_thesis.pdf`.

P. Sehgal. Optimizing energy and performance for server-class file system workloads. Master's thesis, Stony Brook University, May 2010. Technical Report FSL-10-01.

S. Callanan. *Flexible Debugging with Controllable Overhead*. PhD thesis, Computer Science Department, Stony Brook University, August 2009. Technical Report FSL-09-01, `http://www.fsl.cs.sunysb.edu/docs/callanan09phdthesis/callanan09phdthesis.pdf`.

R. P. Spillane. SAMT performance proof. Technical Report FSL-09-03, Stony Brook University Computer Science, June 2009. www.fsl.cs.sunysb.edu/~rick.

D. Dean. The visual development of GCC plug-ins with GDE. Master's thesis, Stony Brook University, April 2009. Technical Report FSL-09-04.

R. Kothiyal. Energy and performance evaluation of lossless file data compression on computer systems. Master's thesis, Stony Brook University, April 2009. Technical Report FSL-09-02.

R. P. Spillane, S. Gaikwad, E. Zadok, C. P. Wright, and M. Chinni. Enabling transactional file access via lightweight kernel extensions. Technical Report FSL-08-05, Stony Brook University Computer Science, August 2008.

A. Traeger. *Analyzing Root Causes of Latency Distributions*. PhD thesis, Computer Science Department, Stony Brook University, August 2008. Technical Report FSL-08-04.

J. Seyster. Techniques for visualizing software execution. Technical Report FSL-08-03, Computer Science Department, Stony Brook University, March 2008. www.fsl.cs.sunysb.edu/docs/debugging-tr/software-visualization-tr.pdf.

S. Callanan. Remote debugging with controllable overhead. Technical Report FSL-08-02, Computer Science Department, Stony Brook University, March 2008. www.fsl.cs.sunysb.edu/docs/debugging-tr/debugging-proposal.pdf.

G. Sivathanu. *End-to-End Abstractions for Application-Aware Storage*. PhD thesis, Computer Science Department, Stony Brook University, May 2008. Technical Report FSL-08-01.

D. P. Quigley. PLEASE: Policy language for easy administration of SELinux. Master's thesis, Stony Brook University, May 2007. Technical Report FSL-07-02, www.fsl.cs.sunysb.edu/docs/dquigley-msthesis/dquigley-msthesis.pdf.

N. Joukov. *Versatile, Portable, and Efficient File System Profiling*. PhD thesis, Computer Science Department, Stony Brook University, December 2006. Technical Report FSL-06-05.

C. P. Wright. *Extending ACID Semantics to the File System via ptrace*. PhD thesis, Computer Science Department, Stony Brook University, May 2006. Technical Report FSL-06-04.

N. Joukov, A. Traeger, R. Iyer, C. P. Wright, and E. Zadok. Latency as a viable metric for OS profiling. Technical Report FSL-06-03, Computer Science Department, Stony Brook University, April 2006.

R. Spillane, C. P. Wright, G. Sivathanu, and E. Zadok. Rapid file system development using ptrace. Technical Report FSL-06-02, Computer Science Department, Stony Brook University, January 2006.

C. P. Wright, R. Spillane, G. Sivathanu, and E. Zadok. Extending ACID semantics to the file system. Technical Report FSL-06-01, Computer Science Department, Stony Brook University, January 2006. www.fsl.cs.sunysb.edu/docs/amino-tr/amino.pdf.

G. Sivathanu and E. Zadok. A versatile persistent caching framework for file systems. Technical Report FSL-05-05, Computer Science Department, Stony Brook University, December 2005. www.fsl.cs.sunysb.edu/docs/cachefs-tr/cachefs.pdf.

N. Joukov, A. Traeger, C. P. Wright, and E. Zadok. Benchmarking file system benchmarks. Technical Report FSL-05-04, Computer Science Department, Stony Brook University, December 2005. www.fsl.cs.sunysb.edu/docs/fsbench/fsbench.pdf.

S. Satnur. Storage virtualization with a stackable file system. Master's thesis, Stony Brook University, December 2005. Technical Report FSL-05-03, www.fsl.cs.sunysb.edu/docs/sunil-msthesis/sunil-msthesis.pdf.

E. Zadok, S. Callanan, R. Grosu, A. Rai, S. Smolka, and M. R. True. Runtime verification for high-confidence OS kernels: A Monte Carlo approach. Technical Report FSL-05-02, Computer Science Department, Stony Brook University, November 2005.

A. Rai and E. Zadok. On the role of static analysis in operating system checking and runtime verification. Technical Report FSL-05-01, Computer Science Department, Stony Brook University, May 2005. www.fsl.cs.sunysb.edu/docs/kgcc-rpe/kgcc.pdf.

E. Zadok, C. Wright, A. Aranya, A. Das, Y. Miretskiy, K.K. Muniswamy-Reddy, and A. Himmer. Stackable file systems and methods thereof, December 2005. US Patent App. 10/862,212.

A. Kashyap. File system extensibility and reliability using an in-kernel database. Master's thesis, Stony Brook University, December 2004. Technical Report FSL-04-06, www.fsl.cs.sunysb.edu/docs/kbdbfs-msthesis/kbdbfs.pdf.

A. Aranya. Versatile file system tracing with Tracefs. Master's thesis, Stony Brook University, August 2004. Technical Report FSL-04-05, www.fsl.cs.sunysb.edu/docs/tracefs-msthesis/tracefs.pdf.

G. Sivathanu, C. P. Wright, and E. Zadok. Enhancing file system integrity through checksums. Technical Report FSL-04-04, Computer Science Department, Stony Brook University, May 2004. www.fsl.cs.sunysb.edu/docs/nc-checksum-tr/nc-checksum.pdf.

A. Traeger, A. Rai, C. P. Wright, and E. Zadok. NFS file handle security. Technical Report FSL-04-03, Computer Science Department, Stony Brook University, May 2004. www.fsl.cs.sunysb.edu/docs/nfscrack-tr/nfscrack.pdf.

C. P. Wright and E. Zadok. Operating system support for extensible secure file systems. Technical Report FSL-04-02, Computer Science Department, Stony Brook University, May 2004. www.fsl.cs.sunysb.edu/docs/secfs-rpe/secfs.pdf.

C. P. Wright, J. Dave, P. Gupta, H. Krishnan, E. Zadok, and M. N. Zubair. Versatility and Unix semantics in a fan-out unification file system. Technical Report FSL-04-01b, Computer Science Department, Stony Brook University, October 2004. www.fsl.cs.sunysb.edu/docs/unionfs-tr/unionfs.pdf.

K. Muniswamy-Reddy. Versionfs: A versatile and user-oriented versioning file system. Master's thesis, Stony Brook University, December 2003. Technical Report FSL-03-03, www.fsl.cs.sunysb.edu/docs/versionfs-msthesis/versionfs.pdf.

C. P. Wright, J. Dave, and E. Zadok. Cryptographic file systems performance: What you don't know can hurt you. Technical Report FSL-03-02, Computer Science Department, Stony Brook University, August 2003. www.fsl.cs.sunysb.edu/docs/nc-perf/perf.pdf.

E. Zadok, J. Osborn, A. Shater, C. P. Wright, K. Muniswamy-Reddy, and J. Nieh. Reducing storage management costs via informed user-based policies. Technical Report FSL-02-02, Computer Science Department, Stony Brook University, September 2002. www.fsl.cs.sunysb.edu/docs/equota-policy/policy.pdf.

A. Purohit, J. Spadavecchia, C. Wright, and E. Zadok. Improving application performance through system call composition. Technical Report FSL-02-01, Computer Science Department, Stony Brook University, June 2003. www.fsl.cs.sunysb.edu/docs/cosy-perf/.

O. C. Leonard, J. Nieh, E. Zadok, J. Osborn, A. Shater, and C. Wright. The design and implementation of elastic quotas: A system for flexible file system management. Technical Report CUCS-014-02, Computer Science Department, Columbia University, June 2002. www.cs.columbia.edu/~library.

E. Zadok. *FiST: A System for Stackable File System Code Generation*. PhD thesis, Computer Science Department, Columbia University, May 2001. www.fsl.cs.sunysb.edu/docs/zadok-phd-thesis/thesis.pdf.

Wenke Lee, Matthew Miller, Salvatore Stolfo, Kahil Jallad, Christopher T Park, Erez Zadok, and Vijay Prabhakar. Toward cost-sensitive modeling for intrusion detection. Technical Report CUCS-002-00, Computer Science Department, Columbia University, 2000. http://academiccommons.columbia.edu/download/fedora_content/download/ac:110332/CONTENT/cucs-002-00.pdf.

Erez Zadok, Johan M Andersen, Ion Badulescu, and Jason Nieh. Performance of size-changing algorithms in stackable file systems. Technical Report CUCS-023-00, Computer Science Department, Columbia University, 2000. http://academiccommons.columbia.edu/download/fedora_content/download/ac:110385/CONTENT/cucs-023-00.pdf.

E. Zadok. Stackable file systems as a security tool. Technical Report CUCS-036-99, Computer Science Department, Columbia University, December 1999. www.cs.columbia.edu/~library.

E. Zadok and I. Bădulescu. Usenetfs: A stackable file system for large article directories. Technical Report CUCS-022-98, Computer Science Department, Columbia University, July 1998. www.cs.columbia.edu/~library.

E. Zadok, I. Bădulescu, and A. Shender. Cryptfs: A stackable vnode level encryption file system. Technical Report CUCS-021-98, Computer Science Department, Columbia University, June 1998. www.cs.columbia.edu/~library.

E. Zadok. *FiST: A File System Component Compiler*. PhD thesis, Computer Science Department, Columbia University, April 1997. www.fsl.cs.sunysb.edu/docs/zadok-thesis-proposal.

E. Zadok. Discovery and hot replacement of replicated read-only file systems, with application to mobile computing. Master's thesis, Computer Science Department, Columbia University, October 1997. Technical Report CUCS-036-94, www.cs.columbia.edu/~library.

E. Zadok and A. Lih. PGMAKE: A Portable Distributed Make System. Technical Report CUCS-035-94, Computer Science Department, Columbia University, July 1994. www.cs.columbia.edu/~library.

Manuals

C. P. Wright and E. Zadok. *The Auto-pilot Benchmarking Suite User Manual*, 2.1 edition, July 2005. www.filesystems.org/docs/auto-pilot/.

J. S. Pendry, N. Williams, and E. Zadok. *Am-utils User Manual*, 6.1b3 edition, July 2003. www.am-utils.org.

PATENTS

Multi-Tier Caching, U.S. Patent 9,959,279, issued May 1, 2018.

Multi-Tier Caching, U.S. Patent 9,355,109, issued May 31, 2016.

Systems and methods for detection of new malicious executables, U.S. Patent 7,979,907B2, issued July 12, 2011.

System and methods for detection of new malicious executables, U.S. Patent 7,487,544B2, issued February 3, 2009.

ACADEMIC SERVICE

Program Committee Chair

2019-2020	The 2020 USENIX Annual Technical Conference (ATC 2020) program committee co-chair.
2015	The 13th USENIX Conference on File and Storage Technologies (FAST 2015) program committee co-chair.
2012	The 5th Israeli Experimental Systems Conference (SYSTOR 2012), program committee co-chair.
2010	The First Usenix Workshop on Sustainable Information Technology (SustainIT 2010), program committee co-chair.
2008	The First File-systems and Storage Benchmarking workshop, program chair (hosted at the Storage Systems Research Center at the University of California at Santa Cruz).
2006	ACM Storage Security and Survivability (StorageSS) workshop, program committee co-chair.
2005	Usenix Annual Conference 2005, Invited Talks program co-chair
2004	Usenix Security 2004, Works-in-Progress program chair
2004	Usenix Security 2004, Poster presentations program chair
2003	USENIX FreeNIX 2003 Annual Conference, program chair

Panels

Oct 2021	NSF CAREER review panel
June 2013	Panelist, EMC University Day, "University/Industry Relationships"
Jan 2012	NSF Sustainability Research Network (SRN) program
May 2010	Department of Energy, Computational and Technology Research program
Feb 2009	Panel moderator, The First Usenix Workshop on Sustainable Information Technology (SustainIT 2010), "The Present and Future of Sustainability R&D"
Oct 2009	Panelist, Workshop on Hot Topics in Storage and File Systems (HotStorage'09), "Rethinking File Systems"
Oct 2009	NSF CAREER (CSR+CNS) review panel

Apr 2007	NSF Computer Systems Research (CSR) Parallel and Distributed Operating Systems (PDOS) review panel
Aug 2004	NSF CyberTrust PI meeting panelist, "Software Assurance," Pittsburgh, PA
Aug 2004	Workshop on Information Assurance Education, Stony Brook University (WIAED 2004), panel chair, "How to establish your own IA Curriculum"
Apr 2004	NSF Distributed Systems and Compilers (DSC) review panel
Apr 2001	NSF ITR Operating Systems and Compilers (OSC) review panel, small grants

Editor-in-Chief

Nov 2012–present	Editor-in-Chief, ACM Transactions on Storage (TOS). `https://dl.acm.org/journal/tos/`

Editor

2014	Editorial Board, USENIX Journal of Education in System Administration (JESA). `www.usenix.org/jesa/`
Aug 2009–2012	Associate Editor, ACM Transactions on Storage (TOS). `https://tos.acm.org/`
Jul 2001–Jan 2009	Bulletin Editor, Webmaster, and lists maintainer for IEEE Technical Committee on Operating Systems and Application Environments (TCOS). `www.tcos.org`.

