

UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEAL BOARD

AMAZON.COM, INC., AMAZON WEB SERVICES, INC., and AMAZON.COM
SERVICES LLC,

Petitioner

v.

B.S.D. CROWN, LTD.,

Patent Owner.

IPR2025-00057

U.S. Patent No. 8,934,887

DECLARATION OF DR. PAUL D. MARTIN

Declaration

I declare that all statements made herein on my own knowledge are true and that all statements made on information and belief are believed to be true, and further, 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 Title 18 of the United States Code.

By:  _____

Paul D. Martin, Ph.D.

TABLE OF CONTENTS

I.	PROFESSIONAL BACKGROUND.....	5
	A. Teaching	6
	B. Research	7
	C. Patents.....	12
	D. Industry	12
II.	TECHNOLOGY BACKGROUND.....	15
	A. Remote Access Technology	15
	B. Cloud Virtualization Technology.....	19
	1. Virtualization Technology	19
	2. Cloud Virtualization.....	23
	3. Cloud-Based Streaming	25
	C. Mobile Cloud Computing.....	27
III.	RELEVANT LEGAL STANDARDS	30
IV.	PERSON OF ORDINARY SKILL IN THE ART (“POSITA”).....	32
V.	’887 PATENT OVERVIEW	33
	A. Specification.....	33
	B. Prosecution History	36
VI.	CLAIM CONSTRUCTION	39
VII.	DETAILED EXPLANATION OF GROUNDS.....	39
	A. Ground 1A: Yoon.....	39
	1. Yoon.....	39
	2. Claim 1	43
	3. Claim 2: The method of claim 1, further comprising: emulating, by the server, mobile device HW and executing the mobile device OS on the emulated mobile device HW.....	52
	4. Claim 3	53

TABLE OF CONTENTS (continued)

	Page
5. Claim 5: The method of claim 1, further comprising: receiving, by the server, the input information from a plurality of different mobile devices, wherein a same instance of the mobile device OS processes the input information from each of the plurality of different mobile devices to produce the stream of user interface information.	54
6. Claim 20	55
7. Claim 21	57
8. Claim 28	57
B. Ground 1B: Yoon + Nix.....	62
1. Nix.....	62
2. Motivation to Combine	63
3. Claim 4: The method of claim 1, further comprising: providing, by the server, a mobile device gateway for handling and providing internet access, voice calls and a gateway to a public switched telephone network (PSTN).	64
4. Claim 14: The method of claim 11, further comprising: providing, by the server, a mobile device gateway for handling and providing internet access, voice calls and a gateway to a public switched telephone network (PSTN).	66
5. Claim 22: The method of claim 20, further comprising: providing, by the server, a mobile device gateway for handling and providing internet access, voice calls and a gateway to a public switched telephone network (PSTN).	66
C. Ground 1C: Yoon + Lubonski + Richardson.....	66
1. Lubonski.....	66
2. Richardson	70

TABLE OF CONTENTS
(continued)

	Page
3. Motivation to Combine	71
4. Claim 6: The method of claim 1, wherein the stream of user interface information comprises a plurality of different types of data and the server sends each different type of data to the mobile device in a separate data stream.....	75
5. Claim 7: The method of claim 6, wherein at least one of said separate data streams comprises a quality of service different from a quality of service of another of said separate data streams.	77
6. Claim 8: The method of claim 6, wherein at least one of said separate data streams comprises an encoding method and a bandwidth different from an encoding method and a bandwidth of another of said separate data streams.	78
7. Claim 11	79
8. Claim 12: The method of claim 11, further comprising: emulating, by the server, mobile device HW and executing the mobile device OS on the emulated mobile device HW.....	81
9. Claim 13	81
10. Claim 15: The method of claim 11, further comprising: receiving the input information from a plurality of different mobile devices, wherein a same instance of the mobile device OS processes the input information from the plurality of different mobile devices to produce the stream of user interface information.....	81
11. Claim 16: The method of claim 11, wherein at least one of said separate data streams comprises a quality of service different from a quality of service of another of said separate data streams.	82

TABLE OF CONTENTS **(continued)**

	Page
12. Claim 17: The method of claim 11, wherein at least one of said separate data streams comprises an encoding method and a bandwidth different from an encoding method and a bandwidth of another of said separate data streams.	82
13. Claim 23: The method of claim 20, wherein the stream of user interface information comprises a plurality of different types of data and the server sends each different type of data to the mobile device in a separate data stream.....	82
14. Claim 24: The method of claim 23, wherein at least one of said separate data streams comprises a quality of service different a quality of service of another of said separate data streams.....	82
15. Claim 25: The method of claim 23, wherein at least one of said separate data streams comprises an encoding method and a bandwidth different from an encoding method and a bandwidth of another of said separate data streams.	83
D. Ground 2A: Overton.....	83
1. Overton.....	83
2. Claim 1	85
3. Claim 2: The method of claim 1, further comprising: emulating, by the server, mobile device HW and executing the mobile device OS on the emulated mobile device HW.....	95
4. Claim 3	95
5. Claim 5: The method of claim 1, further comprising: receiving, by the server, the input information from a plurality of different mobile devices, wherein a same instance of the mobile device OS processes the input information from	

TABLE OF CONTENTS (continued)

	Page
each of the plurality of different mobile devices to produce the stream of user interface information.	96
6. Claim 20	97
7. Claim 21	99
8. Claim 28	99
E. Ground 2B: Overton + Nix.....	106
1. Motivation to Combine	106
2. Claim 4: The method of claim 1, further comprising: providing, by the server, a mobile device gateway for handling and providing internet access, voice calls and a gateway to a public switched telephone network (PSTN).	110
3. Claim 14: The method of claim 11, further comprising: providing, by the server, a mobile device gateway for handling and providing internet access, voice calls and a gateway to a public switched telephone network (PSTN).	112
4. Claim 22: The method of claim 20, further comprising: providing, by the server, a mobile device gateway for handling and providing internet access, voice calls and a gateway to a public switched telephone network (PSTN).	112
F. Ground 2C: Overton + Lubonski + Richardson.....	113
1. Motivation to Combine	113
2. Claim 6: The method of claim 1, wherein the stream of user interface information comprises a plurality of different types of data and the server sends each different type of data to the mobile device in a separate data stream.....	116
3. Claim 7: The method of claim 6, wherein at least one of said separate data streams comprises a	

TABLE OF CONTENTS **(continued)**

	Page
quality of service different from a quality of service of another of said separate data streams.	118
4. Claim 8: The method of claim 6, wherein at least one of said separate data streams comprises an encoding method and a bandwidth different from an encoding method and a bandwidth of another of said separate data streams.	118
5. Claim 11	119
6. Claim 12: The method of claim 11, further comprising: emulating, by the server, mobile device HW and executing the mobile device OS on the emulated mobile device HW.....	121
7. Claim 13	121
8. Claim 15: The method of claim 11, further comprising: receiving the input information from a plurality of different mobile devices, wherein a same instance of the mobile device OS processes the input information from the plurality of different mobile devices to produce the stream of user interface information.....	121
9. Claim 16: The method of claim 11, wherein at least one of said separate data streams comprises a quality of service different from a quality of service of another of said separate data streams.	122
10. Claim 17: The method of claim 11, wherein at least one of said separate data streams comprises an encoding method and a bandwidth different from an encoding method and a bandwidth of another of said separate data streams.	122
11. Claim 23: The method of claim 20, wherein the stream of user interface information comprises a plurality of different types of data and the server	

TABLE OF CONTENTS **(continued)**

	Page
sends each different type of data to the mobile device in a separate data stream.....	122
12. Claim 24: The method of claim 23, wherein at least one of said separate data streams comprises a quality of service different a quality of service of another of said separate data streams.....	122
13. Claim 25: The method of claim 23, wherein at least one of said separate data streams comprises an encoding method and a bandwidth different from an encoding method and a bandwidth of another of said separate data streams.	123
G. Ground 3A: Lu	123
1. Lu	123
2. Claim 1	127
3. Claim 2: The method of claim 1, further comprising: emulating, by the server, mobile device HW and executing the mobile device OS on the emulated mobile device HW.....	133
4. Claim 3	134
5. Claim 5: The method of claim 1, further comprising: receiving, by the server, the input information from a plurality of different mobile devices, wherein a same instance of the mobile device OS processes the input information from each of the plurality of different mobile devices to produce the stream of user interface information.	135
6. Claim 20.....	136
7. Claim 21	137
H. Ground 3B: Lu + Lubonski + Richardson	138
1. Motivation to Combine	138

TABLE OF CONTENTS (continued)

	Page
2. Claim 6: The method of claim 1, wherein the stream of user interface information comprises a plurality of different types of data and the server sends each different type of data to the mobile device in a separate data stream.....	140
3. Claim 7: The method of claim 6, wherein at least one of said separate data streams comprises a quality of service different from a quality of service of another of said separate data streams.	141
4. Claim 8: The method of claim 6, wherein at least one of said separate data streams comprises an encoding method and a bandwidth different from an encoding method and a bandwidth of another of said separate data streams.	142
5. Claim 11	143
6. Claim 12: The method of claim 11, further comprising: emulating, by the server, mobile device HW and executing the mobile device OS on the emulated mobile device HW.....	144
7. Claim 13	144
8. Claim 15: The method of claim 11, further comprising: receiving the input information from a plurality of different mobile devices, wherein a same instance of the mobile device OS processes the input information from the plurality of different mobile devices to produce the stream of user interface information.....	145
9. Claim 16: The method of claim 11, wherein at least one of said separate data streams comprises a quality of service different from a quality of service of another of said separate data streams.	145
10. Claim 17: The method of claim 11, wherein at least one of said separate data streams comprises an	

TABLE OF CONTENTS
(continued)

	Page
encoding method and a bandwidth different from an encoding method and a bandwidth of another of said separate data streams.	145
11. Claim 23: The method of claim 20, wherein the stream of user interface information comprises a plurality of different types of data and the server sends each different type of data to the mobile device in a separate data stream.....	146
12. Claim 24: The method of claim 23, wherein at least one of said separate data streams comprises a quality of service different a quality of service of another of said separate data streams.....	146
13. Claim 25: The method of claim 23, wherein at least one of said separate data streams comprises an encoding method and a bandwidth different from an encoding method and a bandwidth of another of said separate data streams.	146

1. I, Dr. Paul D. Martin of Potomac, Maryland, do hereby declare:

2. I am making this declaration at the request of Amazon.com, Inc., Amazon Web Services, Inc., and Amazon.com Services LLC (“Amazon” or “Petitioner”) in the matter of *Amazon.com, Inc. et al v. BSD Crown, Ltd.*, IPR2025-00057.