Book Reviews

Summer 2014	Technical reviewer for *The Design and Implementation of the FreeBSD Operating System (2nd Edition)*, Addison-Wesley, 2014, by Marshall Kirk McKusick, George Neville-Neil, and Robert N.M. Watson.
Summer 2005	Technical reviewer for *Understanding the Linux Kernel*, O'Reilly, 3rd ed., by P. Bovet and M. Cesati
Summer 2005	Technical reviewer for *The Design and Implementation of the FreeBSD Operating System*, Addison-Wesley, 2005, by Marshall Kirk McKusick and George V. Neville-Neil (two chapters)
Summer 2002	Technical reviewer for *Understanding the Linux Kernel*, O'Reilly, 2nd ed., by P. Bovet and M. Cesati
January 1994	Chapter 14 in *UNIX Systems for Modern Architectures* by Curt Schimmel (Addison-Wesley, 1994)

Program Committee Member

2019	The 2019 USENIX Annual Technical Conference (ATC 2019), Renton, WA.
2018	The 10th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage '18), Boston, MA.
2018	The 9th Annual Non-Volatile Memories Workshop (NVME 2018), San Diego, CA.
2017	The 4th FAST/Linux Summit, Santa Clara, CA.
2016–2017	The 15th USENIX Conference on File and Storage Technologies (FAST '17), Santa Clara, CA.

2016	The 3rd FAST/Linux Summit, Santa Clara, CA.
2015	The 2nd FAST/Linux Summit, Santa Clara, CA.
2014	USENIX 2nd Summit for Educators in System Administration (SESA 2014), Seattle, Washington (co-located with LISA'14).
2014	The 7th Israeli ACM Experimental Systems Conference (ACM SYSTOR 2014), Haifa, Israel.
2014	The 12th USENIX Conference on File and Storage Technologies (FAST '14), Santa Clara, CA.
2014	The 1st FAST/Linux Summit, Santa Clara, CA.
2013	USENIX 1st Summit for Educators in System Administration (SESA 2013), Seattle, Washington (co-located with LISA'13).
2013	Interactions of NVM/Flash with Operating-Systems and Workloads Workshop (INFLOW 2013), Farmington, PA (co-located with SOSP'13).
2013	International Conference on Parallel and Distributed Systems (ICPADS 2013), Seoul, Korea.
2011	Operating System Support for Next Generation Large Scale NVRAM (NVRAMOS'11), Jeju Island, South Korea.
2011	The 4th Israeli Experimental Systems Conference (SYSTOR 2011), Haifa, Israel.
2011	The 27th IEEE Symposium on Massive Storage Systems and Technologies (MSST2011), Research Track. Colorado.
2011	The 2011 USENIX Annual Technical Conference
2010	The 26th IEEE Symposium on Massive Storage Systems and Technologies (MSST2010), Research Track. Incline Village, Nevada.
2010	The 3rd Israeli Experimental Systems Conference (SYSTOR 2010), Haifa, Israel.
2009	The 21st International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2009).
2009	The first Usenix workshop on the theory and practice of provenance (TAPP '09).
2009	The 2009 ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS 2009), New York, NY.
2009	The 2nd Israeli Experimental Systems Conference (SYSTOR 2009), Haifa, Israel.
2008	Fifth USENIX Conference on File and Storage Technologies (FAST '08), San Jose, CA.
2007	The Third ACM International Workshop on Storage Security and Survivability (StorageSS 2007), Alexandria, VA
2007	Fifth USENIX Conference on File and Storage Technologies (FAST '07)
2006	CCGrid Cluster Security workshop (in conjunction with CCGrid 2006, the Sixth IEEE/ACM International Symposium on Cluster Computing and the Grid), Singapore, Korea.

2005	The 3rd International IEEE Security in Storage Workshop (co-sponsored by Usenix), San Francisco, CA
2005	The First ACM International Workshop on Storage Security and Survivability (StorageSS), "The Paradigm Shift to Info-Centric Protection," Fairfax, VA
2004–2005	CCGrid Cluster Security workshop (in conjunction with CCGrid 2005, the Fifth IEEE/ACM International Symposium on Cluster Computing and the Grid), Cardiff, UK
2004	Workshop on Information Assurance Education, Stony Brook University (WIAED 2004), organizer
2002	USENIX FreeNIX 2002 Annual Conference
2001	First USENIX Conference on File and Storage Technologies (FAST '02)

Session Chair

Oct 2020	Keynote by Richardo Biancini, the 13th ACM International System and Storage Conference
Jul 2019	Co-chair, ACM Workshop on on Hot Topics in Storage and File Systems (HotStorage), "File Systems"
Jan 2015	The annual NSF Secure and Trustworthy Computing (SaTC) program PI meeting (SaTC-PI 2015), "Benchmarking and workloads"
Feb 2014	The 12th USENIX Conference on File and Storage Technologies (FAST '14), "Performance and Efficiency"
June 2011	The 2011 Usenix Annual Technical Conference (Usenix ATC '11), "Storage Deduplication"
July 2013	The 6th Israeli Experimental Systems Conference (SYSTOR '13), "Flash Optimization"
May 2011	The 4th Israeli Experimental Systems Conference (ACM SYSTOR '11), "Adaptation and performance"
May 2009	The 2nd Israeli Experimental Systems Conference (ACM SYSTOR '09), "Deduplication"
Feb 2009	The first Usenix workshop on the theory and practice of provenance (TAPP '09)
May 2009	Fifth USENIX Conference on File and Storage Technologies (FAST '07), "Beyond the Machine Room"
Nov 2005	ACM International Workshop on Storage Security and Survivability (StorageSS), "Cryptographic Storage Security"
	Usenix Security 2004, Works-in-Progress session chair
Aug 2004	Usenix Security 2004, Works-in-Progress session chair
Aug 2004	Usenix Security 2004, Poster presentations session chair
Jun 2002	USENIX FreeNIX Annual Conference, "File Systems"

Jun 2001	USENIX FreeNIX Annual Conference, “Scripting”

Referee

Feb 2022	USENIX FAST 2022
Nov 2020	ACM Transactions on Storage (TOS)
Oct 2018	IEEE Transactions on Parallel and Distributed Systems (TDPS)
Aug 2017	ACM Transactions on Storage (TOS)
2016	Runtime Verification (RV) 2016
2011	Sustainable Computing Informatics and Systems journal
Nov 2010	ACM SIGMOD 2010
Oct 2010	IEEE Internet Computing
Jan 2010	ACM Transactions on Computers (TOCS)
Dec 2009	IEEE Transactions on Parallel and Distributed Systems (TPDS)
Feb 2009	USENIX FAST 2009
Feb 2007	USENIX Annual conference 2007
Jan 2007	Sigmetrics 2007
Jul 2006	Elsevier’s Information Sciences journal
Feb 2006	USENIX Annual conference 2006
Aug 2005	USENIX FAST 2005
Aug 2004–present	ACM Transactions on Storage (TOS) journal
Jun 2004	USENIX OSDI 2004
Nov 2003	USENIX FAST 2004
Dec 2002	USENIX FREENIX 2003
Jun 2002	USENIX OSDI 2002
May 2002	ACM Mobile Networks and Applications (MONET)
Mar 2002	16th Annual ACM International Conference on Supercomputing (ICS)
Mar 2002	LCTES/SCOPES ’02: ACM SIGPLAN Joint Conference on Languages, Compilers, and Tools for Embedded Systems and Software and Compilers for Embedded Systems.
Jan 2002	USENIX FreeNIX Annual Conference
Mar 2001	8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)
Jan 2001	USENIX Annual Conference
Jan 2001	USENIX FreeNIX Annual Conference
May 2000	USENIX LISA 14
Dec 2000	USENIX Annual Conference

May 1999	USENIX LISA 13
May 1999	ACM Twelfth Annual Symposium on User Interface Software and Technology (UIST '99)
Mar 1996	2nd International Mobile Computing Conference
1994	Trends in Software: Software Configuration Management

Committee Service

2021–present	Tiger Team lead, "Data Storage, Data Management, Distributed Ledger Technologies, Cybersecurity, and Biometrics," Stony Brook University.
2020–present	Steering Committee member, USENIX Annual Technical Conference (USENIX ATC)
2020–present	Steering Committee member, ACM Workshop on on Hot Topics in Storage and File Systems (HotStorage)
2021–2022	Co-Chair, President's "One Campus IT Operations" (OCITO) working group, Stony Brook University.
2017–2020	Member, Provost Committee on Faculty Mentorship, Stony Brook University
2017	Member, ad-hoc SL-2 hiring committee for CS Staff Assistant, Computer Science Department, Stony Brook University
2016–2020	Member, STRIDE project advisory committee, Institute for Advanced Computational Science (IACS) and Computer Science Department, Stony Brook University
2016–2019	Chair, Research and IT/Building Operations committee, Computer Science Department, Stony Brook University (note: also official mentor to all junior CS faculty).
2016-2017	Member, ad-hoc hiring committee for CEAS Grant Writer, College of Engineering and Applied Science, Stony Brook University
2016	Member, ABET funding cmte, College of Engineering and Applied Science, Stony Brook University
2015–present	Steering Committee member, USENIX Filesystem and Storage Technologies Conference (USENIX FAST)
2014	Member, Executive committee, Computer Science Department, Stony Brook University
2014	Member, ad-hoc committee for selection of departmental award, Computer Science Department, Stony Brook University
2014	Member, Provost's Outstanding Lecturer Selection Committee, Stony Brook University
2013	Member, Provost's Interdisciplinary Faculty Cluster Selection Committee, Stony Brook University
2013–present	Undergraduate Curriculum Committee, Computer Science Department, Stony Brook University
2013–present	Member, Provost's Massive Open Online Courses (MOOCS) Task Force, Stony Brook University

2012–present	Steering Committee member, The Israeli ACM Experimental Systems Conference (ACM SYSTOR)
2011–2015	Co-Chair, IT Operations committee, Computer Science Department, Stony Brook University
2011–present	Member, University Senate Information Technology Committee, Stony Brook University
2011–present	Member, President's Information Technology Steering Committee on Email and Collaboration, Stony Brook University
2006–2011	Director of IT Operations, Computer Science Department, Stony Brook University
2010–present	Advanced Energy Research and Technology Center (AERTC), Center Director Search Committee member, Stony Brook University
2012	Member, Grant Writer Recruiting committee, Computer Science Department, Stony Brook University
2011	Staff Recruiting committee chair (Windows Server administrator), Computer Science Department, Stony Brook University
2010	Staff Recruiting committee chair (Unix/Networking administrator), Computer Science Department, Stony Brook University
2010	Departmental 5-year Strategic Planning Committee member, Computer Science Department, Stony Brook University
2010	Faculty Recruiting Committee, Computer Science Department, Stony Brook University
2010	Staff Recruiting committee member (Network Administrator), CEWIT Center, Stony Brook University
2010	President's Teaching Excellence Award Selection Committee, Stony Brook University
2009	Staff Recruiting committee chair (Windows client administrator), Computer Science Department, Stony Brook University
2009	Staff Recruiting committee chair (CVC manager), Computer Science Department, Stony Brook University
2009–present	Advanced Energy Research and Technology Center (AERTC), scientific advisor board member, Stony Brook University
2009	President's Teaching Excellence Award Selection Committee, Stony Brook University
2007–2008	Staff Recruiting committee chair (webmaster), Computer Science Department, Stony Brook University
2008–present	New CS building committee, Computer Science Department, Stony Brook University
2007	Staff Recruiting committee chair (Unix administrator), Computer Science Department, Stony Brook University
2007–2010	Co-Chair, IT Operations committee, Computer Science Department, Stony Brook University

2006–present	Center of Excellence in Wireless and Information Technologies (CEWIT) faculty advisory board member, Computer Science Department, Stony Brook University
2006–2007	Chair, Resources and Facilities sub-committee, part of an 8-year departmental review committee. Computer Science Department, Stony Brook University.
2003–present	Visibility (Ranking) committee, Computer Science Department, Stony Brook University
2003–present	Graduate Committee, Computer Science Department, Stony Brook University
2003–present	Permanent Graduate Quals committee, Computer Science Department, Stony Brook University
2003–present	Center of Excellence in Wireless and Information Technologies (CEWIT) committee, Computer Science Department, Stony Brook University
2002–present	Web Site Design committee, Computer Science Department, Stony Brook University
2002–present	Operations Committee, Computer Science Department, Stony Brook University
2001–2006	Faculty Recruiting Committee, Computer Science Department, Stony Brook University
2002–2006	Graduate Recruiting Poster Committee, Computer Science Department, Stony Brook University
2005	SFS Program Director Recruiting committee, Computer Science Department, Stony Brook University
2004	Staff Recruiting committee, Computer Science Department, Stony Brook University
2004	Undergraduate Curriculum Review ad-hoc committee, Computer Science Department, Stony Brook University
2003	PI Award Interface (PIAI) Review committee, Stony Brook University
2003	Graduate Brochure committee, Computer Science Department, Stony Brook University
Spring 2002	Graduate Student Admissions committee, Computer Science Department, Stony Brook University
2002–2003	Graduate Quals Revisions committee, Computer Science Department, Stony Brook University
Spring 2002	Graduate Student Admissions Committee, Computer Science Department, Stony Brook University
Fall 2001	Undergraduate Curriculum Committee, Computer Science Department, Stony Brook University
Fall 2001	Domestic Graduate Student Recruiting Committee, Computer Science Department, Stony Brook University
Fall 2001	Quality of Life Committee, Computer Science Department, Stony Brook University
2001	Ph.D. and M.S. Recruiting Committee, Computer Science Department, Stony Brook University

1998–2000	Ph.D. Committee, Columbia University
1998–1999	Ph.D. Admissions Committee, Columbia University
1990–1999	Facilities Committee, Columbia University

Ph.D. Dissertation Defense Committee

Dec 2021	*Appearance Aware Visualization of High Dimensional Data*, Jenny Hyunjung Lee, Computer Science Department, Stony Brook University.
Aug 2020	Committee Chair, Muhammad Wajahat, *Cost-Efficient Dynamic Management of Cloud Resources via Supervised Learning*, Computer Science Department, Stony Brook University.
Jul 2019	Committee chair, *Analytical Approaches for Dynamic Scheduling in Cloud Environments*, Seyyed Ahmad Javadi, Computer Science Department, Stony Brook University.
Jan 2019	Adviser, *A Practical Auto-Tuning Framework for Storage Systems*, Zhen Cao, Computer Science Department, Stony Brook University.
Oct 2017	*Optimizing System Software with B^ε Trees*, William Jannen, Computer Science Department, Stony Brook University.
May 2017	Committee Chair, Yifeng Sun, *Protection Mechanisms for Virtual Machines on Virtualized Servers*, Computer Science Department, Stony Brook University.
Apr 2017	Adviser, *Kurma: Efficient and Secure Multi-Cloud Storage Gateways for Network-Attached Storage*, Ming Chen, Computer Science Department, Stony Brook University.
Sep 2014	Committee Chair, *Achieving Regulatory Compliance in Data Management*, Sumit Bajaj, Computer Science Department, Stony Brook University.
May 2014	Adviser, *GreenDM: A Versatile Tiering Hybrid Drive for the Trade-Off Evaluation of Performance, Energy, and Endurance*, Zhichao Li, Computer Science Department, Stony Brook University.
May 2014	Committee Chair, *Efficient Implementation Techniques for Block-Level Cloud Storage Systems* Dilip Simha, Computer Science Department, Stony Brook University.
Dec 2013	Adviser, *Multi-dimensional Workload Analysis and Synthesis for Modern Storage Systems*, Vasily Tarasov, Computer Science Department, Stony Brook University.
May 2012	Committee Chair, *Practical Oblivious Outsourced Data Processing*, Peter Williams, Computer Science Department, Stony Brook University.
Jan 2012	Adviser, *Efficient, Scalable, and Versatile Application and System Transaction Management for Direct Storage Layers*, Richard P. Spillane, Computer Science Department, Stony Brook University.
Aug 2009	*Virtualization Mechanisms for Mobility, Security and System Administration*, Shaya Potter, Computer Science Department, Columbia University.
Aug 2009	Adviser, *Flexible Debugging with Controllable Overhead*, Sean Callanan, Computer Science Department, Stony Brook University.
Sep 2008	Committee chair, *Repairable File and Storage Systems*, Ningning Zhu, Computer Science Department, Stony Brook University.

Aug 2008 — Adviser, *Analyzing Root Causes of Latency Distributions*, Avishay Traeger, Computer Science Department, Stony Brook University.

Aug 2008 — *Programmable Ethernet Switches and Their Applications*, Srikant Sharma, Computer Science Department, Stony Brook University.

May 2008 — Committee chair, *Practical Information Flow Based Techniques to Safeguard Host Integrity"*, Weiqing Sun, Computer Science Department, Stony Brook University.

Feb 2008 — Adviser, *End-to-End Abstractions for Application-Aware Storage*, Gopalan Sivathanu (advisee), Computer Science Department, Stony Brook University.

Dec 2006 — Adviser, *Versatile, Portable, and Efficient File System Profiling*, Nikolai Joukov (advisee), Computer Science Department, Stony Brook University.

Jul 2006 — Committee chair, *Availability, Fairness, and Performance Optimization in Storage Virtualization Systems*, Gang "Jason" Peng, Computer Science Department, Stony Brook University.

Jun 2006 — *Implementation Techniques for Scalable, Secure and QoS-Guaranteed Enterprise-Grade Wireless LANs*, Fanglu Guo, Computer Science Department, Stony Brook University.

May 2006 — Adviser, *Extending ACID Semantics to the File System via* `ptrace`, Charles P. Wright (advisee), Computer Science Department, Stony Brook University.

Aug 2002 — *Authoring, Compression, and Playback of Active Video Content*, Anindya Neogi, Computer Science Department, Stony Brook University.

M.S. Thesis Defense Committee

May 2021 — *Transfer Learning Techniques for Sequence Labeling in Network File System Specifications*, Amanpreet Singh, Computer Science Department, Stony Brook University.

Dec 2016 — Adviser, *To FUSE or not to FUSE? Analysis and Performance Characterization of the FUSE User-Space File System Framework*, Bharath Kumar Reddy Vangoor, Computer Science Department, Stony Brook University.

May 2015 — Adviser, *Finding the Right Balance: Security vs. Performance with Network Storage Systems*, Arun Olappamanna Vasudevan, Computer Science Department, Stony Brook University.

May 2012 — Adviser, *A Context Aware Block Layer: The Case for Block Layer Deduplication*, Amar Mudrankit, Computer Science Department, Stony Brook University.

May 2012 — Adviser, *From Tuples to Files: a Fast Transactional System Store and File System*, Pradeep Shetty, Computer Science Department, Stony Brook University.

December 2010 — *TRACECUT: a mechanism of providing history based pointcuts on top of InterAspect*, Ketan Dixit, Computer Science Department, Stony Brook University.

May 2010 — Adviser, *Stopping Data Races Using Redflag*, Abhinav Duggal, Computer Science Department, Stony Brook University.

May 2010 — Adviser, *Optimizing Energy and Performance for Server-Class File System Workloads*, Priya Sehgal, Computer Science Department, Stony Brook University.

May 2010 — *Model Checking the Kaminsky DNS Cache-Poisoning Attack Using PRISM*, Tushar Sudas Deshpande, Computer Science Department, Stony Brook University.

May 2010 — Committee chair, *LFSM (Log Structured Flash Storage Manager)* Gautham Meruva, Computer Science Department, Stony Brook University.

Apr 2009 — Adviser, *The Visual Development of GCC Plug-ins with GDE* Daniel J. Dean, Computer Science Department, Stony Brook University.

Apr 2009 — Adviser, *Energy and Performance Evaluation of Lossless File Data Compression on Computer Systems* Rachita Kothiyal, Computer Science Department, Stony Brook University.

May 2007 — Adviser, *PLEASE: Policy Language for Easy Administration of SELinux*, David P. Quigley, Computer Science Department, Stony Brook University.

Jun 2006 — *Towards the Minimal TCP Offload Architecture*, Lakshmi Kumar Tiruppukuzhi, Computer Science Department, Stony Brook University.

Jun 2006 — *An Approach to Protect System Integrity from Untrusted Applications*, Gaurav Poothia, Computer Science Department, Stony Brook University.

Dec 2005 — Adviser, *Storage Virtualization with a Stackable File System*, Sunil Satnur, Computer Science Department, Stony Brook University.

Dec 2004 — Adviser, *File System Extensibility and Reliability Using an in-Kernel Databases System*, Aditya Kashyap, Computer Science Department, Stony Brook University.

Aug 2004 — Adviser, *Versatile File System Tracing with Tracefs*, Akshat Aranya, Computer Science Department, Stony Brook University.

Mar 2004 — *System and Network Management Using State Machines*, Mohan-Krishna Channa-Reddy, Computer Science Department, Stony Brook University.

Dec 2003 — Adviser, *Versionfs: A Versatile and User-Oriented Versioning File System*, Kiran-Kumar Muniswamy-Reddy (advisee), Computer Science Department, Stony Brook University.

Aug 2003 — *WattProbe: Software-Based Empirical Extraction of Hardware Energy Models*, Manish Prasad, Computer Science Department, Stony Brook University.

Jun 2003 — Adviser, *A System for Improving Application Performance Through System Call Composition*, Amit Purohit, Computer Science Department, Stony Brook University.

Ph.D. Thesis-Proposal (Prelim) Committee

Jun 2022 — Adviser, *Using Machine Learning to Improve Operating Systems' I/O Subsystems*, Ibrahim "Umit" Akgun, Computer Science Department, Stony Brook University.

Apr 2022 — *Exploring Large Parameter Spaces with Machine Learning and Data Visualization*, Anjul Tyagi, Computer Science Department, Stony Brook University.

Apr 2019 — *Cost-Efficient Dynamic Management of Cloud Resources through Supervised Learning*, Muhammad Wajahat, Computer Science Department, Stony Brook University.

Dec 2018 — *Predicting the Future of Archival Storage Using Simulation*, James Byron, Computer Science Department, UC Santa Cruz.

Nov 2018	Committee chair, *Analytical Approaches for Dynamic Scheduling in Cloud Environments*, Seyyed Ahmad Javadi, Computer Science Department, Stony Brook University.
Jun 2018	Committee chair, *Appearance Aware Visualization of High Dimensional Data*, Jenny Hyunjung Lee, Computer Science Department, Stony Brook University.
Apr 2018	Adviser, *A Practical, Real-Time Auto-Tuning Framework for Storage Systems*, Zhen Cao, Computer Science Department, Stony Brook University.
Apr 2017	*Optimizing System Software with B^ε Trees*, William Jannen, Computer Science Department, Stony Brook University.
Nov 2015	Adviser, *KURMA: Geo-Distributed Secure Middlewares for Cloud-Backed Network Attached Storage*, Ming Chen, Computer Science Department, Stony Brook University.
Dec 2014	*Economic Modeling of Long-Term Digital Storage*, Preeti Gupta, Computer Science Department, the University of California at Santa Cruz.
Oct 2013	*Achieving Regulatory Compliance in Data Management*, Sumit Bajaj, Computer Science Department, Stony Brook University.
Aug 2013	Committee chair, *Implementation Techniques for Software-Defined Distributed Storage Systems*, Dilip Simha, Computer Science Department, Stony Brook University.
May 2013	Adviser, *System-Aware Resource Scheduling for Performance, Energy, and Reliability in Tiered Storage Systems*, Zhichao Li, Computer Science Department, Stony Brook University.
Apr 2013	Adviser, *Multi-dimensional Workload Analysis and Synthesis for Modern Storage Systems*, Vasily Tarasov, Computer Science Department, Stony Brook University.
Jan 2012	Adviser, *Runtime Verification of Kernel-Level Concurrency Using Compiler-Based Instrumentation*, Justin Seyster, Computer Science Department, Stony Brook University.
Aug 2011	Adviser, *Efficient, Scalable, and Versatile Application and System Transaction Management for Direct Storage Layers* Richard P. Spillane, Computer Science Department, Stony Brook University.
May 2011	*Practical Oblivious Outsourced Data Processing* Peter Williams, Computer Science Department, Stony Brook University.
Dec 2008	*Organizing, Indexing, and Searching Large-Scale File Systems* Andrew Leung, Computer Science Department, the University of California at Santa Cruz.
Mar 2008	Adviser, *Remote Debugging with Controllable Overhead* Sean Callanan, Computer Science Department, Stony Brook University.
Feb 2008	Adviser, *Analyzing Root Causes of Latency Distributions*, Avishay Traeger, Computer Science Department, Stony Brook University.
Oct 2007	*Practical Information Flow Based Techniques to Safeguard Host Integrity"*, Weiqing Sun, Computer Science Department, Stony Brook University.
Aug 2007	Committee chair, *OS-level Virtualization and Its Applications*, Yang Yu, Computer Science Department, Stony Brook University.
May 2007	Adviser, *End-to-End Abstractions for Application-Aware Storage*, Gopalan Sivathanu, Computer Science Department, Stony Brook University.

Jun 2006 — *Building Enterprise-Grade Wireless LAN with High Mobility, Capacity and Security*, Fanglu Guo, Computer Science Department, Stony Brook University.

Apr 2006 — Adviser, *Versatile, Portable, and Efficient File System Profiling*, Nikolai Joukov, Computer Science Department, Stony Brook University.

Jan 2006 — Adviser, *Extending ACID Semantics to the File System*, Charles P. Wright, Computer Science Department, Stony Brook University.

Aug 2005 — Committee chair, *Repairable File and Storage Systems*, Ningning Zhu, Computer Science Department, Stony Brook University.

Dec 2004 — Committee chair, *Programmable Ethernet switch networks and their applications*, Srikant Sharma, Computer Science Department, Stony Brook University.

Sep 2004 — Committee chair, *Availability Support and Performance Optimization in Stonehenge*, Gang "Jason" Peng, Computer Science Department, Stony Brook University.

Mar 2002 — *Stonehenge: A High Performance Virtualized Storage Cluster With QoS Guarantees*, Lan Huang, Computer Science Department, Stony Brook University.

Oct 2001 — *A System for Collaborative Web Resource Categorization and Ranking*, Maxim Lifantsev.

May 2001 — *Authoring, Compression, and Playback of Active Video Content*, Anindya Neogi, Computer Science Department, Stony Brook University.

Ph.D. Research Proficiency Exam (RPE) Committee

Apr 2022 — Committee chair, *Optimizing Near-Data Processing for Spark*, Sri Pramodh Rachuri, Computer Science Department, Stony Brook University.

Jan 2022 — Adviser, *Model-Checking Support for File System Development*, Yifei Liu, Computer Science Department, Stony Brook University.

Aug 2020 — *An Experimental Evaluation of Cache-Adaptive Algorithms*, Abiyaz Chowdhury, Computer Science Department, Stony Brook University.

May 2019 — Adviser, *Re-Animator: Versatile High-Fidelity System-Call Tracing and Replaying* Ibrahim "Umit" Akgun, Computer Science Department, Stony Brook University.

Aug 2018 — Committee chair, *Reducing tail latency in web applications by addressing variability in service times*, Amoghavarsha Suresh, Computer Science Department, Stony Brook University.

Apr 2018 — Committee chair, *Realizing an Elastic Memcached via Cached Data Migration*, Ubaid Ullah Hafeez, Computer Science Department, Stony Brook University.

Sep 2016 — Dynamic Interference-Aware Load Balancing, Seyyed Ahmad Javadi, Computer Science Department, Stony Brook University.

May 2016 — Committee chair, *Leveraging Machine Learning for Black-Box Autoscaling*, Muhammad Wajahat, Computer Science Department, Stony Brook University.