3. In preparing this Declaration, I considered the following materials:

- U.S. Patent No. 8,934,887 (“’887 Patent”) (EX1001);
- File History of the ’887 Patent (“’887 FH”) (EX1002);
- Korean Patent Application Publication No. 10-2012-0046807 (“Yoon”) (EX1005);
- U.S. Patent No. 9,424,052 (“Overton”) (EX1006);
- U.S. Patent Application Publication No. 2009/0305790 A1 (“Lu”) (EX1007);
- M. Lubonski, V. Gay and A. Simmonds, “An adaptation architecture to improve QoS of multimedia services for enterprise remote desktop protocols,” Next Generation Internet Networks, 2005, Rome, Italy, 2005, pp. 149-156 (Lubonski”) (EX1008);
- Richardson, Iain E., The H.264 Advanced Video Compression Standard (2nd ed.), 2010, Wiley Publishing (“Richardson”) (EX1009);
- U.S. Patent Application Publication No. 2006/0098619 A1 (“Nix”) (EX1010);

- U.S. Patent Application Publication No. 2001/0052012 A1 (“Rinne”) (EX1015);
- M. A. Talaat, M. A. Koutb, H. M. Kelash and R. H. Aboelez, “Content-aware adaptive video streaming system,” 2005 International Conference on Information and Communication Technology, Cairo, 2005, pp. 265-276 (“Talaat”) (EX1016);
- A Few More Reasons Why RDP is Insecure.
<https://thehackernews.com/2023/07/a-few-more-reasons-why-rdp-is-insecure.html> (EX1017);
- Understanding the Remote Desktop Protocol (RDP).
<https://learn.microsoft.com/en-us/troubleshoot/windows-server/remote/understanding-remote-desktop-protocol> (EX1018);
- Remote Desktop Services overview in Windows Server.
<https://learn.microsoft.com/en-us/windows-server/remote/remote-desktop-services/remote-desktop-services-overview> (EX1019);
- Remote Desktop clients for Remote Desktop Services and remote PCs.
<https://learn.microsoft.com/en-us/windows-server/remote/remote-desktop-services/clients/remote-desktop-clients> (EX1020);

- Remote takeover: How RealVNC conquered the world.
<https://www.cam.ac.uk/research/features/remote-takeover-how-realvnc-conquered-the-world> (EX1021);
- Download VNC Server by RealVNC.
<https://www.realvnc.com/en/connect/download/vnc/> (EX1022);
- Download VNC Viewer by RealVNC.
<https://www.realvnc.com/en/connect/download/viewer/> (EX1023);
- Robert. P. Goldberg. Survey of virtual machine research. IEEE Computer Magazine, 7(6):34-45, 1974.
<https://web.eecs.umich.edu/~prabal/teaching/eecs582-w11/readings/Gol74.pdf> (EX1024);
- R. J. Creasy. The Origin of the VM/370 Time-Sharing System. IBM Journal of Research and Development, 25(5):483-490, September 1981.
https://pages.cs.wisc.edu/~stjones/proj/vm_reading/ibmrd2505M.pdf (EX1025);
- VMware Celebrates Seven Years of Continual Innovation and Execution.
<https://news.broadcom.com/releases/7year> (EX1026);
- [Xen-devel] [ANNOUNCE] Xen high-performance x86 virtualization.
<https://sourceforge.net/p/xen/mailman/message/5533663/> (EX1027);

- News (older entries). https://www.virtualbox.org/wiki/News_pre_4_3 (EX1028);
- Ten years of KVM. <https://lwn.net/Articles/705160/> (EX1029);
- Hyper-V Made Available. <https://adtmag.com/articles/2008/06/26/hyperv-made-available.aspx> (EX1030);
- Amazon EC2 beta. https://aws.amazon.com/blogs/aws/amazon_ec2_beta/ (EX1031);
- Microsoft launches Windows Azure. <https://news.microsoft.com/announcement/microsoft-launches-windows-azure/> (EX1032);
- Reflecting on our ten year App Engine journey. <https://cloud.google.com/blog/products/gcp/reflecting-on-our-ten-year-app-engine-journey/> (EX1033);
- Guide to Security for Full Virtualization Technologies. <https://doi.org/10.6028/NIST.SP.800-125> (EX1034);
- Review: OnLive -- The What, Why and Who of Gaming in the Cloud. <https://www.wired.com/2011/06/review-onlive-the-what-why-and-who-of-gaming-in-the-cloud/> (EX1035);
- Gaikai is Live! https://dperry.com/2011/02/06/gaikai_is_live/ (EX1036);

- Qi, H., & Gani, A. (2012, May). Research on mobile cloud computing: Review, trend and perspectives. In 2012 second international conference on digital information and communication technology and its applications (DICTAP) (pp. 195-202). IEEE. <https://arxiv.org/pdf/1206.1118> (EX1037);
- Kumar, Karthik, and Yung-Hsiang Lu. Cloud computing for mobile users: Can offloading computation save energy?. Computer 43, no. 4 (2010): 51-56. <https://www.cs.purdue.edu/homes/bb/cs590/handouts/computer.pdf> (EX1038);
- Dinh, H.T., Lee, C., Niyato, D. and Wang, P. (2013), A survey of mobile cloud computing: architecture, applications, and approaches. Wirel. Commun. Mob. Comput., 13: 1587-1611. <https://doi.org/10.1002/wcm.1203> (EX1039);

and any other materials cited herein.

I. Professional Background

4. I am the Chief Scientist at Harbor Experts, Inc. and a part-time Lecturer in the Department of Computer Science at Johns Hopkins University. I hold B.S., M.S.E., and Ph.D. degrees from Johns Hopkins University, with all degrees, including my doctorate, being in computer science. My current curriculum vitae (CV) is attached to this my opening report as Exhibit 1004.

5. I am an expert in the technical subject matter areas relevant to this case. All opinions and facts stated in this report are true and correct to the best of my

knowledge. If called upon to testify, I could and would testify to the truth of the following. The company for which I work, Harbor Experts, is being compensated at the rate of \$675 per hour for time that I spend on this case. Harbor Experts is also being reimbursed for my reasonable and customary expenses. My compensation is not dependent in any way on the results of the lawsuit or the substance of my testimony.

A. Teaching

6. I am a part-time lecturer in the Department of Computer Science at Johns Hopkins University. I currently teach classes on computer and network security and applied cryptography. In the past I have taught classes on the topic of hardware hacking, including analyzing, modifying, and repairing computer hardware and embedded devices. I also occasionally substitute teach classes on network security and cryptography.

7. My classes cover a diverse array of topics including hardware and software security design, analysis and reverse engineering for vulnerability assessment; hardware and software-based attacks on computer components including RAM and CPUs; applied cryptography; computer architecture; computer networking; and component-level repair and modification. My classes also utilize hardware and software emulation and virtualization techniques in order to present

students with a variety of real-world environments in which to learn and perform hands-on projects.

B. Research

8. In spring 2011, I completed my bachelor's in computer science at Johns Hopkins University. In fall 2011, I began working towards my Ph.D. in computer science, also at Johns Hopkins University, and I began researching computer security and privacy systems in order to both design novel technologies for securing computing systems and to also bridge the interface gap between users and their technology. As such, my research focused not only on developing technical solutions to novel security and privacy systems but to also presenting these solutions in a way that non-technical users could understand. Part of my research also focused on non-deterministically generating virtualized computer networks in order to provide malware education environments for students. I also developed an emulator for analyzing malware code segments as part of a student project with the NSA. This emulator was integrated into an IDA Pro plugin and was able to observe how binary instruction snippets taken from malware samples modified their computing environment.

9. During my tenure as a Ph.D. student at Johns Hopkins, I was fully funded as a research assistant and/or teaching assistant throughout all of my semesters. I was a member of the Upsilon Pi Epsilon computer science honor

society, as well as its treasurer for the 2014-2015 school year. I was also a member of the Johns Hopkins chapter of the Association for Computing Machinery. I earned an award for outstanding teaching assistant in the Computer Science department for the year 2014-2015. I co-instructed a short course called, “Introduction to Hardware Hacking,” with a colleague, Dr. Michael Rushanan, which was the highest-rated course in the computer science department (based on student reviews) during the winter session in which it was offered. In this course, we offered lessons on a variety of topics including modifying game consoles and device firmware; electronics repair; binary analysis and modification, network traffic analysis; and web-based vulnerability assessment and exploitation.

10. I represented Johns Hopkins University as a student speaker at numerous open houses and admitted student days to help introduce prospective students to the computer science department. I was selected to speak to the external advisory board for the computer science department to provide insight into the student experience in the department. I served the department as the faculty liaison czar for several semesters, in which I attended and documented faculty meetings, which I then summarized for the students in the department. I also mentored and supervised a high school student on a year-long student project in software design and I supervised a variety of undergraduate research assistants on some of my research projects.

11. I finished my Ph.D. by successfully defending my dissertation, entitled “Securing Medical Devices and Protecting Patient Privacy in the Technological Age of Healthcare,” on February 12, 2016, at the age of 26. My doctoral thesis tells the story of a secure healthcare practice of the future in which technology is seamless to use for healthcare providers while providing a much higher standard of security than is typical today. It focuses on the juxtaposition of usability and security with an emphasis on simplicity, automation, and error-proofing of security controls.

12. During my doctorate and afterward, I have produced, peer-reviewed, and published research in areas such as fingerprinting, anomaly detection, multifactor authentication, and embedded systems security. In many of these cases, I focused not only on designing novel security systems but also on designing usable web-based security systems for controlling them and understanding their data output. In fact, some of my research has formed the basis for commercial products and services. For example, in my work on integrated audit and access control, which was funded in part by Accenture Labs, I developed a Hadoop-based application to perform large-scale statistical analysis of audit logs from an electronic medical record system. As part of this work, I also designed a web application to automatically produce human-readable reports and graphs for use in consulting. Accenture subsequently patented this technology.

13. Similarly, in my research at Applied Communication Sciences in 2013, I designed and implemented a web-based traffic visualization dashboard and analysis system for field area smart grid networks that could be used to quickly gain an understanding of the current state of a SmartGrid network as well as to detect unexpected anomalies in the network. Applied Communication Sciences subsequently patented this work and continued to build on the project. To my knowledge, they sold and/or still sell this product as part of their SecureSmart Managed Security Service product offering.

14. I have worked on and published two research projects related to improving the quality and ease-of-use of authentication technologies, especially in healthcare settings. In one project, I and my co-authors designed an indoor location tracking system consisting of unspoofable Bluetooth Low Energy beacons. These beacons were placed around a building and mapped to a backend such that a user could report which beacons he or she was within range of in order to be used as a secondary authentication mechanism for accessing patient medical records. In the way that we designed and envisioned the system, a doctor would log into a mobile device which would connect to a web-based backend. As the doctor walked past patient rooms, he or she would be automatically presented with medical records of nearby patients. In another project, I and my co-authors designed an authentication bracelet that would receive a Kerberos ticket upon login to a modified computer

terminal through use of low-energy electrical signals transmitted over the wearer's skin. Upon use of other terminals throughout the hospital, this Kerberos ticket could be sent back over the wearer's skin to the custom contact on the terminal in order to allow the user to login without a password. The bracelet was also designed to immediately lose the cryptographic secret upon being removed from the user.

15. My recent research has focused primarily on embedded systems security with an emphasis on binary analysis, anomaly detection and automated security enforcement. In one case, I designed a security system that can be soldered directly to the CPU of embedded devices in order to perform control-flow integrity for purposes of profile building and enforcement. In another case, I designed a system to automatically discern name and version information from binaries on embedded devices in order to build a software profile of the platform configuration of the device which can then be cross-referenced with a vulnerability database. I have also written a patent on this specific technique.

16. As of 2023 I am a member of the program committee for the IEEE Symposium on Security and Privacy. As a member of this committee, I peer review research that has been submitted for publication at the Symposium. I have recently reviewed research on memory forensics, side channel attacks, password vaults, differential privacy, computational theory, memory module design, and hardware security.

C. Patents

17. I am a named inventor on six patents. A full listing of these patents can be found on my CV, attached as Exhibit A.

D. Industry

18. I began working in the software industry in 2008, when I obtained my first independent consulting client—Brandeis University. I created a program for their computer repair shop's ticketing system to interface with a label printer. This enabled the computer repair shop to automatically print labels associated with tickets in order to easily identify customers' devices.

19. Shortly after finishing my engagement with Brandeis University, I began working for the Johns Hopkins University Digital Research and Curation Center as a student programmer. There, I created a batch importer for a digital archiving system called DSpace in order to enable hundreds of digitally archived works per week to automatically be added to the repository. I next ported a web interface for statistics tracking and visualization to the same digital archiving system in order to allow library staff to more easily visualize and understand data about how the digital archiving system was being used through a common web-based interface.

20. In 2009, I began working for a security consulting company called Independent Security Evaluators in Baltimore, Maryland. I worked on a wide variety of security and privacy projects including projects to test and break the DRM scheme

of a magazine distribution application for IOS and Android (at the behest of the developer of said application), projects to test implementation code for cryptographic secret splitting, projects to assist with fuzz testing for security vulnerabilities of a variety of software applications, software development projects to automate testing of antivirus and antimalware solutions, and a variety of other security-related projects. Of note, I designed automation systems to create and restore virtualized snapshots of Windows machines on which I tested antivirus software on actual malware samples as part of a project for Consumer Reports. I worked for ISE as an intern throughout the semesters and summers until August 2011. In 2010, I also designed a security architecture for a large-scale digital curation system meant to be a major inter-university initiative to create a successor to existing digital curation systems that could be used for decades.