Apr 2016 — *Spatial Big Data Management and Analytics on Cloud environments and modern computing infrastructures* Hoang Vo, Computer Science Department, Stony Brook University.

Jan 2016	Adviser, *Parametric Optimization of Storage Systems* Zhen Cao, Computer Science Department, Stony Brook University.
Dec 2015	Adviser, *Design and Implementation of an Open-Source Deduplication Platform for Research* Sonam Mandal, Computer Science Department, Stony Brook University.
Sep 2012	*The Design and Application of Content Addressable Storage Systems*, William Jannen Computer Science Department, Stony Brook University.
Aug 2012	Committee chair, *Interaction for Visualization of Multidimensional Data*, Puripant Ruchikachorn, Computer Science Department, Stony Brook University.
Nov 2011	Committee chair, *Trusted Hardware in Secure Data Management*, Sumit Bajaj, Computer Science Department, Stony Brook University.
Sep 2011	Committee chair, *Isolated and Efficient Execution of Unmodified Network Device Drivers*, Yifeng Sun, Computer Science Department, Stony Brook University.
Sep 2011	Committee chair, *The Art of Data Deduplication*, DilipSimha N M, Computer Science Department, Stony Brook University.
Sep 2011	Adviser, *Power and Performance in Compression Systems: A Control Theoretical Approach with Evaluation*, Zhichao Li, Computer Science Department, Stony Brook University.
Sep 2011	Committee chair, *Isolated and Efficient Execution of Unmodified Network Device Drivers*, Sun Yifeng, Computer Science Department, Stony Brook University.
Sep 2011	*Defensive Techniques for Untrustworthy Operating Systems*, Committee chair, Rui Qiao, Computer Science Department, Stony Brook University.
Aug 2011	*Computational Modeling and Analysis of Cardiac Arrhythmia*, Abhishek Murthy, Computer Science Department, Stony Brook University.
Feb 2010	Adviser, *An End-to-End Performance Evaluation of NFSv4 Implementations*, Vasily Tarasov, Computer Science Department, Stony Brook University.
Aug 2009	*Attacks and Defenses: Unix File-System race conditions*, Xiang Cai, Computer Science Department, Stony Brook University.
Mar 2008	Adviser, *Techniques for Visualizing Software Execution*, Justin Seyster, Computer Science Department, Stony Brook University.
May 2006	Committee chair, *Challenges of Long-Term Digital Archiving*, Maohua Lu, Computer Science Department, Stony Brook University.
Feb 2006	*A Survey of Program Transformation Languages and Systems*, Michael Gorbovitski, Computer Science Department, Stony Brook University.
Jun 2005	*Peer to Peer File Download and Streaming*, Gang Wu, Computer Science Department, Stony Brook University.
May 2005	Adviser, *Debugging and Optimizing Systems With Runtime Instrumentation*, Sean Callanan, Computer Science Department, Stony Brook University.
May 2005	Adviser, *File System Benchmarking: Fallacies and Pitfalls*, Avishay Traeger, Computer Science Department, Stony Brook University.

May 2005 — Adviser, *On the Role of Static Analysis in Operating System Checking and Runtime Verification*, Abhishek Rai, Computer Science Department, Stony Brook University.

May 2005 — Adviser, *Ensuring Data Integrity in Storage: Techniques and Applications*, Gopalan Sivathanu, Computer Science Department, Stony Brook University.

Feb 2005 — *A Survey on Virtualization Techniques*, Susanta Nanda, Computer Science Department, Stony Brook University.

Sep 2004 — *Enterprise Digital Rights Management: Solutions against Information Theft by Insiders*, Yang Yu, Computer Science Department, Stony Brook University.

May 2004 — Adviser, *Operating System Support for Extensible Secure File Systems*, Charles P. Wright, Computer Science Department, Stony Brook University.

Feb 2004 — *Traffic Analysis: From Stateful Firewall to Network Intrusion Detection System*, Fanglu Guo, Computer Science Department, Stony Brook University.

Feb 2004 — *Damage Recovery from Malicious Code and Operator Errors*, Weiqing Sun, Computer Science Department, Stony Brook University.

Sep 2003 — Adviser, *Internet Worms as Internet-Wide Threat*, Nikolai Joukov, Computer Science Department, Stony Brook University.

Mar 2003 — *Cache Oblivious Algorithms: Theory and Practice*, Piyush Kumar, Computer Science Department, Stony Brook University.

Jan 2003 — Committee chair, *CDN: Content Distribution Network*, Gang "Jason" Peng, Computer Science Department, Stony Brook University.

Aug 2001 — *Pattern Based Intrusion Detection Systems*, Prem Uppuluri, Computer Science Department, Stony Brook University.

Other One-Time Committee Service

2019 — Office of Vice-President for Research, Seed Grant review committee Bi-annual Awards committee, Stony Brook University.

Aug 2012 — Bi-annual Awards committee, Computer Science Department, Stony Brook University.

Nov 2011 — Member, Steering Committee on High-Performance Computing, Stony Brook University and Brookhaven National Labs.

Oct 2010 — Annual Merit Raise committee, Computer Science Department, Stony Brook University.

Aug 2010 — Bi-annual Awards committee, Computer Science Department, Stony Brook University.

Oct 2009 — Annual Merit Raise committee, Computer Science Department, Stony Brook University.

Sep 2008 — Annual Merit Raise committee, Computer Science Department, Stony Brook University.

Sep 2007 — Annual Merit Raise committee, Computer Science Department, Stony Brook University.

Jan 2007 — Examiner for CS Ph.D. Qualifying Exams, Computer Science Department, Stony Brook University.

Jul 2006	Bi-annual Awards committee, Computer Science Department, Stony Brook University.
Apr 2006	Judge, *Compilers* session. Graduate Research Conference (GRC), Computer Science Department, Stony Brook University.
Sep 2005	Annual Merit Raise committee, Computer Science Department, Stony Brook University.
Aug 2005	Ad-hoc Steering Committee on Strategic Planning, Computer Science Department, Stony Brook University.
May 2004	Judge, *Security* session. Graduate Research Conference (GRC), Computer Science Department, Stony Brook University.
May 2004	Examiner for CS Ph.D. Qualifying Exams, Computer Science Department, Stony Brook University.
Jan 2004	Examiner for CS Ph.D. Qualifying Exams, Computer Science Department, Stony Brook University.
May 2003	Judge, *Security* session. Graduate Research Conference (GRC), Computer Science Department, Stony Brook University.
Oct 2003	Ph.D. qualifying exam chair (Hardware exam), Computer Science Department, Stony Brook University.
Oct 2003	Examiner for CS Ph.D. Qualifying Exams, Computer Science Department, Stony Brook University.
May 2003	Examiner for CS Ph.D. Qualifying Exams, Computer Science Department, Stony Brook University.
May 2002	Examiner for CS Ph.D. Qualifying Exams, Computer Science Department, Stony Brook University.
May 2001	Examiner for CS Ph.D. Qualifying Exams, Computer Science Department, Stony Brook University.

Other Departmental Service

2008–present	Course Coordinator, Systems Administration, (ISE-311)
2003–present	Course Coordinator, Introduction to Programming in C (CSE-130)
2002–present	Course Coordinator, Advanced Systems Programming in Unix/C (CSE-376)
2001–present	Course Coordinator, Operating Systems (CSE-506)
2003–2005	Course Coordinator, Intermediate Programming in C and C++ (CSE-230)

Community Work

Nov 2008	"How Computers Work," guest speaker at Mount Elementary 4th grade (teacher: Jo-Ann Hudecek), Stony Brook Three Village School District, Stony Brook, NY.
2003	Stony Brook Child Care Services Inc. (SBCCSI), fund-raising committee member
2002–present	Faculty sponsor of the Stony Brook Linux Users Group (SBLUG)
1997–present	Maintainer of FiST, a stackable file system template system

1992–present Maintainer of Amd, the Berkeley Automounter (am-utils)

Shepherding

2016–2017 The 15th USENIX Conference on File and Storage Technologies (FAST '17) (1 paper)

2011 The 2011 Usenix Annual Technical Conference (Usenix ATC 2011) (1 paper)

2009 The 2009 ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS 2009) (2 papers)

2009 Fifth USENIX Conference on File and Storage Technologies (FAST '09) (1 paper)

2007 USENIX FAST Annual Conference (2 papers)

2003 USENIX FreeNIX Annual Conference (2 papers)

2002 USENIX FreeNIX Annual Conference (4 papers)

2001 First USENIX Conference on File and Storage Technologies (FAST '02) (1 paper)

Software Released or Maintained

2010–present Wrapfs Stackable File System for Linux

2007–present GCC Plugin Infrastructure.

2005–present Auto-Pilot: A Suite of Tools for Conducting Benchmarks, `www.filesystems.org/project-autopilot.html`

2004–present Unionfs: A Stackable Unification File System, `https://unionfs.filesystems.org`. (As of January 2007, Unionfs has been a part of Andrew Morton's Linux Kernel source tree. Andrew is the person in charge of supplying new patches to Linus Torvalds for new kernel releases, therefore Unionfs is on its way to becoming a part of the Linux kernel.)

1997–present FiST: Stackable File System Language and Templates, `www.filesystems.org/`

1992–present Am-Utils: The Berkeley Automounter Suite of Utilities, `www.am-utils.org`

2004–2005 I^3FS: In-Kernel Integrity Checker and Intrusion Detection File System, `www.fsl.cs.sunysb.edu/project-i3fs.html`

2003–2004 KBDB: In-Kernel Berkeley DB Databases NFScrack: NFS File Handle Security, KBDBFS: File System Extensibility and Reliability Using an in-Kernel Database, `www.fsl.cs.sunysb.edu/project-kbdb.html`

2002 ENF: Enhancing NFS Cross-Administrative Domain Access, `ftp://ftp.fsl.cs.sunysb.edu/pub/enf/`

1999 PGmake: Parallel GNU Make, `www.fsl.cs.sunysb.edu/project-pgmake.html`

TALKS

Invited Talks

May 2017 *Oh, what a tangled web we weave: practice and deception in system analysis*, Performance Evaluation and Analysis Meetup, IBM Research–Haifa, Haifa, Israel.

May 2017 *vNFS: Maximizing NFS Performance with Compounds and Vectorized I/O*, ACM SYSTOR 2017 (highlights paper track), Haifa, Israel.

Feb 2017	*Oh, what a tangled web we weave: practice and deception in system analysis*, Tintri, Inc., Mountain View, CA.
Dec 2016	*Oh, what a tangled web we weave: practice and deception in system analysis*, Harvey Mudd College, Claremont, CA.
Jun 2016	*Oh, what a tangled web we weave: practice and deception in system analysis* Keynote address at inaugural workshop: The 1st International Workshop on System Analytics and Characterization (SAC'16, co-located with ACM SIGMETRICS), Antibes Juan-Les-Pins, France.
Feb 2016	*Parametric Optimization of Storage Systems*, VMware Inc., Palo Alto, CA.
Feb 2006	*File System and Storage Benchmarking: On Formalizing Filebench Workloads*, a Birds-of-Feather (BoF) talk presented at the 2016 Usenix FAST conference, Santa Clara, CA.
Jun 2012	*Storage Security Overview*, CA/CEWIT center inauguration workshop, Tel Aviv University, Tel Aviv, Israel.
May 2012	*Synthesizing Realistic Workload Models and Dedup Datasets*, IBM Haifa Research Labs, Tel Aviv, Israel.
May 2012	*Optimizing Performance and Energy Use of Large Data Systems*, NSF Industry/University Center (I/UCRC) for Dynamic Data Analytics (CDDA) 2012 workshop, Stony Brook, NY.
Feb 2012	*Scaling OS Storage Stack Performance Using NVRAM Technologies*, Fusion-io, San Jose, CA
Nov 2011	*Scaling and Understanding Large Data I/O Performance*, NSF Industry/University Center (I/UCRC) for Dynamic Data Analytics (CDDA) 2011 workshop, Rutgers, NJ.
Nov 2011	*Scaling I/O Performance for Large Data Sets*, 2011 CEWIT Center Conference, Hauppauge, NY.
Nov 2011	*Scaling OS Storage Stack Performance Using NVRAM Technologies*, Operating System Support for Next Generation Large Scale NVRAM (NVRAMOS'11), Jeju Island, South Korea.
Aug 2011	*Optimizing Server Performance and Energy*, High-End Computing Filesystems and I/O Workshop (HEC FSIO 2011), Arlington, VA.
May 2011	*On the Energy Consumption and Performance of Systems Software*, The 4th Israeli Experimental Systems Conference (ACM SYSTOR '11), Haifa, Israel.
Feb 2011	*FSL Storage Research Overview*, Riverbed, Inc, Menlo Park, CA.
Nov 2010	*Server-Class Energy and Performance Evaluations, or, Mommy, Daddy, where did my Joules go?*, An SBCS Faculty Symposium series talk, sponsored by the Stony Brook Computer Society's ACM Chapter. Stony Brook University, Computer Science Department, Stony Brook, NY.
May 2010	*Server-Class Energy and Performance Evaluations*, The Third Israeli Experimental Systems Conference (ACM SYSTOR '10), Haifa, Israel.