21. In spring 2011, I completed my bachelor's in computer science at Johns Hopkins University, and I also retained an independent consulting client, the University of Michigan, for whom I performed a penetration test and security assessment of a cloud-based research system that they planned to deploy.

22. In February 2016, I joined Harbor Labs full-time as a research scientist. In January 2019, I was promoted to Director of Firmware Analysis. In December 2023, I was promoted to Vice President of Applied Research and Technology. In October 2024, the division of Harbor Labs that I managed became Harbor Experts.

I became Chief Scientist at Harbor Experts during this time. At Harbor Experts, I manage client engagements related to all of the analysis work that Harbor Labs performs. I am also lead Harbor Experts' research projects.

23. A substantial portion of my role at Harbor Experts is in managing source code review engagements. As part of this work, I have reviewed software systems of varying sizes, often totaling in the millions or billions of lines of code. I have reviewed products in the security space, television-based set top boxes, network appliances, numerous web-based enterprise systems, email management systems, telephony products, embedded system bootloaders, social network platforms, virtualization platforms and countless other products.

24. As part of my work at Harbor Experts, and in my previous jobs and research, I have configured and managed many virtualization servers and emulation environments. Virtualization and emulation are core parts of the security analysis work that I perform, and as such, deep knowledge of these technologies is a prerequisite to be productive in this field. I continue to manage and use such environments on a regular basis.

25. Prior to joining Harbor Experts, I was also the technical and development lead for a firmware security analysis engine for a product developed at Harbor Labs called Firmware IQ. Firmware IQ is designed to analyze the firmware of embedded devices for security vulnerabilities in an automated fashion. In one

configuration of the product, a developer uploads a firmware image to a web-based portal which uses a broker to forward the firmware to an engine for analysis. The engine unpacks the firmware, breaks it into its constituent components, and then performs more than a hundred automated security checks in order to find vulnerabilities in the firmware image. The results are then reported through JSON to a web-based presentation system which displays them in an interactive fashion. Part of this work explored using emulation technologies to dynamically analysis embedded device firmware for security vulnerabilities.

26. A more detailed list of my engagements at Harbor Labs can be found on my attached vita.

II. Technology Background

A. Remote Access Technology

27. Remote access in computing refers to utilizing protocols and software applications to allow users to connect and control a computer (or other computation devices) from a remote location. Remote access enables users to interact with remote systems, perform tasks, and access resources while not physically present with the controlled system.

28. Remote access technology operates by gathering input information from a user—such as keyboard strokes, mouse movements, and commands that a user performs—and then executing that input on a remote device that the user is

attempting to control. Users provide these inputs directly on a general-purpose computer, a personal digital assistant (PDA), a smartphone, or another digital device that is in close proximity with the user. A remote access protocol is utilized to transmit these inputs to be executed on the remote device, as if these inputs were initiated directly on that remote device. Remote control protocols have been developed and utilized since the 1990s.

29. Remote Desktop Protocol (RDP) one of the most widely used protocols utilized for remote access. RDP, first introduced in 1996 as part of the Windows NT 4.0 Terminal Server edition,¹ is a proprietary protocol developed by Microsoft that enables users to connect to and control remote Windows-based computers over a network connection.² RDP provides a graphical interface for users to access and manage remote desktops and applications, transmitting input data (such as keyboard

¹ A Few More Reasons Why RDP is Insecure.
<https://thehackernews.com/2023/07/a-few-more-reasons-why-rdp-is-insecure.html>
(EX1017).

² Understanding the Remote Desktop Protocol (RDP).
<https://learn.microsoft.com/en-us/troubleshoot/windows-server/remote/understanding-remote-desktop-protocol> (EX1018).

and mouse actions) from the client to the server and sending display data back to the client.

30. Microsoft implements RDP with several software applications, with the most notable ones being “Remote Desktop Services” and “Remote Desktop clients.” Remote Desktop Services, originally known as Terminal Server in the 1996 Windows NT 4.0 release, is installed on the device that needs to be controlled by users, e.g., a Windows Server which technicians access remotely, that facilitates a server role that enables multiple users to access and use applications remotely.³ Remote Desktop clients are applications that allow users to connect to a remote Windows machine (e.g., the Windows Server installed with Remote Control Services, as discussed above) using RDP. Remote Desktop clients are currently available on various platforms, including Windows, macOS, iOS, and Android.⁴

³ Remote Desktop Services overview in Windows Server.
<https://learn.microsoft.com/en-us/windows-server/remote/remote-desktop-services/remote-desktop-services-overview> (EX1019).

⁴ Remote Desktop clients for Remote Desktop Services and remote PCs.
<https://learn.microsoft.com/en-us/windows-server/remote/remote-desktop-services/clients/remote-desktop-clients> (EX1020).

31. Remote Frame Buffer (RFB) protocol is another widely used protocol in remote access. RFB is specifically associated with a Virtual Network Computing (VNC), a graphical desktop-sharing system that allows users to remotely control another computer over a network.

32. VNC was developed in the late 1990s by researchers at the Olivetti Research Lab in Cambridge, England, and was initially released in 1998.⁵ VNC is platform-independent, meaning it can work across various operating systems including Windows, macOS, Linux, iOS, and Android.

33. Similar to RDP applications that I discussed above, the VNC architecture also operates on a client-server model, where the VNC server runs on the remote machine to be controlled, transmitting screen data and receiving input commands, while the VNC client is used by the remote user to view and interact with the server's desktop. RealVNC, the company that originates from researchers

⁵ Remote takeover: How RealVNC conquered the world.
<https://www.cam.ac.uk/research/features/remote-takeover-how-realvnc-conquered-the-world> (EX1021).

at the Olivetti Research Lab,⁶ develops RealVNC Server⁷ as the server on the remote machine to be controlled, and RealVNC Viewer⁸ as the application utilized by the user.

B. Cloud Virtualization Technology

1. Virtualization Technology

34. Virtualization refers to the process of creating a virtual version of computing resources, such as servers, storage devices, networks, and applications.

35. Suppose that an organization needs to run multiple applications on multiple operating systems. Without virtualization, this organization would need to prepare a plurality of physical computers, each with their own hardware processors and hardware storage devices. However, with virtualization, multiple virtual instances can be created, each running a standalone operating system and applications. Each virtual instance would not need its own physical hardware; multiple instances can share the same hardware, which reduces costs and simplifies

⁶ *Id.*

⁷ Download VNC Server by RealVNC.
<https://www.realvnc.com/en/connect/download/vnc/> (EX1022).

⁸ Download VNC Viewer by RealVNC.
<https://www.realvnc.com/en/connect/download/viewer/> (EX1023).

management. Additionally, virtualization enables applications to operate reliably across various environments, allowing multiple virtual instances with different operating systems to share the same hardware. Such flexibility helps organizations respond more effectively to their needs and provides the option to scale resources as required.

36. Within virtualization, Virtual Machines (VMs) are isolated environments that emulate physical computers. Each VM runs its own operating system and applications, enabling different workloads to coexist on the same hardware. Therefore, issues in one VM would not affect another.

37. VMs are created and managed by hypervisors. A hypervisor is a software that interfaces between physical hardware and the VMs. Its primary role is to manage the allocation of physical resources—such as Central Processing Unit (CPU) resources, memory, and storage—to each VM, ensuring that they operate efficiently and securely. By creating a layer of abstraction, the hypervisor enables multiple VMs to run on a single physical machine. Additionally, it facilitates communication between VMs and the underlying hardware, managing tasks such as input/output operations and network connectivity. Hypervisors can be classified into two main types: Type 1 (“bare-metal”), which runs directly on the hardware without a host operating system, and Type 2 (“hosted”), which operates on top of an existing operating system.

38. Virtualization has been an established idea since the 1970s. In the 1974 article “Survey of virtual machine research,” researchers already defined the terms “virtual machine systems” and “virtual machines.”⁹ Researchers also listed examples of virtual machines available at that time, such as IBM’s VM/370 mainframe.¹⁰

39. Commercial virtualization technologies are already widely adopted in the 2000s. VMware, for example, is a virtualization technology company that commercially released a Type-2 hypervisor VMware Workstation in 1999. In 2001, VMware further released a Type-1 hypervisor VMware ESX Server and Type-2 hypervisor VMware GSX Server.¹¹ Also in the 2000s, Citrix released a Type-1

⁹ Robert. P. Goldberg, *Survey of Virtual Machine Research*, IEEE COMPUTER MAGAZINE, 7(6):34-45, 1974.
<https://web.eecs.umich.edu/~prabal/teaching/eecs582-w11/readings/Gol74.pdf>
(EX1024).

¹⁰ R. J. Creasy, *The Origin of the VM/370 Time-Sharing System*, IBM JOURNAL OF RESEARCH AND DEVELOPMENT, 25(5):483-490, September 1981.
https://pages.cs.wisc.edu/~stjones/proj/vm_reading/ibmrd2505M.pdf (EX1025).

¹¹ VMware Celebrates Seven Years of Continual Innovation and Execution.
<https://news.broadcom.com/releases/7year> (EX1026).

hypervisor XenServer in 2003;¹² innotek released Type-2 hypervisor VirtualBox,¹³ which was later acquired by SUN Microsystems and then Oracle, becoming the Oracle VirtualBox that we know today.

40. In the late 2000s, operating system developers also start to release hypervisors built-in to their systems. Kernal-Based Virtual Machine (KVM) is a Type 1 hypervisor that was first integrated into the Linux kernel announced in 2006, and was merged into the Linux kernel mainline in 2007.¹⁴ Microsoft's Hyper-V is also a Type-1 hypervisor that was first announced and integrated into Windows Server in 2008.¹⁵

¹² [Xen-devel] [ANNOUNCE] Xen high-performance x86 virtualization (Oct. 2, 2003) <https://sourceforge.net/p/xen/mailman/message/5533663/> (EX1027).

¹³ News (older entries). https://www.virtualbox.org/wiki/News_pre_4_3 (EX1028).

¹⁴ Amit Shah, *Ten years of KVM* (Nov. 2, 2026) <https://lwn.net/Articles/705160/> (EX1029).

¹⁵ Hyper-V Made Available. <https://adtmag.com/articles/2008/06/26/hyperv-made-available.aspx> (EX1030); see also, Hyper-V RTM announcement. Available today from the Microsoft Download Centre. (Archived 2008.06.29).

2. Cloud Virtualization

41. A cloud computing environment is an infrastructure that provides access to computing resources over a network, e.g., the internet. Cloud virtualization, therefore, is a technology that allows virtualization to be managed within a cloud computing environment. In cloud virtualization, computing resources (such as a virtualized server) operate within a cloud computing environment where virtualization technology manages resource allocation among virtualized instances. Together, these elements facilitate the creation of virtual instances in the cloud capable of running various operating systems and applications simultaneously. With the usage of a cloud computing environment, cloud virtualization further enhances resource utilization, scalability, and flexibility benefits of virtualization.

42. In 2006, Amazon Web Services (AWS) began offering IT infrastructure services to businesses in a cloud computing environment. As an example, Amazon announced Elastic Compute Cloud (EC2) in 2006, that allowed users to rent virtual

<https://web.archive.org/web/20080629055418/http://blogs.technet.com/jhoward/archive/2008/06/26/hyper-v-rtm-announcement-available-today-from-the-microsoftdownload-centre.aspx>

servers in the cloud.¹⁶ AWS utilized virtualization technologies that assigns virtual CPUs and storage to provide scalable and flexible computing resources to these EC2 instances.¹⁷ Similarly, Microsoft announced Windows Azure in 2008;¹⁸ Google announced Google App Engine (GAE) in 2008, which later expanded to the Google Cloud Platform (GCP) that we know today.¹⁹ AWS, Azure, and GAE are all cloud-based Platform as a Service (PaaS) offerings, backed by virtualization technology, that enable developers to build, deploy, and manage applications without the need to manage the underlying infrastructure.

43. In the early 2010s, with the wide adoption of cloud computing environments, entities like the National Institute of Standards and Technology

¹⁶ Amazon EC2 beta, https://aws.amazon.com/blogs/aws/amazon_ec2_beta/ (EX1031).