Feb 2010	*On the Science of Power Management: Encouraging Sustainability R&D*, First USENIX Workshop on Sustainable Information Technology (SustainIT'10), San Jose, CA.
Feb 2010	*FSL Research Overview*, Data Domain, Inc, Santa Clara, CA.
Nov 2009	Priya Sehgal and Erez Zadok, *Improving Energy & Performance For Server Workloads*, The third Advanced Energy Research & Technology Center (AERTC) conference, Hauppauge, NY.
July 2009	*Energy and Performance Characterization of Storage Stacks*, IBM Watson Research Labs, Hawthorne, NY.
May 2009	*DHIS: Discriminating Hierarchical Storage*, The Israeli Experimental Systems Conference (ACM SYSTOR '09), Haifa, Israel.
May 2009	*Energy and Performance Evaluation of Lossless File Data Compression on Server Systems*, The Israeli Experimental Systems Conference (ACM SYSTOR '09), Haifa, Israel.
February 2009	*How to Cheat in Benchmarking*, a Birds-of-Feather (BoF) talk presented at the 2009 Usenix FAST conference, San Francisco, CA.
Nov 2008	*Mommy, Daddy, Why is My New Laptop So Dog Slow?!*, Parents' Day lecture, Stony Brook University, Stony Brook, NY.
May 2007	*Why is my new laptop so dog slow?!*, An SBCS Faculty Symposium series talk, sponsored by the Stony Brook Computer Society's ACM Chapter. Stony Brook University, Computer Science Department, Stony Brook, NY.
July 2006	*Securing Applications using Operating System Transactions*, Center for Information Protection, NSF I/UCRC organizational workshop, Stony Brook, NY.
June 2006	*Adding security to file systems, one layer at a time*, IBM's Storage Systems Strategy meeting, Mountain View, CA.
June 2006	*File System Benchmarking: Fallacies, Pitfalls, and Beyond*, VMware, Inc., Palo Alto, CA.
December 2005	*File System Benchmarking and Tools: Past, Present, and Future*, a Birds-of-Feather (BoF) talk presented at the 2006 Usenix FAST conference, San Francisco, CA.
December 2005	*File System Benchmarking: Fallacies, Pitfalls, and Beyond*, Penn State University, Computer Science Department, State College, PA.
October 2005	*File System Benchmarking: Fallacies, Pitfalls, and Beyond*, IBM Watson Research Labs, Hawthorne, NY.
October 2005	*File System Benchmarking: Fallacies, Pitfalls, and Beyond*, CSE-600 talk, Stony Brook University, Computer Science Department, Stony Brook, NY.
September 2005	*File System Benchmarking: Fallacies, Pitfalls, and Beyond*, UC Santa Cruz, Computer Science Department, Santa Cruz, California.
June 2005	*Adding security to file systems, one layer at a time*, Second Security Summit, Storage Networking Industry Association (SNIA), Pittsburgh, Pennsylvania
May 2005	*File System Security*, 35th Anniversary Celebration, Stony Brook University, Computer Science Department (CS@35), Stony Brook, New York

April 2005	*Efficient and Safe Execution of User-Level Code in the Kernel*, the 2005 NSF Next Generation Software Workshop, in conjunction with the 2005 International Parallel and Distributed Processing Symposium (IPDPS), Denver, Colorado
October 2004	*File System and OS Security*, talk given to Korean delegation led by Dr. Ho, Stony Brook University, Stony Brook, New York
October 2004	*File System and OS Security*, Computer Associates, Islandia, New York
June 2003	*ASK: Active Spam Killer*, Annual Usenix Conference, FREENIX Track, San Antonio, Texas
February 2003	*File System Security*, New York State Cyber-Security Symposium, Utica, New York
April 2000	*FiST: A System for File System Code Generation*, Computer Science Department, University of Chicago
March 2000	*FiST: A System for File System Code Generation*, Computer Science Department, University of Rochester
March 2000	*FiST: A System for File System Code Generation*, Computer Science Department, Stony Brook University
March 2000	*FiST: A System for File System Code Generation*, Computer Science Department, University of California and Santa Barbara
March 2000	*FiST: A System for File System Code Generation*, Computer Science Department, Penn State University
February 2000	*FiST: A System for File System Code Generation*, Computer Science Department, Bell Labs, New Jersey

Interviews

November 4, 2005	LIA magazine, with Ellen Sterling; what small businesses can do to protect themselves.
June 29, 2001	Dr. Dobb's Technetcast, with Phillippe Lourier; my NFS and Automounters book, status of Linux file system development.

Other Presentations

Mar 2012	Erez Zadok, Steering Committee Town Hall meeting, presenting Google Apps for Education, Stony Brook University.
Feb 2012	Daniel C. Rosenthal, University of California, Santa Cruz; David S. H. Rosenthal, Stanford University Libraries; Ethan L. Miller and Ian F. Adams, University of California, Santa Cruz; Mark W. Storer, NetApp; Erez Zadok, Stony Brook University. *Toward an Economic Model of Long-Term Storage* WIP presented at the *Tenth USENIX Conference on File and Storage Technologies (FAST '12)*, San Jose, CA.
Feb 2012	Zhichao Li, Stony Brook University; Kevin M. Greenan and Andrew W. Leung, EMC Corporation; Erez Zadok, Stony Brook University. *Power Consumption in Enterprise-Scale Backup Storage Systems* Poster presented at the *Tenth USENIX Conference on File and Storage Technologies (FAST '12)*, San Jose, CA.
Feb 2012	V. Tarasov and S. Kumar, Stony Brook University; J. Ma, Harvey Mudd College; D. Hildebrand and A. Povzner, IBM Almaden Research; G. Kuenning, Harvey Mudd College; E. Zadok, Stony Brook University *Extracting Flexible, Replayable Models from*

Large Block Traces Poster presented at the *Tenth USENIX Conference on File and Storage Technologies (FAST '12)*, San Jose, CA.

Nov 2011 — Zhichao Li, R. Grosu, S. A. Smolka, S. D. Stoller, and E. Zadok, Stony Brook University. *Power and Performance in Compression Systems: A Control Theoretical Approach with Evaluation* Poster presented at the *Eighth CEWIT conference (CEWIT '11)*, Hauppauge, NY.

Oct 2011 — Zhichao Li, R. Grosu, S. A. Smolka, S. D. Stoller, and E. Zadok, Stony Brook University. *Power and Performance in Compression Systems: A Control Theoretical Approach with Evaluation* Poster presented at the *Fifth AERTC conference (AERTC '11)*, Buffalo, NY.

May 2011 — Vasily Tarasov, Koundinya Santhosh Kumar, and Erez Zadok, Stony Brook University; Geoff Kuenning, Harvey Mudd College. *T2M: Converting I/O Traces to Workload Models*, Poster presented at the *13th USENIX Workshop on Hot Optics in Operating Systems (HotOS '11)*, Napa, CA.

Feb 2011 — Vasily Tarasov, Koundinya Santhosh Kumar, and Erez Zadok, Stony Brook University; Geoff Kuenning, Harvey Mudd College. *T2M: Converting I/O Traces to Workload Models*, WIP and Poster presented at the *Ninth USENIX Conference on File and Storage Technologies (FAST '11)*, San Jose, CA.

Feb 2011 — Erez Zadok, Stony Brook University and Geoff Kuenning, Harvey Mudd College. *Benchmarking and Tracing: New Horizons*, Birds-of-a-feather (BOF) presented at the *Ninth USENIX Conference on File and Storage Technologies (FAST '11)*, San Jose, CA.

Oct 2011 — Justin Seyster, Abhinav Duggal, Prabakar Radhakrishnan, Scott D. Stoller, and Erez Zadok, Stony Brook University. *Redflag: Detailed Runtime Analysis of Kernel-Level Concurrency*, Poster presented at the *9th USENIX Symposium on Operating Systems Design and Implementations (OSDI '10)*, Vancouver, BC, Canada.

Feb 2010 — Vasily Tarasov, Sujay Godbole, and Erez Zadok. *NSFv4 Implementations: Who Performs Better, When, and Why*, poster presented at the *Eighth USENIX Conference on File and Storage Technologies (FAST '10)*, San Jose, CA.

Feb 2010 — Rick Spillane and Erez Zadok. *Down with the VFS and His Insidious Caches!*, WIP presented at the *Eighth USENIX Conference on File and Storage Technologies (FAST '10)*, San Jose, CA.

Jan 2010 — Erez Zadok and Radu Sion. *Energy and Cost Analysis of Cloud Computing*, NSF Industry/University Center (I/UCRC) for Dynamic Data Analytics (CDDA) planning workshop, New York, NY.

Oct 2009 — Priya Sehgal, Vasily Tarasov, and Erez Zadok. *Server Energy and Performance Evaluations*, poster presented at the *2009 CEWIT Center Conference*, Islandia, NY.

Dec 2008 — Rick Spillane, Chaitanya Yalamanchili, Sachin Gaikwad, Manjunath Chinni, and Erez Zadok. *Honor: A Serializing On-Disk Writeback Buffer*. WIP presented at the *Eighth Symposium on Operating Systems Design and Implementation (OSDI '08)*, San Diego, CA.

Mar 2007 — D. Dean, S. Callanan, and E. Zadok. *Developing GCC Transformations Faster with*

Gimple Viz Graduate Research Conference (GRC), Computer Science Department, Stony Brook University.

Mar 2007 — J. Seyster, S. Callanan, R. Grosu, S. Smolka, and E. Zadok. *Memcov A High Performance Memory Profiling Tool*. Graduate Research Conference (GRC), Computer Science Department, Stony Brook University.

Mar 2007 — R. Spillane and E. Zadok. *A Lot for a Little: Transactional File Systems*. Graduate Research Conference (GRC), Computer Science Department, Stony Brook University.

Apr 2006 — R. Spillane, C. P. Wright, and E. Zadok, *Goanna* (Best Paper Award), Graduate Research Conference (GRC), Computer Science Department, Stony Brook University.

Apr 2006 — N. Joukov, H. Papaxenopoulos, and E. Zadok, *Secure Deletion Myths, Issues, and Solutions*. Graduate Research Conference (GRC), Computer Science Department, Stony Brook University.

Sep 2005 — E. Zadok, *A Layered Approach to Securing File Systems*, poster presented at the NSF CyberTrust PI meeting, Newport Beach, CA.

Oct 2005 — N Joukov, R. Iyer, A. Traeger, C. P. Wright, and E. Zadok. Versatile, Portable, and Efficient OS Profiling via Latency Analysis. Poster presented at the *20th ACM Symposium on Operating Systems Principles (SOSP'05)*, Brighton, UK.

Jul 2005 — D. Quigley, C. P. Wright, and E. Zadok. *UnionFS: Namespace Management*. Cyber Corps Summer Symposium 2005, Syracuse, New York.

Apr 2005 — D. Quigley, C. P. Wright, and E. Zadok. *UnionFS*. URECA (undergraduate research) presentation at Stony Brook.

Aug 2004 — Y. Miretskiy, A. Das, C. P. Wright, and E. Zadok. Avfs: An On-Access Anti-Virus File System. Poster presented at the *Workshop on Information Assurance Education* (WIAED 2004), Stony Brook University

Aug 2004 — C. P. Wright, M. C. Martino, and E. Zadok. NCryptfs: A Secure and Convenient Cryptographic File System. Poster presented at the *Workshop on Information Assurance Education* (WIAED 2004), Stony Brook University

May 2004 — P. Gupta, H. Krishnan, C. P. Wright, and E. Zadok. Versatility and Unix Semantics in a Fan-Out Unification File System. Graduate Research Conference (GRC), Computer Science Department, Stony Brook University.

May 2004 — A. Kashyap, J. Dave, H. Krishnan, C. P. Wright, M. Nayyer Zubair, and E. Zadok. Using Berkeley DB in the Linux Kernel. Graduate Research Conference (GRC), Computer Science Department, Stony Brook University.

May 2004 — Y. Miretskiy, A. Das, C. P. Wright, and E. Zadok. Avfs: An On-Access Anti-Virus File System (Best Paper Award). Graduate Research Conference (GRC), Computer Science Department, Stony Brook University.

May 2004 — A. Aranya, C. P. Wright, and E. Zadok Tracefs: A File System to Trace Them All. Graduate Research Conference (GRC), Computer Science Department, Stony Brook University

Apr 2004 — C. P. Wright and E. Zadok. A Stackable Smorgasbord. Poster presented at the *Third USENIX Conference on File and Storage Technologies (FAST '04)*, San Francisco, CA.

Apr 2004	C. P. Wright and E. Zadok. *Avfs: An On-Access Anti-Virus File System* WIP presented at the *Third USENIX Conference on File and Storage Technologies (FAST '04)*, San Francisco, CA.
Apr 2004	K. Muniswamy-Reddy, and E. Zadok. Elastic Quotas. Poster presented at the *12th NASA Goddard, 21st IEEE Conference on Mass Storage Systems and Technologies (MSST 2004)*, College Park, MD.
Nov 2003	C. P. Wright and E. Zadok. Secure and Mobile File Systems. Poster presented at the *CEWIT 2003* conference, Woodbury, NY.
June 2003	P. Gupta, H. Krishnan, K. Muniswamy-Reddy, C. P. Wright, and E. Zadok. Unionfs: A Flexible, and Portable Unioning File System. WIP presented at the *Annual USENIX Technical Conference*, San Antonio, TX.
June 2003	A. Himmer, K. Muniswamy-Reddy, C. P. Wright, and E. Zadok. Versionfs: A Transparent and Portable Versioning File System. WIP presented at the *Annual USENIX Technical Conference*, San Antonio, TX.
May 2003	A. Himmer and E. Zadok. Versionfs: A Transparent and Portable Versioning File System. Graduate Research Conference (GRC), Computer Science Department, Stony Brook University.
May 2003	E. Zadok, J. Osborn, A. Shater, C. P. Wright, K. Muniswamy-Reddy, and J. Nieh. Elastic Quotas: Stretch Your Disk Space. Graduate Research Conference (GRC), Computer Science Department, Stony Brook University.
May 2003	C. P. Wright, M. C. Martino, and E. Zadok. NCryptfs: A Secure and Convenient Cryptographic File System. Graduate Research Conference (GRC), Computer Science Department, Stony Brook University.
May 2003	A. Purohit, C. P. Wright, J. Spadavecchia, and E. Zadok. Cosy: Develop in User-Land, Run in Kernel-Mode. Graduate Research Conference (GRC), Computer Science Department, Stony Brook University.
April 2003	M. Prasad and E. Zadok. Transparent Page Cache Coherence Support for Linux-based Stackable File Systems. WIP presented at the *Second USENIX Conference on File and Storage Technologies (FAST '03)*, San Francisco, CA.
Fall 2002	E. Zadok. Linux Kernel Internals. Guest lecture in CSE-306.
December 2002	A. Purohit, J. Spadavecchia, C. Wright, and E. Zadok. Improving Application Performance Through System Call Composition WIP presented at the *Fifth Symposium on Operating Systems Design and Implementation (OSDI '02)*, Boston, MA.
December 2002	C. Wright, M. Martino, and E. Zadok. NCryptfs: A Secure and Convenient Cryptographic File System. WIP presented at the *Fifth Symposium on Operating Systems Design and Implementation (OSDI '02)*, Boston, MA.
December 2002	E. Zadok, J. Osborn, C. P. Wright, K. Muniswamy-Reddy. Elastic Quotas. WIP presented at the *Fifth Symposium on Operating Systems Design and Implementation (OSDI '02)*, Boston, MA.
June 2002	A. Purohit, J. Spadavecchia, C. Wright, and E. Zadok. Improving Application Performance Through System Call Composition. WIP presented at the *Annual USENIX Technical Conference*, Monterey, CA.

June 2002 — O. C. Leonard, J. Nieh, E. Zadok, J. Osborn, A. Shater, and C. Wright. The Design and Implementation of Elastic Quotas: A System for Flexible File System Management. WIP presented at the *Annual USENIX Technical Conference*, Monterey, CA.

January 2002 — J. Spadavecchia and E. Zadok. Enhancing NFS Cross-Administrative Domain Access. WIP presented at the *First USENIX Conference on File and Storage Technologies (FAST '02)*, Monterey, CA.

December 1999 — E. Zadok and J. Nieh. FiST: A Language for Stackable File Systems. Poster presented at the *17th ACM Symposium on Operating Systems Principles (SOSP)*, Kiawah Island, SC.

TEACHING

Instructor

Spring 2022 — Research in Computer Science (CSE-487), Computer Science Department, Stony Brook University

Spring 2022 — Advanced Systems Programming in Unix/C (CSE-376), Computer Science Department, Stony Brook University

Spring 2022 — Advanced Project in Computer Science I (CSE-523), Computer Science Department, Stony Brook University

Spring 2022 — Advanced Project in Computer Science I (CSE-524), Computer Science Department, Stony Brook University

Spring 2022 — Graduate Practicum in Teaching (CSE-698), Computer Science Department, Stony Brook University

Spring 2022 — Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University

Fall 2021 — Advanced Project in Computer Science I (CSE-523), Computer Science Department, Stony Brook University

Fall 2021 — Advanced Project in Computer Science II (CSE-524), Computer Science Department, Stony Brook University

Fall 2021 — Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University

Fall 2021 — M.S. Thesis Research (CSE-599), Computer Science Department, Stony Brook University

Fall 2021 — Seminar in Concurrency (CSE-643), Computer Science Department, Stony Brook University

Fall 2021 — Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University

Summer 2021 — M.S. Internship in Research (CSE-596), Computer Science Department, Stony Brook University

Summer 2021 — Internship in Research (CSE-696), Computer Science Department, Stony Brook University

Spring 2021	Advanced Systems Programming in Unix/C (CSE-376), Computer Science Department, Stony Brook University
Spring 2021	Advanced Project in Computer Science I (CSE-523), Computer Science Department, Stony Brook University
Spring 2021	Advanced Project in Computer Science I (CSE-524), Computer Science Department, Stony Brook University
Spring 2021	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Spring 2021	Graduate Practicum in Teaching (CSE-698), Computer Science Department, Stony Brook University
Spring 2021	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Fall 2020	Research in Computer Science (CSE-487), Computer Science Department, Stony Brook University
Fall 2020	Senior Honors Research Project II (CSE-499), Computer Science Department, Stony Brook University
Fall 2020	Advanced Project in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Fall 2020	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Fall 2020	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Summer 2020	Research in Computer Science (CSE-487), Computer Science Department, Stony Brook University
Summer 2020	Senior Honors Research Project I (CSE-495), Computer Science Department, Stony Brook University
Summer 2020	M.S. Internship in Research (CSE-596), Computer Science Department, Stony Brook University
Spring 2020	Advanced Systems Programming in Unix/C (CSE-376), Computer Science Department, Stony Brook University
Spring 2020	Advanced Project in Computer Science I (CSE-523), Computer Science Department, Stony Brook University
Spring 2020	Advanced Project in Computer Science I (CSE-524), Computer Science Department, Stony Brook University
Spring 2020	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Spring 2020	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University

Fall 2019	Advanced Project in Computer Science I (CSE-523), Computer Science Department, Stony Brook University
Fall 2019	Advanced Project in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Fall 2019	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Fall 2019	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Summer 2019	M.S. Internship in Research (CSE-596), Computer Science Department, Stony Brook University
Summer 2019	Internship in Research (CSE-696), Computer Science Department, Stony Brook University
Spring 2019	Graduate Level *Operating Systems* (CSE-506), Computer Science Department, Stony Brook University
Spring 2019	Advanced Project in Computer Science I (CSE-523), Computer Science Department, Stony Brook University
Spring 2019	Advanced Project in Computer Science I (CSE-524), Computer Science Department, Stony Brook University
Spring 2019	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Spring 2019	Graduate Practicum in Teaching (CSE-698), Computer Science Department, Stony Brook University
Winter 2019	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Fall 2018	Research in Computer Science (CSE-487), Computer Science Department, Stony Brook University
Fall 2018	Advanced Project in Computer Science I (CSE-523), Computer Science Department, Stony Brook University
Fall 2018	Advanced Project in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Fall 2018	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Fall 2018	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Summer 2018	Advanced Project in Computer Science I (CSE-523), Computer Science Department, Stony Brook University
Summer 2018	M.S. Internship in Research (CSE-596), Computer Science Department, Stony Brook University

Summer 2018	Internship in Research (CSE-696), Computer Science Department, Stony Brook University
Spring 2018	Research in Computer Science (CSE-487), Computer Science Department, Stony Brook University
Spring 2018	Graduate Level *Operating Systems* (CSE-506), Computer Science Department, Stony Brook University
Spring 2018	Advanced Project in Computer Science I (CSE-523), Computer Science Department, Stony Brook University
Spring 2018	Advanced Project in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Spring 2018	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Spring 2018	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Fall 2017	Advanced Project in Computer Science I (CSE-523), Computer Science Department, Stony Brook University
Fall 2017	Advanced Project in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Fall 2017	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Summer 2017	Advanced Project in Computer Science I (CSE-523), Computer Science Department, Stony Brook University
Summer 2017	M.S. Internship in Research (CSE-596), Computer Science Department, Stony Brook University
Summer 2017	Internship in Research (CSE-696), Computer Science Department, Stony Brook University
Spring 2017	Advanced Systems Programming in Unix/C (CSE-376), Computer Science Department, Stony Brook University
Spring 2017	Advanced Project in Computer Science I (CSE-523), Computer Science Department, Stony Brook University
Spring 2017	Advanced Project in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Spring 2017	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Fall 2016	Graduate Level *Operating Systems* (CSE-506), Computer Science Department, Stony Brook University
Fall 2016	Advanced Project in Computer Science I (CSE-523), Computer Science Department, Stony Brook University

Fall 2016	Advanced Project in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Fall 2016	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Fall 2016	M.S. Thesis Research (CSE-599), Computer Science Department, Stony Brook University
Fall 2016	Graduate Practicum in Teaching (CSE-698), Computer Science Department, Stony Brook University
Fall 2016	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Summer 2016	M.S. Internship in Research (CSE-596), Computer Science Department, Stony Brook University
Summer 2016	Internship in Research (CSE-696), Computer Science Department, Stony Brook University
Spring 2016	Advanced Systems Programming in Unix/C (CSE-376), Computer Science Department, Stony Brook University
Spring 2016	Research in Computer Science (CSE-487), Computer Science Department, Stony Brook University
Spring 2016	Introduction to Software Engineering and Project Plan (CSE-523), Computer Science Department, Stony Brook University
Spring 2016	Lab in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Spring 2016	M.S. Thesis Research (CSE-599), Computer Science Department, Stony Brook University
Spring 2016	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Fall 2015	Graduate Level *Operating Systems* (CSE-506), Computer Science Department, Stony Brook University
Fall 2015	Senior Honors Research Project II (CSE-496), Computer Science Department, Stony Brook University
Fall 2015	Introduction to Software Engineering and Project Plan (CSE-523), Computer Science Department, Stony Brook University
Fall 2015	Lab in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Fall 2015	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Fall 2015	Graduate Teaching Practicum (CSE-698), Computer Science Department, Stony Brook University

Fall 2015	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Summer 2015	M.S. Internship in Research (CSE-596), Computer Science Department, Stony Brook University
Summer 2015	Internship in Research (CSE-696), Computer Science Department, Stony Brook University
Spring 2015	Topics in Computer Science: Storage Systems (CSE-593), Computer Science Department, Stony Brook University
Spring 2015	Senior Honors Research Project I (CSE-495), Computer Science Department, Stony Brook University
Spring 2015	Introduction to Software Engineering and Project Plan (CSE-523), Computer Science Department, Stony Brook University
Spring 2015	Lab in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Spring 2015	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Spring 2015	M.S. Thesis Research (CSE-599), Computer Science Department, Stony Brook University
Spring 2015	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Fall 2014	Advanced Systems Programming in Unix/C (CSE-376), Computer Science Department, Stony Brook University
Fall 2014	Introduction to Software Engineering and Project Plan (CSE-523), Computer Science Department, Stony Brook University
Fall 2014	Lab in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Fall 2014	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Fall 2014	M.S. Thesis Research (CSE-599), Computer Science Department, Stony Brook University
Fall 2014	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Summer 2014	M.S. Internship in Research (CSE-596), Computer Science Department, Stony Brook University
Summer 2014	Internship in Research (CSE-696), Computer Science Department, Stony Brook University
Spring 2014	Graduate Level *Operating Systems* (CSE-506), Computer Science Department, Stony Brook University

Spring 2014	Introduction to Software Engineering and Project Plan (CSE-523), Computer Science Department, Stony Brook University
Spring 2014	Lab in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Spring 2014	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Spring 2014	Practicum in Teaching (CSE-698), Computer Science Department, Stony Brook University
Fall 2013	Advanced Systems Programming in Unix/C (CSE-376), Computer Science Department, Stony Brook University
Fall 2013	Introduction to Software Engineering and Project Plan (CSE-523), Computer Science Department, Stony Brook University
Fall 2013	Lab in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Fall 2013	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Fall 2013	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Summer 2013	M.S. Internship in Research (CSE-596), Computer Science Department, Stony Brook University
Summer 2013	Internship in Research (CSE-696), Computer Science Department, Stony Brook University
Summer 2013	FT Summer Research (CSE-800), Computer Science Department, Stony Brook University
Spring 2013	Graduate Level *Operating Systems* (CSE-506), Computer Science Department, Stony Brook University
Spring 2013	Introduction to Software Engineering and Project Plan (CSE-523), Computer Science Department, Stony Brook University
Spring 2013	Lab in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Spring 2013	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Spring 2013	Topics in Computer Science: Storage Systems (CSE-595), Computer Science Department, Stony Brook University
Spring 2013	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Fall 2012	*Operating Systems* (CSE-306), Computer Science Department, Stony Brook University
Fall 2012	Introduction to Software Engineering and Project Plan (CSE-523), Computer Science Department, Stony Brook University

Fall 2012	Lab in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Fall 2012	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Fall 2012	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Summer 2012	M.S. Internship in Research (CSE-596), Computer Science Department, Stony Brook University
Summer 2012	Internship in Research (CSE-696), Computer Science Department, Stony Brook University
Summer 2012	FT Summer Research (CSE-800), Computer Science Department, Stony Brook University
Spring 2012	Graduate Level *Operating Systems* (CSE-506), Computer Science Department, Stony Brook University
Spring 2012	Special Project in Computer Science (CSE-522), Computer Science Department, Stony Brook University
Spring 2012	Introduction to Software Engineering and Project Plan (CSE-523), Computer Science Department, Stony Brook University
Spring 2012	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Spring 2012	Topics in Computer Science: Storage Systems (CSE-595), Computer Science Department, Stony Brook University.
Spring 2012	M.S. Thesis Research (CSE-599), Computer Science Department, Stony Brook University
Spring 2012	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Fall 2011	Advanced Systems Programming in Unix/C (CSE-376), Computer Science Department, Stony Brook University
Fall 2011	Lab in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Fall 2011	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Fall 2011	M.S. Thesis Research (CSE-599), Computer Science Department, Stony Brook University
Fall 2011	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Summer 2011	Information Systems Internship (ISE-488), Computer Science Department, Stony Brook University

Summer 2011	M.S. Internship in Research (CSE-596), Computer Science Department, Stony Brook University
Summer 2011	Internship in Research (CSE-696), Computer Science Department, Stony Brook University
Spring 2011	Introduction to Software Engineering and Project Plan (CSE-523), Computer Science Department, Stony Brook University
Spring 2011	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Spring 2011	*M.S. Thesis Research* (CSE-599), Computer Science Department, Stony Brook University
Spring 2011	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Fall 2010	Graduate Level *Operating Systems* (CSE-506), Computer Science Department, Stony Brook University
Fall 2010	Lab in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Fall 2010	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Fall 2010	Practicum in Teaching (CSE-698), Computer Science Department, Stony Brook University
Fall 2010	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Summer 2010	M.S. Internship in Research (CSE-596), Computer Science Department, Stony Brook University
Summer 2010	Internship in Research (CSE-696), Computer Science Department, Stony Brook University
Summer 2010	FT Summer Research (CSE-800), Computer Science Department, Stony Brook University
Spring 2010	Systems Administration (CSE-311), Computer Science Department, Stony Brook University
Spring 2010	Systems Administration (ISE-311), Computer Science Department, Stony Brook University
Spring 2010	Introduction to Software Engineering and Project Plan (CSE-523), Computer Science Department, Stony Brook University
Spring 2010	*M.S. Thesis Research* (CSE-599), Computer Science Department, Stony Brook University
Spring 2010	Practicum in Teaching (CSE-698), Computer Science Department, Stony Brook University