¹⁷ *Id.*

¹⁸ Microsoft launches Windows Azure, <https://news.microsoft.com/announcement/microsoft-launches-windows-azure/> (EX1032).

¹⁹ Max Ross, *Reflecting on our ten year App Engine journey*, (Apr. 13, 2019), <https://cloud.google.com/blog/products/gcp/reflecting-on-our-ten-year-app-engine-journey/> (EX1033).

(NIST) have already established industry guidelines focusing on cloud virtualization technologies. One example would be NIST's Guide to Security for Full Virtualization Technologies (Special Publication 800-125), published in January 2011.²⁰ This guideline discusses "full virtualization technologies... used for operational efficiency, such as in cloud computing, and for allowing users to run applications for multiple operating systems on a single computer."

3. Cloud-Based Streaming

44. "Streaming" refers to the process of delivering video and audio content from a remote server to a user's device in real-time. Cloud-based streaming is a technology that allows users to access and consume digital content over a network without the need to download or install large files on their local devices.

45. A performance-intensive graphic game, for example, could be a data-intensive application that requires a lot of processing power for video and audio content. However, with limited resources and storage on a user's device, running that game locally may lead to poor performance, including lag, low frame rates, and

²⁰ Karen Scarfone, et al., *Guide to Security for Full Virtualization Technologies*, Nat'l Inst. Of Standards & Tech., [https://doi.org/10.6028/NIST.SP.800-125 \(EX1034\)](https://doi.org/10.6028/NIST.SP.800-125 (EX1034)).

compromised graphics quality. This can significantly diminish the gaming experience, making it less enjoyable for users.

46. However, by utilizing cloud-based streaming, the heavy lifting is done on remote servers, allowing users to conduct complex tasks without being constrained by their device's hardware limitations. This includes playing resource-intensive games. Instead of running the game locally on the user's hardware, the game runs on powerful servers in the cloud. The gameplay is captured as video and sent to the user's device, while the user's inputs (like controller or keyboard actions) are sent back to the server. The server then processes these user inputs, calculates responses in the game, in turn sends updated gameplay to the user's device. By leveraging powerful cloud infrastructure, game providers can offer high-quality media experiences while minimizing the hardware requirements for end-users.

47. Cloud-based game streaming services have been commercially available since at least the 2010s. OnLive, a provider of cloud virtualization technologies, launched an online game streaming platform in 2010, which allowed users to stream games over the internet.²¹ Some games were even enabled with 3D

²¹ Review: OnLive -- The What, Why and Who of Gaming in the Cloud, <https://www.wired.com/2011/06/review-onlive-the-what-why-and-who-of-gaming-in-the-cloud/> (EX1035).

effects, which required intensive computing power in the 2010s.²² A similar streaming service was Gaikai, released in 2011, which also allowed users to stream games directly from the cloud.²³ Upon release, Gaikai was able to enable users to play popular titles like Mass Effect: 2 instantly in their browsers without downloading.²⁴

48. Following the examples of OnLive and Gaikai, larger providers joined the development of cloud-based streaming platforms, including NVIDIA GeForce Now, Xbox Cloud Gaming, PlayStation Now, Google Stadia, Steam Remote Play, and Amazon Luna.

C. Mobile Cloud Computing

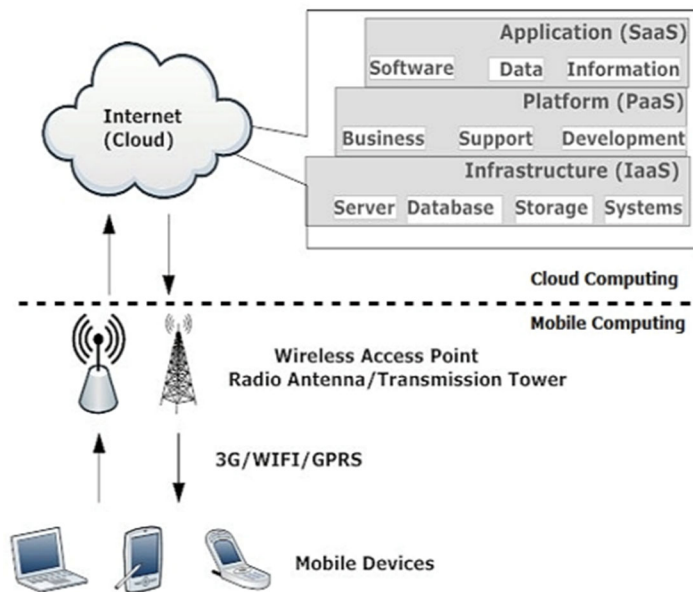
49. Mobile cloud computing (MCC) is a branch of cloud computing. It refers to technology that combines mobile computing with cloud computing, which enables mobile device users to not only access data, but also conduct processing in a cloud computing environment. Therefore, mobile devices can avoid running resource-intensive processes directly on the device itself. The following diagram

²² *Id.*

²³ Gaikai is Live!, (Feb. 6, 2011),
[https://dperry.com/2011/02/06/gaikai_is_live/\(EX1036\)](https://dperry.com/2011/02/06/gaikai_is_live/(EX1036)).

²⁴ *Id.*

from a May 2012 article provides a high-level architecture diagram of mobile cloud computing infrastructure, which demonstrates the communication between a mobile device to a cloud computing environment through wireless communications (e.g., 3G, Wi-Fi, GPRS).²⁵



50. Mobile devices are limited in terms of processing power, battery life, and storage capacity, compared to general-purpose computers. This is especially true if mobile devices try to run resource-intensive tasks like gaming or machine learning. Leveraging cloud computing resources to run complex applications,

²⁵ Han Qi & Abdullah Gani, *Research on mobile cloud computing: Review, trend and perspectives*, IEEE (pp. 195-202), <https://arxiv.org/pdf/1206.1118> (EX1037).

mobile cloud computing can provide a smoother user experience and enables richer functionalities without overburdening the mobile device.

51. These benefits of mobile cloud computing have already been well recognized in the field in early 2010s. For example, in a 2010 paper “Cloud Computing For Mobile Users: Can Offloading Computations Save Energy?” researchers identified energy-saving to be a major goal of mobile cloud computing, and concluded that cloud computations potentials save energy for mobile users. The paper further discusses the energy overhead for privacy, security, reliability, and data communications of mobile cloud computing.²⁶

52. Multiple groups of researchers in early 2010s also provide observations on the infrastructure, and the trends of mobile cloud computing. As an example, a 2011 article identified various applications utilizing mobile cloud computing at that time, including mobile commerce apps, mobile learning apps, mobile healthcare apps, and mobile gaming apps. This article further discusses communication and

²⁶ Karthik Kumar & Yung-Hsiang Lu, *Cloud computing for mobile users: Can offloading computation save energy?*, Comput. 43, no. 4, pp. 51-56 (Apr. 2010), <https://www.cs.purdue.edu/homes/bb/cs590/handouts/computer.pdf> (EX1038).

computation issues that arise with mobile cloud computing.²⁷ A May 2012 paper further provides three main optimization approaches in mobile cloud computing, which focus on inherent limitations of mobile devices, quality of communication, and division of application services.²⁸

III. Relevant Legal Standards

53. I have been asked to provide my opinions as to whether claims 1-8, 11-17, 20-25, and 28 of the '887 Patent would have been obvious to a person of ordinary skill in the art ("POSITA") at the time of the filing date of the '887 Patent.

54. I am an engineer by training and profession. The opinions I express in this declaration involve the application of my technical knowledge and experience to the evaluation of certain prior art with respect to the '887 Patent. In addition, I understand that the following legal principles apply.

55. It is my understanding that, in determining whether claims of the '887 Patent are unpatentable as obvious in this proceeding, the claim terms are generally given their ordinary and customary meaning as understood by a person of ordinary

²⁷ Dinh, H.T., Lee, C., Niyato, D. and Wang, P. (2013), A survey of mobile cloud computing: architecture, applications, and approaches. *Wirel. Commun. Mob. Comput.*, 13: 1587-1611. <https://doi.org/10.1002/wcm.1203> (EX1039).

²⁸ EX1037.

skill in the relevant art. A POSITA would read the claim terms in the context of the entire patent specification in which they appear, as well as the prosecution history of the patent.

56. It is my understanding that a claim is unpatentable under 35 U.S.C. § 103 if the claimed subject matter as a whole would have been obvious to a POSITA at the time of the alleged invention. I also understand that an obviousness analysis takes into account the scope and content of the prior art, the differences between the claimed subject matter and the prior art, and the level of ordinary skill in the art at the time of the invention.

57. In determining the scope and content of the prior art, it is my understanding that a reference is considered relevant prior art if it falls within the field of the inventor's endeavor. In addition, a reference is relevant prior art if it is reasonably pertinent to the particular problem with which the inventor was involved. A reference is reasonably pertinent if it logically would have commended itself to an inventor's attention in considering his problem. If a reference relates to the same problem as the claimed invention, that supports use of the reference as prior art in an obviousness analysis.

58. To assess the differences between prior art and the claimed subject matter, it is my understanding that 35 U.S.C. § 103 requires the claimed invention to be considered as a whole. This "as a whole" assessment involves showing that

one of ordinary skill in the art at the time of invention, confronted by the same problems as the inventor and with no knowledge of the claimed invention, would have selected the elements from the prior art and combined them in the claimed manner.

59. It is my further understanding that several rationales may be applied for combining references or modifying a reference to show obviousness of claimed subject matter. These rationales include (but are not limited to): combining prior art elements according to known methods to yield predictable results; simple substitution of one known element for another to obtain predictable results; a predictable use of prior art elements according to their established functions; applying a known technique to a known device (method or product) ready for improvement to yield predictable results; choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success; and some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify a prior art reference or to combine prior art teachings to arrive at the claimed invention.

IV. Person of Ordinary Skill in the Art (“POSITA”)

60. It is my understanding that when interpreting the claims of the ’887 Patent, I must do so based on the perspective of a person of ordinary skill in the art

(“POSITA”) at the relevant priority date. My understanding is that the earliest claimed priority date of the ’887 Patent is May 31, 2012.

61. It is my opinion that one of ordinary skill in the art in the field of mobile cloud computing in 2012 would have had a bachelor’s degree and two years’ experience in cloud computing or a similar field, or a master’s degree, in computer science or similar field.

62. I am a person of at least ordinary skill in the art.

V. ’887 Patent Overview

A. Specification

63. The ’887 Patent discloses “a network in which a mobile device is configured as an input/output interface for a user such that actual processing, storage and network interfaces are performed or provided by a remote server.” ’887 Patent, Abstract. The ’887 Patent asserts that “decoupling mobile device HW [hardware] capabilities from actual mobile device performance and a user experience, [] enables upgrading mobile device functionality, performance, OS and applications without the need to change the HW or the local SW [software] of the mobile device itself.” *Id.*, 2:20-24. Specifically, the ’887 Patent states that:

The mobile device 10 is configured to minor to the server 12 all or a portion of information input to the mobile device 10. That is, the mobile device 10 receives input information from the input components and

sends the input information to the server 12 for processing by the server 12. The mobile device 10 (i.e., the hardware (HW) and local software (SW) of the mobile device 10) thus does not perform the processing locally at the mobile device 10 itself. For example, the mobile device 10 may mirror input information including touch screen gestures, pressed buttons, sensor data, audio data and video data to the server 12 over a packet based network, and the server 12 processes the input information to produce a display and/or audio output based on the input information.

Id., 4:53-65.