Spring 2010	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Fall 2009	Graduate Level *Operating Systems* (CSE-506), Computer Science Department, Stony Brook University
Fall 2009	Lab in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Fall 2009	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Fall 2009	*M.S. Thesis Research* (CSE-599), Computer Science Department, Stony Brook University
Fall 2009	Practicum in Teaching (CSE-698), Computer Science Department, Stony Brook University
Fall 2009	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Spring 2009	Advanced Systems Programming in Unix/C (CSE-376), Computer Science Department, Stony Brook University
Spring 2009	Undergraduate Teaching Practicum (CSE-475), Computer Science Department, Stony Brook University
Spring 2009	Introduction to Software Engineering and Project Plan (CSE-523), Computer Science Department, Stony Brook University
Spring 2009	Lab in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Spring 2009	Proficiency Requirement in Computer Science (CSE-587), Computer Science Department, Stony Brook University
Spring 2009	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Spring 2009	M.S. Thesis Research (CSE-599), Computer Science Department, Stony Brook University
Spring 2009	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Fall 2008	Graduate Level *Operating Systems* (CSE-506), Computer Science Department, Stony Brook University
Fall 2008	Lab in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Fall 2008	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Fall 2008	M.S. Thesis Research (CSE-599), Computer Science Department, Stony Brook University

Fall 2008	Practicum in Teaching (CSE-698), Computer Science Department, Stony Brook University
Fall 2008	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Summer 2008	FT Summer Research (CSE-800), Computer Science Department, Stony Brook University
Spring 2008	Advanced Systems Programming in Unix/C (CSE-376), Computer Science Department, Stony Brook University
Spring 2008	Introduction to Software Engineering and Project Plan (CSE-523), Computer Science Department, Stony Brook University
Spring 2008	Lab in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Spring 2008	Proficiency Requirement in Computer Science (CSE-587), Computer Science Department, Stony Brook University
Spring 2008	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Spring 2008	Practicum in Teaching (CSE-698), Computer Science Department, Stony Brook University
Spring 2008	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Fall 2007	Graduate Level *Operating Systems* (CSE-506), Computer Science Department, Stony Brook University
Fall 2007	Introduction to Software Engineering and Project Plan (CSE-523), Computer Science Department, Stony Brook University
Fall 2007	Lab in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Fall 2007	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Fall 2007	Practicum in Teaching (CSE-698), Computer Science Department, Stony Brook University
Fall 2007	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Summer 2007	FT Summer Research (CSE-800), Computer Science Department, Stony Brook University
Spring 2007	Advanced Systems Programming in Unix/C (CSE-376), Computer Science Department, Stony Brook University
Spring 2007	Undergraduate Teaching Practicum (CSE-475), Computer Science Department, Stony Brook University

Spring 2007	Introduction to Software Engineering and Project Plan (CSE-523), Computer Science Department, Stony Brook University
Spring 2007	Lab in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Spring 2007	Topics in Computer Science: Secure Storage (CSE-590), Computer Science Department, Stony Brook University. Co-taught with Prof. Radu Sion.
Spring 2007	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Spring 2007	M.S. Thesis Research (CSE-599), Computer Science Department, Stony Brook University
Spring 2007	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Fall 2006	Graduate Level *Operating Systems* (CSE-506), Computer Science Department, Stony Brook University
Fall 2006	Introduction to Software Engineering and Project Plan (CSE-523), Computer Science Department, Stony Brook University
Fall 2006	Lab in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Fall 2006	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Fall 2006	M.S. Thesis Research (CSE-599), Computer Science Department, Stony Brook University
Fall 2006	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Summer 2007	FT Summer Research (CSE-800), Computer Science Department, Stony Brook University
Spring 2006	Advanced Systems Programming in Unix/C (CSE-376), Computer Science Department, Stony Brook University
Spring 2006	Topics in Computer Science: File Systems (CSE-594), Computer Science Department, Stony Brook University
Spring 2006	Introduction to Software Engineering and Project Plan (CSE-523), Computer Science Department, Stony Brook University
Spring 2006	Proficiency Requirement in Computer Science (CSE-587), Computer Science Department, Stony Brook University
Spring 2006	M.S. Thesis Research (CSE-599), Computer Science Department, Stony Brook University
Spring 2006	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University

Fall 2005	Graduate Level *Operating Systems* (CSE-506), Computer Science Department, Stony Brook University
Fall 2005	Lab in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Fall 2005	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Fall 2005	M.S. Thesis Research (CSE-599), Computer Science Department, Stony Brook University
Fall 2005	Practicum in Teaching (CSE-698), Computer Science Department, Stony Brook University
Fall 2005	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Spring 2005	Research in Computer Science (CSE-487), Computer Science Department, Stony Brook University
Summer 2005	Lab in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Summer 2005	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Summer 2005	FT Summer Research (CSE-800), Computer Science Department, Stony Brook University
Spring 2005	Advanced Systems Programming in Unix/C (CSE-376), Computer Science Department, Stony Brook University
Spring 2005	Advanced Operating Systems (CSE-624), Computer Science Department, Stony Brook University
Spring 2005	Research in Computer Science (CSE-487), Computer Science Department, Stony Brook University
Spring 2005	Introduction to Software Engineering and Project Plan (CSE-523), Computer Science Department, Stony Brook University
Spring 2005	Lab in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Spring 2005	Proficiency Requirement in Computer Science (CSE-587), Computer Science Department, Stony Brook University
Spring 2005	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Spring 2005	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Fall 2004	Graduate Level *Operating Systems* (CSE-506), Computer Science Department, Stony Brook University

Fall 2004	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Fall 2004	M.S. Thesis Research (CSE-599), Computer Science Department, Stony Brook University
Fall 2004	Practicum in Teaching (CSE-698), Computer Science Department, Stony Brook University
Fall 2004	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Spring 2004	Graduate Level *Operating Systems* (CSE-506), Computer Science Department, Stony Brook University
Spring 2004	Lab in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Spring 2004	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Spring 2004	M.S. Thesis Research (CSE-599), Computer Science Department, Stony Brook University
Spring 2004	Practicum in Teaching (CSE-698), Computer Science Department, Stony Brook University
Spring 2004	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Fall 2003	Advanced Systems Programming in Unix/C (CSE-376), Computer Science Department, Stony Brook University
Fall 2003	Introduction to Software Engineering and Project Plan (CSE-523), Computer Science Department, Stony Brook University
Fall 2003	Lab in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Fall 2003	Proficiency Requirement in Computer Science (CSE-587), Computer Science Department, Stony Brook University
Fall 2003	Independent Study in Computer Science (CSE-593), Computer Science Department, Stony Brook University
Fall 2003	M.S. Thesis Research (CSE-599), Computer Science Department, Stony Brook University
Summer 2003	Research in Computer Science (CSE-487), Computer Science Department, Stony Brook University
Summer 2003	Intro to Software Engineering (CSE-523), Computer Science Department, Stony Brook University
Spring 2003	Graduate Level *Operating Systems* (CSE-506), Computer Science Department, Stony Brook University

Spring 2003	Senior Honors Research Project II (CSE-496), Computer Science Department, Stony Brook University
Spring 2003	Intro to Software Engineering (CSE-523), Computer Science Department, Stony Brook University
Spring 2003	Lab in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Spring 2003	Independent Study in Computer Science (CSE-587), Computer Science Department, Stony Brook University
Spring 2003	M.S. Thesis Research (CSE-599), Computer Science Department, Stony Brook University
Spring 2003	Graduate Teaching Practicum (CSE-698), Computer Science Department, Stony Brook University
Fall 2002	Advanced Systems Programming in Unix/C (CSE-376), new course developed, Computer Science Department, Stony Brook University
Fall 2002	Senior Honors Research Project I (CSE-495), Computer Science Department, Stony Brook University
Fall 2002	Intro to Software Engineering (CSE-523), Computer Science Department, Stony Brook University
Fall 2002	Lab in Computer Science II (CSE-524), Computer Science Department, Stony Brook University
Fall 2002	Independent Study in Computer Science (CSE-587), Computer Science Department, Stony Brook University
Fall 2002	M.S. Thesis Research (CSE-599), Computer Science Department, Stony Brook University
Summer 2002	Intro to Software Engineering (CSE-523), Computer Science Department, Stony Brook University
Spring 2002	Graduate Level *Operating Systems* (CSE-506), Computer Science Department, Stony Brook University
Spring 2002	Undergraduate Teaching Practicum (CSE-475), Computer Science Department, Stony Brook University
Spring 2002	Research in Computer Science (CSE-487), Computer Science Department, Stony Brook University
Spring 2002	Intro to Software Engineering (CSE-523), Computer Science Department, Stony Brook University
Spring 2002	M.S. Thesis Research (CSE-599), Computer Science Department, Stony Brook University
Spring 2002	Internship in Research (CSE-698), Computer Science Department, Stony Brook University

Spring 2002	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Fall 2001	*Special Topics in Software Engineering: Systems Programming* (CSE/ISE-391), Computer Science Department, Stony Brook University
Fall 2001	M.S. Thesis Research (CSE-599), Computer Science Department, Stony Brook University
Fall 2001	Ph.D. Dissertation Research (CSE-699), Computer Science Department, Stony Brook University
Spring 2001	Graduate Level *Operating Systems* (CSE-506), Computer Science Department, Stony Brook University
Spring 2001	*Advanced File Systems*, seminar, Computer Science Department, Stony Brook University
November 1999	Tutorial, *Using Amd and the Automounter Utilities*, USENIX LISA 13, Seattle, Washington
October 1999	*NFS: Protocols, Programming and Implementation* (guest lecture in Internet Systems Programming), Computer Science Department, Columbia University
1997–1999	*Operating System Vulnerabilities* (guest lecture in Network Security), Computer Science Department, Columbia University
Fall 1995	*Data Structures in C*, Computer Science Department, Columbia University
1986	*Programming in BASIC*, A.A. Computers, Holon, Israel
1986	*Programming in Logo*, A.A. Computers, Holon, Israel

Teaching Assistant

Spring 1995	*Computer Security*, Computer Science Department, Columbia University
Fall 1994	*Operating Systems*, Computer Science Department, Columbia University
Spring 1992	*Object-Oriented Programming and C++*, Computer Science Department, Columbia University
Spring 1989	*Data Structures*, Computer Science Department, Columbia University
Fall 1989	*Software Design Lab,* Computer Science Department, Columbia University
Fall 1988	*Introduction to Programming in Pascal*, Computer Science Department, Columbia University

GRADUATE STUDENTS (PH.D.)

Ph.D. Students Graduated

May 2014–Jan 2019	Zhen Cao Dissertation title: A Practical Auto-Tuning Framework for Storage Systems Current Position: Google, Inc. (Mountain View, CA)
May 2012–Apr 2017	Ming Chen. Dissertation title: Kurma: Efficient and Secure Multi-Cloud Storage Gateways for Network-Attached Storage Current Position: Google, Inc. (New York, NY)

Sep 2009–May 2014	Zhichao Li. Dissertation title: GreenDM: A Versatile Tiering Hybrid Drive for the Trade-Off Evaluation of Performance, Energy, and Endurance Current Position: VMware, Inc. (Palo Alto, CA)
Jan 2008–Nov 2013	Vasily Tarasov (IBM Ph.D. Fellow). Dissertation title: Multi-dimensional Workload Analysis and Synthesis for Modern Storage Systems Current Position: IBM Research–Almaden, Inc. (San Jose, CA)
Jan 2006–Dec 2013	Justin Seyster (domestic) Dissertation title: Runtime Verification of Kernel-Level Concurrency Using Compiler-Based Instrumentation Current Position: Apple Computer, Inc. (Cupertino, CA)
Jan 2008–Feb 2012	Richard Spillane (domestic). Dissertation title: Efficient, Scalable, and Versatile Application and System Transaction Management for Direct Storage Layers. Current Position: Apple Computer, Inc. (Cupertino, CA)
Sep 2007–May 2011	Vinay Pai (domestic). Dissertation title: Incentive Mechanisms for Peer-to-Peer Streaming. Current Position: MuCash (New York, NY)
Sep 2003–Aug 2009	Sean Callanan (domestic). Dissertation title: Flexible Debugging with Controllable Overhead. Current Position: Apple Computer, Inc. (Cupertino, CA)
Sep 2003–Aug 2008	Avishay Traeger (domestic) (IBM Ph.D. Fellow). Dissertation title: Analyzing Root Causes of Latency Distributions. Current Position: Research Staff Member, IBM Haifa Research Labs (Tel-Aviv, Israel)
Sep 2003–Feb 2008	Gopalan Sivathanu. Dissertation title: End-to-End Abstractions for Application-Aware Storage. Current position: Google, Inc. (Mountain View, CA).
May 2003–Dec 2006	Nikolai Joukov. Dissertation title: Versatile, Portable, and Efficient File System Profiling. Current position: Research Staff Member, IBM T. J. Watson Research Center (Hawthorne, NY).
May 2003–May 2006	Charles P. Wright (domestic). Dissertation title: Extending ACID Semantics to the File System via `ptrace`. Current position: Research Staff Member, IBM T. J. Watson Research Center (Hawthorne, NY).