64. While the '887 Patent claims to offload processing to a server, it nonetheless acknowledges that the mobile device “encodes the sampled input for transmission.” *Id.*, 7:22-23 It also still “performs intensive processing tasks”:

The basic functionality of the mobile device 10 further comprises receiving, identifying and encapsulating information input or changes made to a user interface (touch screen, physical keys, etc.) into messages/data streams for transmittal to the server 12, receiving, identifying and encapsulating HW sensor data and/or changes thereto into messages/data streams for transmittal to the server 12 (sensors include motion, environmental and position sensors), receiving, encoding (e.g., compression) and encapsulating camera and/or camcorder (e.g., picture and/or video data) and/or microphone (audio data) information and sending the encoded and encapsulated information to the server 12 (encoding may include compression performed to save network bandwidth). The basic functionality of the mobile device 10 further

comprises intensive processing tasks, e.g., encoding/decoding of video, which may be accelerated by HW of the mobile device 10, notifications (receiving at the mobile device 10 from the server 12 notifications of new events and reporting them to the user (e.g., a telephone call, a new E-mail, a new SMS, etc. . . .)) and server-to-client protocol for decoding video, audio, image and text data (user interface information) separately streamed from the server 12 to the mobile device 10, integration of the video, image and/or text data into display data for producing a display and reproduction thereof by the display screen with reproduction of audio data by the speaker.

Id., 7:38-62. Thus, a POSITA would have understood that the '887 Patent merely teaches that the mobile device does not process the input information to produce a display, audio, or HW action by the mobile device while still allowing for other processing by the mobile device.

65. The '887 Patent discloses that the server emulates components of the mobile device to execute a mobile device OS, processes the input received from the mobile device, and streams user interface information back to the mobile device:

That is, the server 12 executes a mobile device operating system (OS) (e.g., Google Android 4.1 or Windows phone 8) and one or more applications (e.g., a Game or GPS navigation SW) on emulated mobile device HW based on the input information received from the mobile device 10. The server streams all animations and/or graphics changes, e.g., changes in games, web browsing and scrolling screens that occur on the virtual mobile device of the mobile OS and emulated HW, to the mobile device 10 as a

stream of user interface information. The user interface information is data that can be used to recreate or produce a display and/or audio without any processing thereof other than the decoding and playback or reproduction of the user interface information itself by the mobile device 10.

Id., 4:66-5:12.

B. Prosecution History

66. The '887 Patent issued from U.S. Application 13/606,675 ("'675 Application"). On December 30, 2013, the Examiner rejected all pending claims of the '675 Application as obvious over combinations of various prior art. '887 File History, 127-165. Relevant here, the Applicant amended the pending independent claims 20, 30, and 47 (issued as claims 1, 11, and 20, respectively) to recite that the server produces "*a stream of* user interface information" for use in reproducing a display, audio, or HW action at the mobile device. '887 File History, 182-190. The last independent claim (39, which issued as claim 20) was amended to specify that the server produces user interface information "based on the same instance of the mobile device OS." *Id.*, 187. The applicant also made the following arguments to distinguish the newly amended independent claims from the prior art.

67. Regarding claim 20 (issued claim 1), Applicant argued that the alleged invention is distinct from the prior art because all processing of input data occurs at the server:

In Applicants' claimed method of independent claim 20, all input data processing functionality is executed at the server, and the local, mobile physical device itself does not execute any such functionality. In accordance with the claimed invention, the server is the only entity that executes this mobile device data processing functionality using the mobile device input that the server is sent from the mobile device.

Id., 195 (emphasis in original).

68. Regarding claims 30 (issued claim 11) and 39 (issued claim 20), Applicant argued that the alleged invention is distinct from the prior art because the multiple data streams arrive at a mobile device from the same emulation:

Put differently, applicants' claimed invention provides multiple data streams (e.g., audio, video, compressed text, voice and/or sensors) arriving from a single mobile device emulation. [Prior art] fails to teach or suggest this limitation because [prior art] expressly describes the use of multiple streams arriving from multiple applications.

Id., 198. Applicant also distinguished prior art in which allegedly different instances of a video game served multiple clients:

[Prior art] teaches a system in which multiple instances of Game Logic 210 are provided to multiple clients 110. For example, [Prior art] explains that a "server is ... configured to serve multiple video games to multiple clients". Each copy of the video game is not the same instance of that video game. In accordance with applicants' claimed invention, the same instance of a virtual mobile device serves multiple mobile devices.

Id., 198-199 (emphasis in original).

69. Regarding claim 47 (issued claim 28), Applicant distinguished prior art by arguing that the claim more narrowly requires “different kinds” of mobile OS’s:

That is, different kinds of mobile device OS’s are executed and not merely multiple instances of the same mobile device OS. [Prior art] fails to teach or suggests this expressly-recited subject matter.

...

[Prior art] teaches that multiple emulated virtual instances are all the same as one another. Independent claim 47, on the other hand, expressly recites that “the second mobile device OS has at least one of a different OS type, a different OS version and a different OS instance than the first mobile device OS.”

Id., 199-200 (emphasis in original). I consider both the express claim language and Applicant’s arguments during prosecution in my invalidity analysis below.

70. The claims were then allowed following an Examiner’s Amendment to claims 20 and 39. *Id.*, 211-214. Claim 20 (issued claim 1) was amended to recite “at least one of the display, the audio, and the hardware (HW) action by the HW component of the mobile device based on a same instance of the mobile device operating system,” and claim 39 (issued claim 20) was amended to additionally recite “wherein the input information has not been processed to produce a display, audio and a hardware (HW) action by a corresponding HW component of the mobile devices.” *Id.*

VI. CLAIM CONSTRUCTION

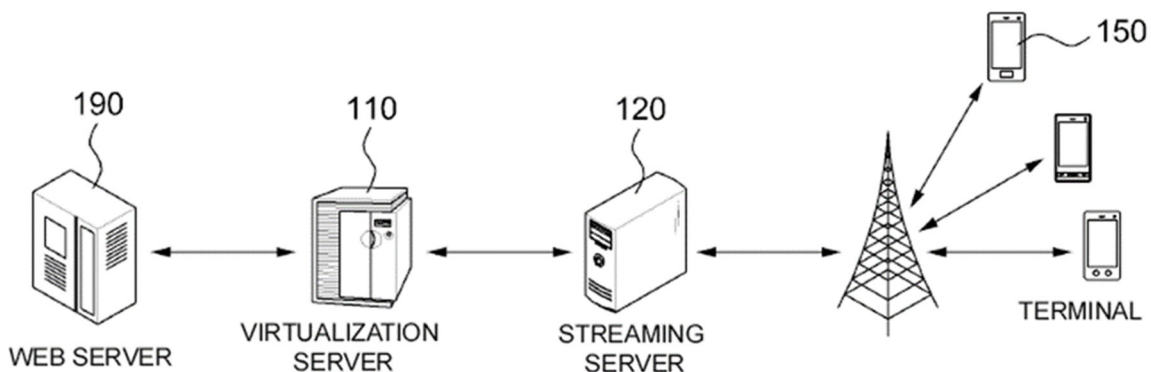
71. I understand that claims should be construed “in accordance with the ordinary and customary meaning of such claim as understood by one of ordinary skill in the art and the prosecution history pertaining to the patent.” In my analysis below, I apply the plain an ordinary meaning of the claim language as would have been understood by a POSITA at the time of the alleged invention, unless otherwise noted.

VII. DETAILED EXPLANATION OF GROUNDS

A. Ground 1A: Yoon

1. Yoon

72. Yoon discloses “a method and an apparatus for providing a virtual environment of a mobile operating system (OS) and a mobile application.” Yoon, Abstract. Yoon teaches a system comprising a terminal 150 (such as a smart phone), a virtualization server 110, a streaming server 120, and a web server 190, all “connected through a high-speed network such as a local area network.” Yoon, 11.



Yoon, Fig. 1.

73. Yoon's terminal comprises a thin client:

The thin client interprets data related to execution of the mobile OS and/or application transmitted by the virtualization server 110 (for example, an execution screen of the mobile OS and/or application, a sound outputted from the mobile OS and/or application), and outputs the data to the display or speaker, etc. of the terminal.

Yoon, 12. The thin client also transmits input information to the terminal 150 "to the virtualization server 110 to allow the mobile OS and/or application [executed at the virtualization server 110] to operate according to the input information." *Id.*

74. Yoon's virtualization server "provides a virtual environment of a mobile OS and an application operating in the mobile OS." Yoon, 7. The virtualization server further includes "a CPU emulation layer configured to emulate a CPU for supporting the mobile OS." Yoon, 10. The CPU emulation layer further comprises a virtual architecture emulation layer 422, a CPU SOC layer 424, and a device emulator 426. Yoon describes the role of each of these components in the excerpts and partial Figure 4 that I have copied below:

The CPU emulation layer 420 emulates a CPU for supporting a mobile OS of a mobile OS stack. The CPU may be a typical CPU for a mobile terminal such as an ARM.

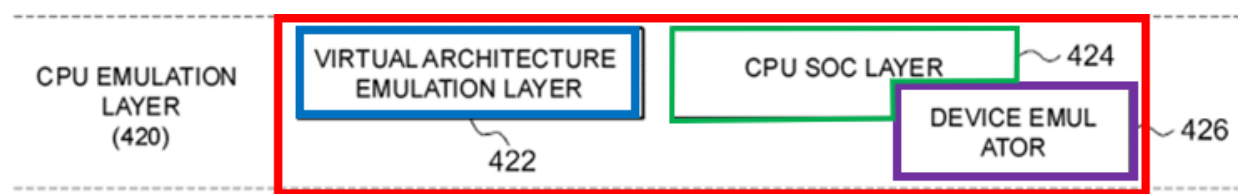
The CPU emulation layer 420 includes a virtual architecture emulation layer 422 and a virtual CPU system on chip (SOC) layer 424.

The virtual architecture emulation layer 422 emulates an architecture for operating the CPU SOC of the virtual CPU SOC layer. The virtual architecture emulation layer 422 may support a widely known architecture such as x86, MIPS, SPARC, ARM architecture, etc. The virtual architecture emulation layer 422 may be a typical architecture emulator such as QEMU.

The virtual CPU SOC layer 424 emulates the CPU SOC operating on the virtual architecture. The virtual CPU SOC layer 424 may be a typical CPU SOC emulator such as Goldfish.

A device emulator 426 may be plugged in the virtual CPU SOC layer 424 to emulate a device of a terminal having a mobile terminal CPU mounted therein. That is, the virtual CPU SOC layer 424 may include the device emulator 426.

Yoon, 21.



Yoon, Fig. 4 (annotated).

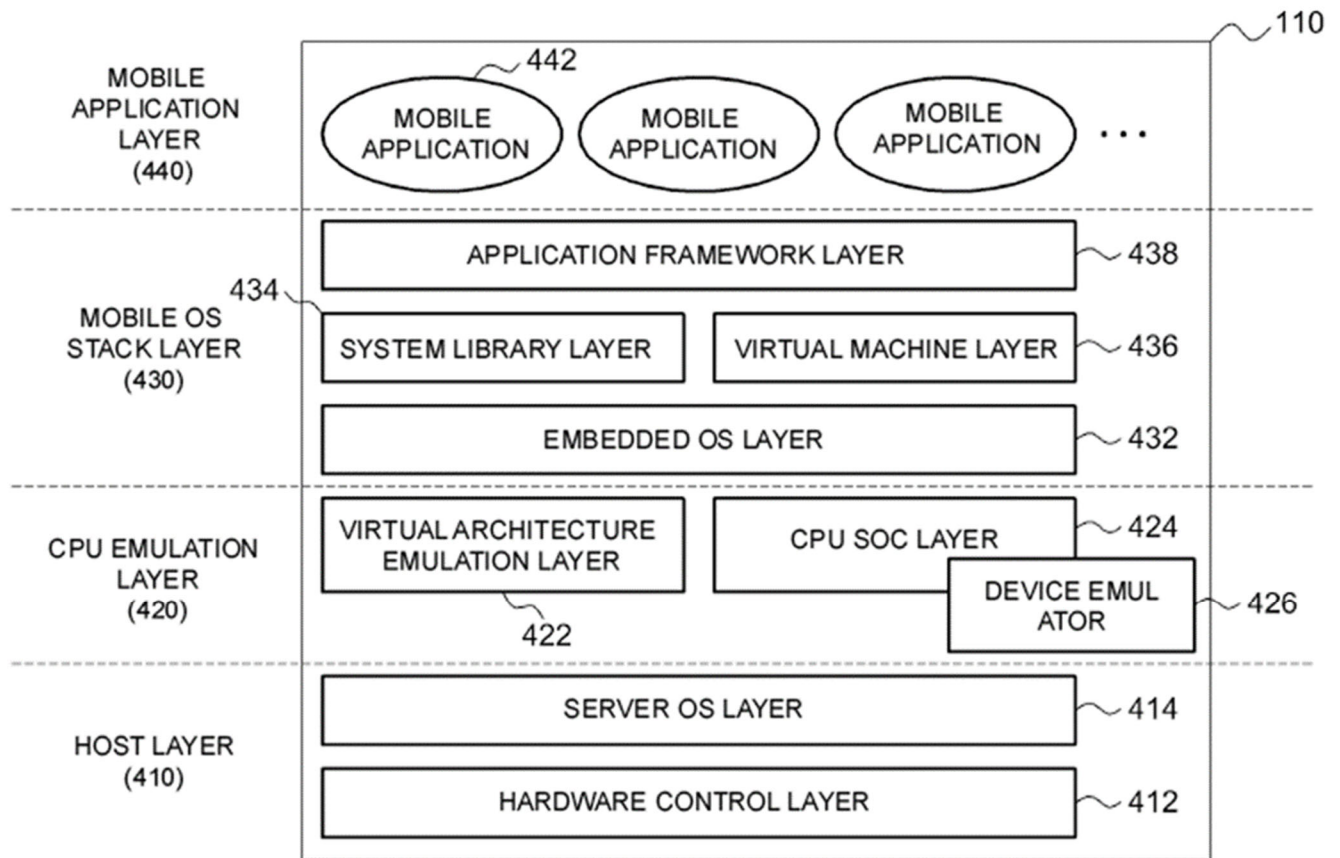
75. Yoon also discloses a streaming server and a web server, but notes that the functions of each of these servers may be integrated into the virtualization server.