Current Ph.D. Students

Jan 2020–present	Yifei Liu
May 2019–present	Meyer “Alex” Merenstein
May 2018–present	Tyler Estro
Sep 2017–present	Ibrahim “Umit” Akgun

Past Ph.D. Students Advised

May 2019–Dec 2021	Wei Su
Nov 2014–Dec 2017	Sun (Jason) Zhen (visiting scholar from China)
Jun 2014–Dec 2016	Sonam Mandal
May 2004–Nov 2004	Anand Kashyap (He was Samir Das student. I advised him on one project and paper that came out of a class project.)
Sep 2001–May 2002	Li Aili (Student came to program interested in systems work, but became interested in graphics and switched areas altogether. Currently advised by Klaus Mueller.)
May 2001–May 2002	Joseph Spadavecchia (domestic) (I advised and funded student for one year. Student dropped out of program to go work and study in Scotland, for personal reasons. We co-authored one paper together.)
May 2002–Aug 2002	Zhenghong "Sam" Yang (Student already dropped from PhD to MS before I funded him under SPIR. Student graduated and now works for IBM.)
Sep 2001–May 2002	Hui Zhang (Student came to program interested in systems work, but became interested in hardware systems. Currently advised by Yuanyuan Yang.)

GRADUATE STUDENTS (M.S.)

Current M.S. Students

May 2022–present	Pradeep Nalluri
Jun 2021–present	Archit Saxena

Past M.S. Students

Jun 2021–May 2022	Manish Adkar
Jun 2021–May 2022	Nasratullah Sultany
Jun 2021–Jun 2022	Pei Liu
Jun 2021–Dec 2021	Rohan Chhabra
Jun 2021–May 2022	Rishabh Srivastava
Jun 2021–May 2022	Shushanth Madhubalan
Jun 2021–May 2022	Tejeshwar Gurram
Jan 2021–Dec 2021	Aakarsh Duvvuru
Sep 2020–Dec 2021	Aadil Shaikh
Jun 2020–May 2021	Dheeraj Ramchandani
Jan 2020–Dec 2020	Abhiraj Smit
Jan 2020–May 2021	Gomathi Ganesan
Jan 2020–Dec 2020	Vigneshwaran Mylsamy
Jan 2020–Dec 2020	Yatna Verma

Aug 2019–May 2020	Aneesh Joshi
Jun 2019–Dec 2019	Sagar Jeevan
Jan 2019–Dec 2019	Akshay Aurora
Jan 2019–Dec 2019	Jatin Sood
Jan 2019–Dec 2019	Krapi Ravindra Shah
Jan 2019–Dec 2019	Mehul Jain
Jan 2019–Dec 2019	Pragesh Jagnani
Jan 2019–Dec 2019	Shobhit Khandelwal
Sep 2018–Dec 2019	Aayush Sureka
Sep 2018–May 2019	Dhivahar Perumal
Sep 2018–Dec 2019	Ritika Nevatia
May 2018–Dec 2019	Nilesh Somani
Jan 2018–Dec 2019	Prafful Agarwal
Jan 2018–Dec 2018	Ankit Aggarwal
Jan 2018–Dec 2018	Dhanashri Patil
Jan 2018–Dec 2018	Manu Mathew
Jan 2018–Dec 2018	Prateek Roy
Jan 2018–Dec 2018	Rahul Rane
Jan 2018–Dec 2018	Rohit Kumar
Jan 2018–Dec 2018	Saish Sali
Jan 2018–Dec 2018	Siddesh Shinde
Jan 2018–Dec 2018	Vineeth Ramesh
Aug 2017–Dec 2018	Noopur Anil Maheshwari
Sep 2017–Apr 2018	Tyler Estro
Jun 2017–May 2018	Mayur Jadhav
May 2017–May 2018	Maryia Maskaliova
Jan 2017–Dec 2017	Darshan Godhia
Jan 2017–Dec 2017	Farhaan Jalia
Jan 2017–Dec 2017	Geetika Babu Bangera
Jan 2017–Dec 2017	Kunal Shah
Jan 2017–Dec 2017	Nehil Shah
Jan 2017–Dec 2017	Nidhi Panpalia

Jan 2017–Dec 2017	Rushabh Shah
Jan 2017–Dec 2017	Sagar Shah
Jan 2017–Dec 2017	Swaminathan Sivaraman
Jan 2017–Dec 2017	Tushar Jain
Sep 2016–Dec 2017	Vinothkumar Raja
Aug 2016–Dec 2017	Arun Ramachandran
Aug 2016–Dec 2017	Mukul Sharma
Aug 2016–Dec 2017	Sachin Tiwari
Aug 2016–Dec 2017	Shivanshu Goswami
Aug 2016–Dec 2016	Vishal Sahu
Apr 2016–May 2017	Leixiang Wu
Jan 2016–Dec 2016	Praveen Kumar Morampudi
Jan 2016–Dec 2016	Vithiya Muthukumar
Jan 2016–Dec 2016	Harshkumar Patel
Jan 2016–Dec 2016	Deepika Peringanji
Jan 2016–Dec 2016	Venkatakrishnan Rajagopalan
Jan 2016–Dec 2016	Vishnu Vardhan Rajula
Jan 2016–Dec 2016	Hari Prasath Raman
Sep 2015–Dec 2016	Shubhi Rani
Jan 2016–Dec 2016	Jasmit Saluja
Sep 2015–Dec 2016	Ashok Sankar Harihara Subramony
Aug 2015–Dec 2016	Bharath Kumar Reddy Vangoor
Sep 2014–May 2016	Dongju Ok
Mar 2015–May 2016	Garima Gehlot
Sep 2015–Dec 2015	Vivek Tiwari
May 2014–Dec 2015	Kumar Sourav
Sep 2014–Dec 2015	Bharat Singh
Sep 2014–Dec 2015	Varun Shastry
May 2014–Dec 2015	Abhishek Gupta
Sep 2014–Dec 2015	Arvind Chaudhary
Sep 2014–Dec 2015	Aashray Arora
Sep 2013–May 2015	Arun Olappamanna Vasudevan

Jan 2014–May 2015	Akhilesh Chaganti
May 2014–May 2015	Udit Kaushik Chitalia
Apr 2014–May 2015	Kelong Wang
Sep 2013–Dec 2014	Soujanya Shankaranarayana
May 2014–Dec 2014	Li Mengyang
Sep 2013–May 2014	Amanpreet Mukker
Jun 2013–May 2014	Sonam Mandal
Sep 2012–Dec 2013	Deepak Jain
Sep 2012–Dec 2013	Sagar Trehan
Sep 2012–Dec 2013	Madhu Srikar Palmur
Sep 2012–May 2013	Benixon Arul Dhas
May 2012–Jun 2013	Karthikeyani Palanisami
Jan 2012–Dec 2012	Binesh Andrews
Jan 2012–Dec 2012	Atul Karmarkar
Sep 2011–Dec 2012	Mandar S. Joshi
Sep 2011–Dec 2012	Ravikant R. Malpani
Sep 2011–May 2012	Rajesh Aavuty
Jan 2011–May 2012	Amar Mudrankit
Jan 2011–Dec 2011	Siddhi Tadpatrikar
Sep 2010–Dec 2011	Samriti Katoch
Sep 2010–Dec 2011	Koundinya Santhosh Kumar
Aug 2010–May 2012	Pradeep Shetty
Jan 2011–May 2011	Koundinya Muppalla
Jan 2010–Dec 2010	Prabakar Radhakrishnan
Jan 2010–Dec 2010	Gyumin Sim
Sep 2009–Dec 2010	Shrikar Archak
Sep 2009–Dec 2010	Saumitra Bhanage
Sep 2009–Dec 2010	Sagar Dixit
Jan 2009–May 2010	Priya Sehgal
Jan 2009–Dec 2009	Karthikeyan Srinaivasan
Sep 2008–May 2010	Abhinav Duggal
Sep 2008–Dec 2009	Sujay Godbole

Jan 2008–Dec 2008	Yamini Allu
Jan 2008–Dec 2008	Manjunath Chinni
Jan 2008–Dec 2008	Sachin Gaikwad
Sep 2007–Jun 2009	Daniel J. Dean (domestic)
Sep 2007–Dec 2008	Himanshu Kanda
Sep 2007–May 2009	Rachita Kothiyal
Sep 2007–Dec 2008	Chaitanya Yalamanchili
Jan 2007–Dec 2007	Kimberly Albrecht (domestic, SFS scholar)
Jan 2007–Dec 2007	Ramya Edara
Jan 2007–Dec 2007	Ivan Deras Tabora
Jan 2007–Dec 2007	Kumar Thangavelu
Sep 2006–Dec 2007	Gopala Suryanarayana
Sep 2006–Dec 2007	Kiron Vijayasankar
Jun 2006–Dec 2007	Richard Spillane (domestic)
Jan 2006–August 2007	Swaminathan Sundararaman
Sep 2006–June 2007	Yiannis Pericleous
Sep 2005–May 2007	David Quigley (domestic, SFS scholar)
Jan 2006–March 2007	Harry Papaxenopoulos (domestic, SFS scholar)
Jan 2006–Dec 2006	Chaitanya Patti
Jan 2005–Dec 2005	Dan Ottavio (domestic)
Jan 2005–Dec 2005	Arun Kumar
Sep 2004–Dec 2005	Naveen Gupta
Sep 2004–Dec 2005	Rakesh Narayan Iyer
Sep 2004–Dec 2005	Sunil Satnur
May 2004–Nov 2004	Swapnil V. Patil
Jan 2004–Dec 2004	Mohammad Nayyer Zubair
Sep 2003–Aug 2005	Abhishek Rai
Sep 2003–Dec 2004	Aditya Kashyap
Sep 2003–Dec 2004	Devaki Pandurang Kulkarni
May 2003–Aug 2002	Akshat Khandelwal
May 2003–Aug 2002	Deepak Rao
May 2003–Aug 2004	Akshat Aranya

May 2003–Dec 2003	Alexander Butler (domestic)
May 2003–Dec 2003	Jay Pradip Dave
May 2003–Dec 2003	Mohan-Krishna Channa-Reddy
May 2003–Dec 2003	Salil Gokhale
May 2003–Jan 2004	Abhijith Das
May 2003–Dec 2004	Miretskiy “Eugene” Yevgeniy (domestic)
Jan 2003–Dec 2003	Harikesavan Pathangi Krishnan
Jan 2003–Dec 2003	Puja Gupta
May 2002–Dec 2002	Delia Paval
May 2002–Dec 2002	Nitin Khosla
May 2002–Dec 2002	Rongqing “Frank” Tu
May 2002–May 2003	Jeffrey R. Osborn (domestic)
May 2002–May 2003	Michael Martino (domestic)
May 2002–May 2003	Zhenghong “Sam” Yang
Sep 2002–Dec 2002	Nishant Nagalia
Sep 2002–Dec 2002	Zhou Zhang
Sep 2002–Dec 2003	Swaroop Karunakara
Sep 2002–May 2003	Ana Centeno
Sep 2002–May 2003	Andrew Himmer (domestic)
Sep 2002–May 2003	Sheshadri Sreenath
Mar 2002–Apr 2003	Manish Prasad
Jan 2002–May 2004	Kiran-Kumar Muniswamy-Reddy
Sep 2001–May 2003	Amit Purohit

UNDERGRADUATE STUDENTS

Jun 2021–present	Michael Arkhangelskiy
Jun 2021–present	Andrew Burford
Jun 2021–present	Michael McNeill

Past Undergraduate Students

Jun 2021–Jun 2022	Pei Liu (domestic, BS program, CSE program)
Jun 2020–Dec 2020	Haolin Yu (BS program, CSE program)
Jun 2020–May 2021	Haewon Julie Lee (BS program, CSE program)
Jun 2020–May 2021	Yang Yang (domestic, BS program, CSE program)

Aug 2019–May 2020	Yinuo Zhang (BS program, CSE program)
Jul 2019–Dec 2019	Abraham Spitalny, (domestic, BS program, CSE program)
May 2018–Dec 2018	Amrith Arunachalam (domestic, BS program, CSE program)
May 2017–May 2018	Kevin Sun (domestic, BS program, CSE program)
Feb 2015–Mar 2016	Leixiang Wu (domestic, BS program, CSE program)
July 2012–Dec 2012	Edgardo Linero, (domestic, BS program, CSE program)
May 2011–May 2012	Ryan Matthew Walsh, (domestic, BS program, ISE program)
May 2010–Aug 2011	Christopher Jutting, (domestic, BS program, ISE program)
Jan 2009–July 2009	Ryan Leif Walsh, (domestic, BS/MS program, Honors program)
Jan 2004–2009	Adam David Alan Martin (domestic)
Sep 2005–May 2008	Josef "Jeff" Sipek (domestic)
Jun 2006–August 2007	Daniel J. Dean (domestic)
Jan 2005–May 2006	Richard Spillane (domestic, BS/MS program, Honors program, Stony Brook URECA summer internship)
Jan 2005–Jul 2005	Tim Wong (domestic)
Jan 2005–Aug 2005	David Quigley (domestic)
May 2003–Dec 2003	Mohammad Nayyer Zubair (Honors program, Stony Brook URECA summer internship)
Jan 2002–May 2002	Evan Chan (domestic)
Jan 2002–May 2002	Reynold Mercado (domestic)
Jan 2002–May 2002	Jeffrey Osborn (domestic)
Jan 2002–May 2002	Julio Salazar (domestic)
Jan 2002–May 2002	Ariye Shater (domestic)
Jan 2002–May 2003	Charles P. Wright (Honors program advisor, domestic)
Jan 2002–May 2002	Yuri Yanpolski (domestic)

HIGH-SCHOOL STUDENTS

Past High-School Students

Sep 2015–Aug 2017	Henry Nelson (domestic), "NFSv4 Compounds (vNFS)" project
June 2005–August 2005	Jordan Hoch (domestic), Simons summer internship, project title "Visual Debugging of Operating Systems State.")

Stony Brook, New York, June 3, 2025