For the purposed of my analysis, I use the embodiment of Yoon's virtualization server that has the functions of the streaming server and web server integrated within.

76. “The streaming server 120 performs a relay role of data transmission and reception between the virtualization server 110 and the terminal 150.” Yoon, 11. “[T]he functions of the streaming server 120...may be integrated into the functions of the virtualization server 110.” Yoon, 15.

77. “The web server 190 provides the web service by...operating in a virtual environment of the virtualization server 110.” Yoon, 11. “[T]he functions of the web server 190...may be integrated into the functions of the virtualization server 110” as well. Yoon, 16-17.

78. Figure 4 depicts the components of Yoon's virtualization server in additional detail:



Yoon, Fig. 4.

2. Claim 1

- a. 1[pre]: A method for emulating a mobile device at a server, the method comprising:**

79. Yoon discloses “a method for the virtualization server 110 to deliver information on the operating mobile OS and/or application to the terminal 150.” Yoon, 13. Yoon further discloses that its “device emulator 426...emulate[s] a device of a terminal having a mobile terminal CPU mounted therein.” Yoon, 21. Yoon thus

discloses a method for its virtualization server (*server*) *emulating* a terminal (*mobile device*).

- b. 1[a]: receiving, by the server, input information from at least one mobile device, wherein the input information has not been processed to produce a display, audio and a hardware (HW) action by a HW component of the mobile device;**

80. Yoon discloses its virtualization server “receiving input information for manipulating the mobile OS or the application from the terminal.” Yoon, 8, *see also*, *id.*, 12 (“The user of the terminal 150 may manipulate the mobile OS and/or application provided by the virtualization service by using an output device (for example, a liquid crystal display (LCD), a speaker, etc.) and an input device (for example, a button, a touch pad, a key pad, a camera, a mike, a mouse, a track point, etc.) of the terminal 150”). Indeed, Yoon explains at length that the virtualization server processes the input information from the terminal. Yoon, 14-16. For example:

The user of the terminal 150 should be able to manipulate the mobile OS and/or application through the input device of the terminal 150 to use the mobile OS and the application operating in the virtual environment.

Yoon, 14.

The virtualization server 150 which receives the input information delivers the input information to the mobile OS and/or application operating in the virtual environment. The mobile OS and the application

may receive the input information from the virtualization server 110 through such a method as an event, interrupt, and polling.

Yoon, 14.

As described above, data related to the virtual environment should be transceived between the virtualization server 110 and the terminal 150. That is, virtual environment information is transmitted from the virtualization server 110 to the terminal 150, and input information is transmitted from the terminal 150 to the virtualization server 110.

Yoon, 14.

In this case, instead of the virtualization server 110 and the terminal 150 directly communicating, the streaming server 120 may transform and process data to be transceived between the virtualization server 110 and the terminal 150, and may transmit it.

Yoon, 15.

The streaming server 120 may transmit the input information transmitted from the terminal 150 to the virtualization server 110. In this case, the transmitted input information may be transformed by the streaming server 120 to conform to specifications used by the virtualization server 110 before being transmitted to the virtualization server 110.

Yoon, 15.

The streaming server 120 may be an additional member, and the functions of the streaming server 120 explained in embodiments of the disclosure may be omitted or may be integrated into the functions of the virtualization server 110. For example, the virtualization server 110

may encode the image of the virtual environment into a video format, and may transmit the same to the terminal 150.

Yoon, 15.

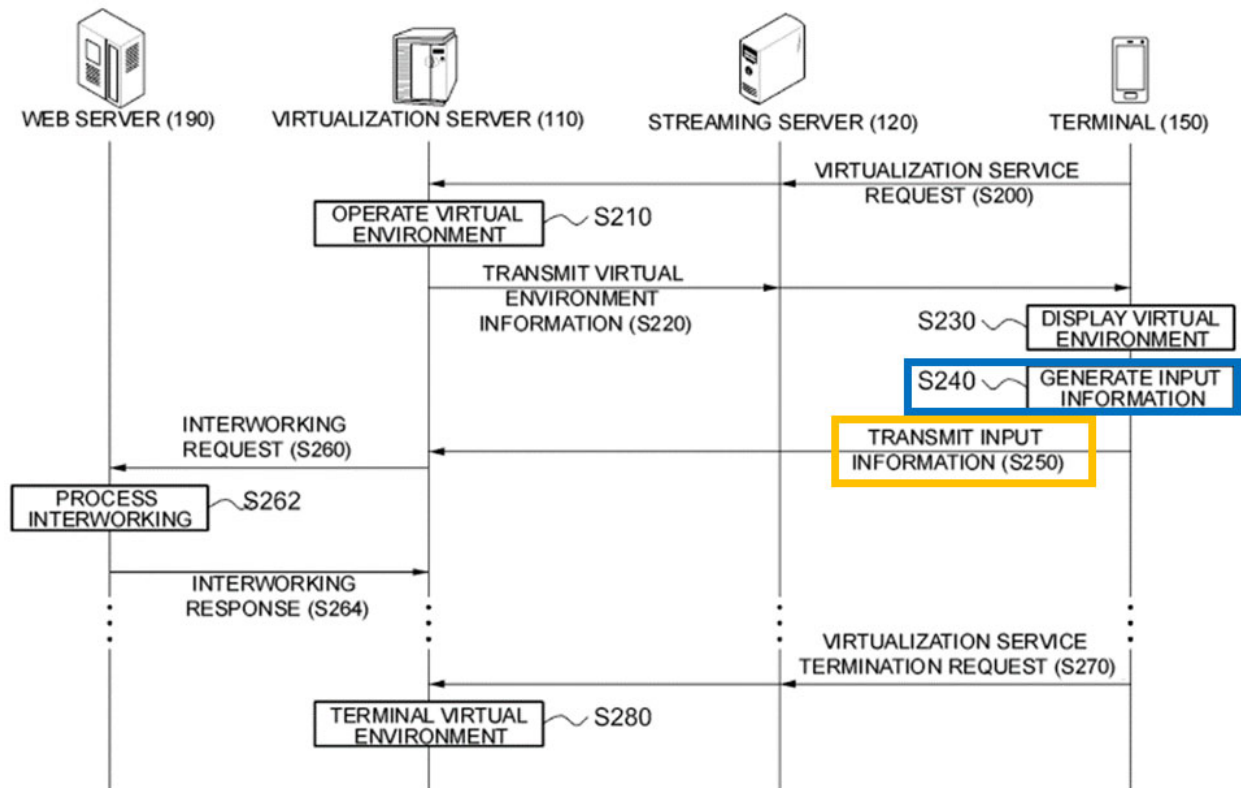
For example, the mobile OS and/or the application may require an input to a specific button on the premise that the terminal is provided with the specific hardware button (for example, a home button, a back button, a search button, etc.).

In this case, an input signal generated by the terminal 150 (or the thin client of the terminal) should be transformed to be suitable for the mobile OS and/or the application (for example, a press/release signal of the specific button of the terminal 150 should be transformed into a press/release signal of another button required for the mobile OS and/or the application). Such transformation may be performed by one or more of the terminal 150, the thin client of the terminal, the streaming server 120, and the virtualization server 110 in whole or part.

Yoon, 16.

81. Notably, as shown by the quotes above, Yoon discloses that input information is expressly “for manipulating the mobile OS or the application” at the virtualization server and that to the extent that transformation of the input information is necessary, the transformation is carried out by the virtualization or streaming server (the functions of which are integrated into the virtualization server, as I note above). Yoon later explicitly discloses that controller 320, which is part of the virtualization server, “may process the input information. The processed input

information may be reflected on operations of the mobile OS or the application.” Yoon, 20. Also notably absent is any disclosure that input information is processed at the terminal to create any of a display, audio, or HW action. Instead, Yoon explicitly discloses that input information is generated by the terminal in S240 [blue] of its Figure 2, and in the next step, S250 [orange], the input information is transmitted to the virtualization server. Yoon, 18; Fig. 2.



Yoon, Fig. 2 (annotated).

82. Thus, a POSITA would have understood that Yoon discloses that input information is processed at the server rather than being processed at the terminal (*the*

input information has not been processed to produce a display, audio and a hardware (HW) action by a HW component of the mobile device).

- c. **1[b]: processing, by a mobile device operating system (OS) executed at the server, the input information to produce a stream of user interface information for use in reproducing at the mobile device at least one of the display, the audio and the hardware (HW) action by the HW component of the mobile device based on a same instance of the mobile device operating system; and**

83. Yoon discloses that its virtualization server executes a mobile device operating system. Yoon, 11 (“mobile OS and an application operating in the mobile OS are executed [at the virtualization server] as if they are directly executed by the terminal 150”), 11-12 (“The virtualization server 110 operates the mobile OS and the application and provides information on the operating mobile OS and/or application to the terminal 150, thereby providing a user of the terminal 150 with the environment in which the mobile OS and/or application is executed as if it is directly executed by the terminal 150.”); 18 (“the mobile OS and/or the application executed in the virtualization server 110...”), 22 (“the device emulator may emulate an execution screen of the mobile OS and/or the application”);

84. Yoon further discloses that the user “manipulate[s] the mobile OS and/or application provided by the virtualization service by using an output device (for example, a liquid crystal display (LCD), a speaker, etc.) and an input device...of

the terminal 150.” Yoon, 12. And the “virtualization server 110 operates the mobile OS and the application and provides information on the operating mobile OS and/or application to the terminal 150.” Yoon, 11. For example, “the virtualization server 110 generates an execution screen image of the running mobile OS and/or application” as well as “a sound generated by the running mobile OS and/or application,” and both are transmitted in a stream to the terminal. Yoon, 14-15. Yoon thus discloses that the mobile device OS executed at the virtualization server processes input from terminal 150 to produce a stream of user interface information.

85. Finally, a POSITA would have understood that the stream of user interface information from the virtualization server is “*based on a same instance of the mobile device operating system.*” Yoon contemplates either a single emulation instance servicing multiple terminals or a single emulation instance servicing a single terminal:

Each of the plurality of CPU emulation layers 420 may be to provide a virtual environment to one or more virtualization service requestors (for example, the user of the terminal 150).

...

That is, a plurality of CPU emulation layers 420 and/or a plurality of mobile OS stack layers 430 may be provided to provide various virtual environments to the terminals 150, and a separate CPU emulation layer 420 and/or mobile OS stack layer 430 may be generated for each virtualization service requestor.

Yoon, 22. But Yoon never discusses multiple emulation instances providing user interface information to the same terminal. Indeed, it would not make sense for multiple emulations to provide user interface information to the same terminal. Doing so would immensely complicate Yoon's system because user inputs would need to be transmitted to multiple emulation instances, each of those instances would need to transmit a stream of user interface information back to the terminal, and the emulations would need to coordinate among themselves to spread the workload without duplication of effort. Yet Yoon neither provides any suggestion for or teaching of how one would want to do so. Thus, to the extent there is any doubt as to whether Yoon teaches or suggests that the stream of user interface information is based on the same instance of the mobile device operating system, POSITA would have understood that it does, and that the alternative is neither taught nor suggested by Yoon.

- d. 1[c]: sending, by the server, the stream of user interface information to the mobile device to reproduce at the mobile device at least one of the display, the audio and the HW action by the HW component of the mobile device.**

86. Yoon discloses that the virtualization server transmits ("*sending*") the stream of video and sound ("*user interface information*") output from the virtualization server to the terminal ("*the mobile device*").

As described above, data related to the virtual environment should be transceived between the virtualization server 110 and the terminal 150. That is, virtual environment information is transmitted from the virtualization server 110 to the terminal 150, and input information is transmitted from the terminal 150 to the virtualization server 110.

...

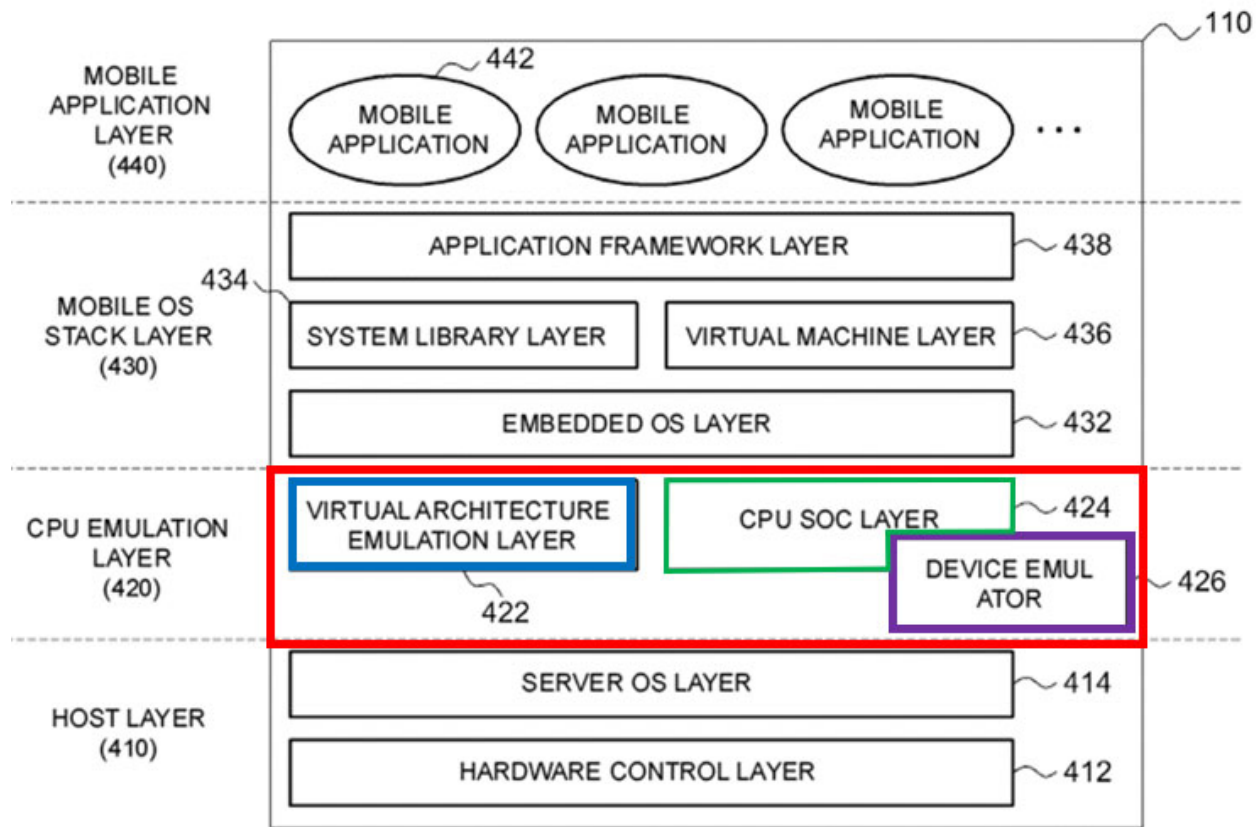
For example, when the virtualization server 110 generates an execution screen image of the running mobile OS and/or application, it may be more efficient to transmit the generated images as a video stream having a high compression rate than to transmit generated images individually.

For example, when the virtualization server 110 outputs a sound generated by the running mobile OS and/or application, it may be more efficient to transmit the sound as a sound of a video stream having a high compression rate than to transmit the sound without compressing.

Yoon, 14-15. The terminal then displays images and plays sounds received from the virtualization server (*“reproduce at the mobile device at least one of the display, the audio, and the HW action...”*). Yoon, 18 (*“the terminal 150 outputs the virtual environment to the output device by using the received virtual environment information. That is, the screen of the mobile OS and/or the application may be displayed on the display, and the sound generated by the mobile OS and/or the application may be outputted through the speaker.”*).

**3. Claim 2: The method of claim 1, further comprising:
emulating, by the server, mobile device HW and executing
the mobile device OS on the emulated mobile device HW.**

87. Yoon discloses its virtualization server emulating mobile device HW and executing the mobile device OS on the emulated mobile device HW. Yoon's Figure 4, which I have annotated below, illustrates the software architecture of the virtualization server. Yoon discloses that the virtualization server's "CPU emulation layer 420 [red] emulates a CPU for supporting a mobile OS." Yoon, 21. Yoon further discloses a "virtual architecture emulation layer 422" [blue], a "virtual CPU system on chip (SOC) layer 424" [green], and a "device emulator 426" [purple] "plugged in the virtual CPU SOC layer 424 to emulate a device of a terminal having a mobile terminal CPU mounted therein" (collectively "*emulating, by the server, mobile device HW*"). Yoon, 21, Fig. 4 (annotated). Yoon also discloses that the mobile device OS is then executed on the emulated mobile device HW. *Id.* ("mobile OS stack layer 430 may operate as an application of the...CPU emulation layer 420, and may provide the function of the mobile OS");



Yoon, Fig. 4 (annotated).

4. Claim 3

- a. 3[a]: The method of claim 1, further comprising:
executing, by the server, at least one application on
the mobile OS;**

88. Yoon discloses that “an application 422 of the mobile OS is executed in the mobile application layer 440” of the virtualization server. Yoon, 21. “The mobile application layer 440 operates as an application of the mobile OS stack layer 430,” which “provide[s] the function of the mobile OS.” *Id.* Thus, the server executes at least one application on the mobile OS.

- b. 3[b]: wherein the at least one application processes the input information to produce the stream of user interface information.**

89. As explained in Section VII.A.2.c, Yoon teaches that the mobile OS *and/or* application processes user input to produce a stream of interface information. Yoon, 12 (“The user of the terminal 150 may manipulate the mobile OS and/or application provided by the virtualization service by using...an input device...of the terminal 150.”). Thus, Yoon discloses that the at least one application processes user input to produce the stream of user interface information.

- 5. Claim 5: The method of claim 1, further comprising: receiving, by the server, the input information from a plurality of different mobile devices, wherein a same instance of the mobile device OS processes the input information from each of the plurality of different mobile devices to produce the stream of user interface information.**

90. Yoon discloses that “[e]ach of the plurality of mobile OS stack layers 430 may be to provide a virtual environment for one or more virtualization service requestors.” Yoon, 22. Thus “each” mobile OS stack layer can provide a virtual environment to more than one virtualization service requestors, which Yoon specifies to be, “for example, the user[s] of the terminal 150.” Yoon, 22. A POSITA would therefore have understood Yoon to teach that the *same instance of the mobile OS*, i.e., the same virtual environment provided by the mobile OS stack layer on the virtualization server, can receive and *process the input information* from a *plurality*

of different mobile devices to produce the stream of user interface information for each (as I discuss in claim 1, Section VII.A.2).

6. Claim 20

- a. 20[pre]: A method for emulating a mobile device at a server, the method comprising:**

91. Yoon discloses or suggests the preamble for the reasons I discuss in Section VII.A.2.a.

- b. 20[a]: receiving, by the server, input information from a plurality of different mobile devices, wherein the input information has not been processed to produce a display, audio and a hardware (HW) action by a corresponding HW component of the mobile devices;**

92. Yoon discloses or suggests this limitation for the reasons I discuss in Sections VII.A.2.b and VII.A.5. Yoon teaches its virtualization server receiving input information that has not been processed from a terminal (Section VII.A.2.b), and that multiple terminals may interact with the same mobile OS instance (Section VII.A.5).

- c. **20[b]: processing, by a same instance of a mobile device operating system (OS) executed at the server, the input information from each of the plurality of different mobile devices to produce user interface information for use in reproducing at the different mobile devices at least one of the display, audio and the HW action by the HW components of the different mobile device based on the same instance of the mobile device OS; and**

93. Yoon discloses or suggests this limitation for the reasons I discuss in Sections VII.A.2.c and VII.A.5. Yoon teaches that the same mobile OS instance processes input information from multiple terminals and produces a media stream to send to each of the terminals, where the stream will be rendered.

- d. **20[c]: sending, by the server, the user interface information to at least one of the plurality of different mobile devices.**

94. Yoon discloses or suggests this limitation for the reasons I discuss in Sections VII.A.2.d and VII.A.5. As I explained, Yoon discloses provide user interface information, i.e., the virtual environment, to a terminal, and that the same instance of the mobile OS can provide a virtual environment to multiple terminals. Thus, a POSITA would have understood that the mobile OS instance sends user interface information to at least one of said terminals (“*at least one of the plurality of different mobile devices*”).

7. Claim 21

- a. 21[a]: The method of claim 20, further comprising: executing, by the server, at least one application on the mobile device OS;**

95. Yoon discloses or suggests this limitation for the reasons I discuss in Section VII.A.4.a.

- b. 21[b]: wherein the at least one application processes the input information to produce the user interface information.**

96. Yoon discloses or suggests this limitation for the reasons I discuss in Section VII.A.4.b.

8. Claim 28

- a. 28[pre]: A method for emulating a mobile device at a server, the method comprising:**

97. Yoon discloses or suggests the preamble for the reasons I discuss in Section VII.A.2.a.

- b. 28[a]: receiving, by server, first input information from a mobile device;**

98. Yoon discloses or suggests this limitation for the reasons I discuss in Section VII.A.2.b.

- c. **28[b]: processing, by a first mobile device operating system (OS) executed at the server, the first input information to produce a first stream of user interface information for use in reproducing at the mobile device at least one of a display, audio and a hardware (HW) action by a HW component of the mobile device; and**

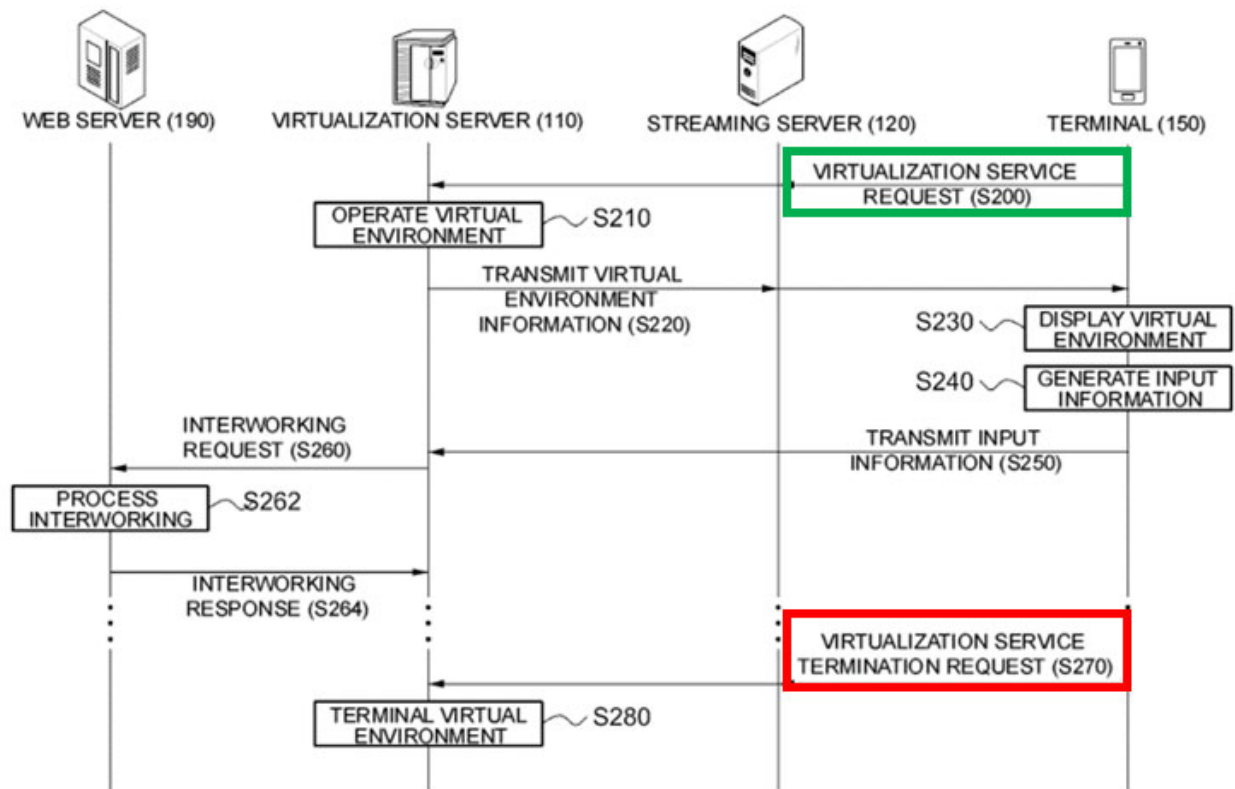
99. Yoon discloses or suggests this limitation for the reasons I discuss in Section VII.A.2.c.

- d. **28[c]: sending, by the server, the first stream of user interface information to the mobile device to reproduce on the mobile device at least one of the display, audio and the HW action by the HW component of the mobile device;**

100. Yoon discloses or suggests this limitation for the reasons I discuss in Section VII.A.2.d.

- e. **28[d]: receiving, by the server, second input information from the mobile device;**

101. Yoon discloses that a terminal may initiate [green] and terminate [red] a virtualization service with the virtualization server, as illustrated in Yoon's Figure 2, which I have annotated below. *E.g.*, Yoon, 17 ("At step S200, the terminal 150 requests a virtualization service from the virtualization server 110."), 19 ("At step S270, the terminal 150 transmits a virtualization service termination request to the virtualization server 110."), Fig. 2 (annotated).



Yoon, Fig. 2 (annotated).

102. Because a terminal can initiate and terminate virtualization sessions, a POSITA would have understood that a terminal could therefore engage in multiple virtualization sessions with the virtualization server by requesting a second session after terminating a first session. Where a terminal engages in a second virtualization session with the virtualization server, a POSITA would have understood that the virtualization server receives second input information from the mobile device during said second session.

103. To the extent Patent Owner (“PO”) argues that this claim requires “different kinds of mobile device OS’s are executed and not merely multiple

instances of the same mobile device OS” (as PO argued during prosecution, Section V.B), Yoon likewise discloses or suggests this limitation. In particular, Yoon discloses that “[a] plurality of mobile OS stack layers 430 may be provided” where “[e]ach of the plurality of mobile OS stack layers 430 may emulate a different OS.” Yoon, 22. A POSITA would have understood or found it obvious that Yoon’s virtualization server could execute multiple instances of the same mobile OS or different kinds of mobile OS’s. A POSITA would have understood that this is what Yoon means by “a different OS” because just previous to that, Yoon states that “[e]ach of the plurality of CPU emulation layers 420 may emulate a different CPU.” Yoon, 22. Because the mobile OS stack layer is built on top of the CPU emulation layer, emulating different CPUs would have been needed to provide a virtual environment for different kinds of OS’s, which were built to run on different kinds of hardware, i.e., CPUs. This is also consistent with Yoon’s stated objective of solving the problem where “an application that is developed for a specific smart phone OS may not be used for other smart phones that do not use the OS.” Yoon, 7. Emulating different kinds of mobile OS’s would allow a user to access applications that would not be available if only multiple instances of the same mobile OS were emulated.

- f. **28[e]: processing, by a second mobile device operating system (OS) executed by the server, the second input information to produce a second stream of user interface information for reproducing at least one of a display, audio and a HW action by a HW component of the mobile device, wherein the second mobile device OS has at least one of a different OS type, a different OS version and a different OS instance than the first mobile device OS; and**

104. As I discuss in Section VII.A.8.e, Yoon discloses or suggests a terminal engaging in multiple virtualization sessions. Yoon also discloses or suggests this limitation to the extent that the claim requires that “different kinds of mobile OS’s are executed” for the reasons I discuss in Section VII.A.8.e. Yoon thus discloses this limitation for the same reasons discussed in Section VII.A.2.c (“processing...” limitation) and because a second virtualization session would either be processed by “a different instance than the first mobile device OS” because the first instance would have been terminated already or be an entirely different kind of mobile OS.

- g. **28[f]: sending, by the server, the second stream of user interface information to the mobile device to reproduce on the mobile device at least one of the display, audio and the HW action by the HW component of the mobile device.**

105. Yoon discloses or suggests this limitation for the reasons I discuss in Section VII.A.2.d.

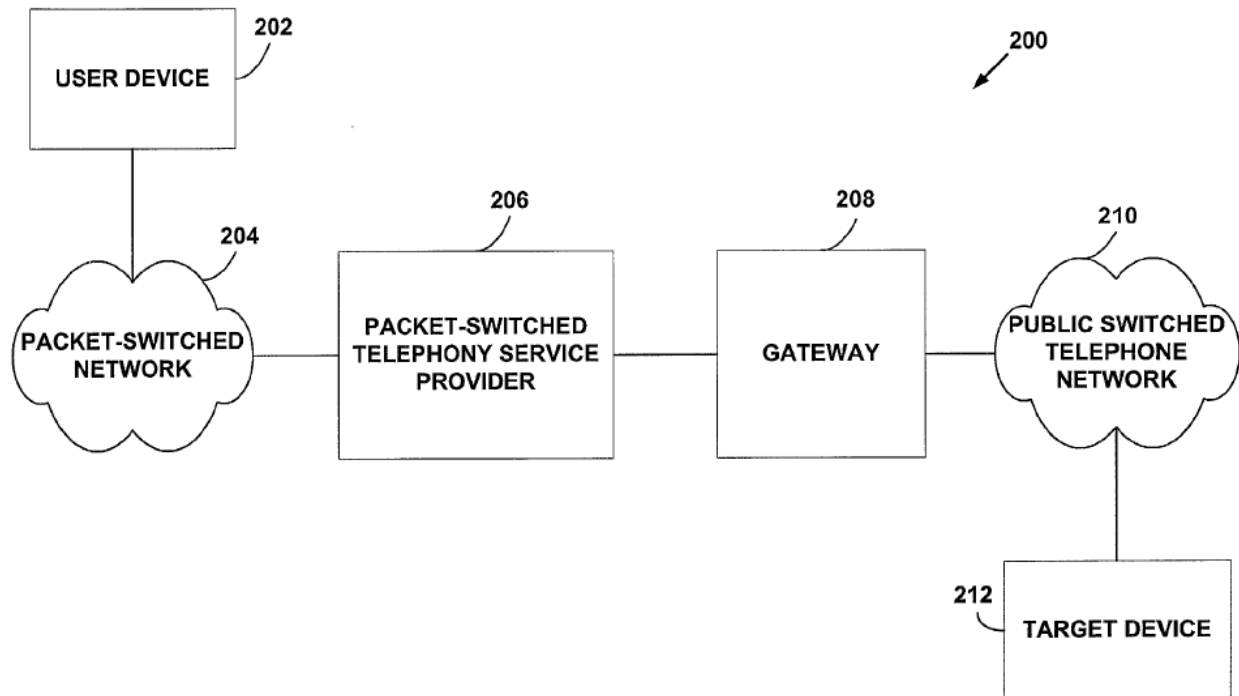
B. Ground 1B: Yoon + Nix

1. Nix

106. Nix teaches a “method for providing packet-switched telephony service.” Nix, Abstract. More specifically, Nix teaches a user device linked to an internet telephony service provider via a patch-switched network such as the internet and a gateway to a public switched telephone network (PSTN) linked to the internet telephony service provider on one end and a PSTN on the other, as illustrated below.

Nix, ¶29, Fig. 2.

FIGURE 2



Nix, Fig. 2

2. Motivation to Combine

107. Yoon discloses that “various web services are provided by a web server.” Yoon, 7. Yoon does not limit the types of web services that can be provided by the web server or provide any specific examples of web services.

108. Nix discloses, “Internet Telephony,” which allows a device connected to the internet to “communicate through the Internet and PSTN to a telephone subscriber at a telephone connected to the PSTN” with the aid of a gateway from the internet to the PSTN. Nix, ¶2. Nix explains that “[n]umerous benefits may be realized through the use of packet-switched telephony,” including cheaper long-distance calls and “convenient interfaces and features [such as]...volume control, a video session, or an address book application.” Nix, ¶3. Nix further provides examples of “handheld device[s] or [] cellular phone[s]” as devices that would increasingly demand internet telephony capabilities. Nix, ¶7.

109. A POSITA would have understood internet telephony to be a type of web service, as mentioned in Yoon. By 2012 (when Yoon was published), internet telephony was widely available on mobile phones and smart phones, such as the terminals in Yoon. For example, Skype was a widely popular service for placing calls to landlines, i.e., a PSTN, from a computer/smart phone. Accordingly, a POSITA would have been motivated to provide a means for internet telephony on Yoon’s emulated terminal/mobile OS/application because it would have been a

common web service that provides users the benefits discussed by Nix. Further, as mentioned by Yoon, providing access to internet telephony via Yoon's web server would also allow access to this feature for user terminals that cannot themselves access it. Yoon, 11 ("Typically...the web service provided by the web server 190 is not able to interwork with an OS/application of the terminal."). Doing so would have combined prior-art elements (Yoon's web server for providing access to web services and Nix's internet telephony, one such web service) according to known methods (providing a gateway to a PSTN at a server, as disclosed by Nix) to yield predictable results (providing internet telephony to Yoon's mobile OS/application). It also would have been no more than the use of a known technique (Nix's internet telephony provided at a server) to improve similar devices/methods (Yoon's web server) in the same way (allowing users to make telephone calls via the internet).

**3. Claim 4: The method of claim 1, further comprising:
providing, by the server, a mobile device gateway for
handling and providing internet access, voice calls and a
gateway to a public switched telephone network (PSTN).**

110. Yoon discloses that the functions of a web server—which provides “various web services” that “may directly provide data through a web browser or a dedicated application”—are integrated into the functions of the virtualization server. Yoon, 7, 16-17. Accordingly, a POSITA would have understood that Yoon's

virtualization server provides *a mobile device gateway for handling and providing internet access.*

111. Nix teaches providing a gateway to a PSTN at a packet-switched telephony service provider. Nix, Fig. 2. Nix discloses transmitting voice data between a user device connected to a packet switched network and a target device connected to a PSTN via a gateway as follows:

According to the exemplary system 200, a user located at the user device 202 may initiate a call to a target user located at the target device 212 by participating in a call initiation process involving the user device 202 and the PTSP 206. For example, the user device 202 may access a web-site maintained by the PTSP 206, using a web-browser, such as Microsoft Internet Explorer or Netscape Navigator. As another example, the user device 202 may execute a calling application to contact the PTSP 206 through the packet-switched network 204. The call initiation process may include choosing a phone number or address book entry to call, and may include specifying other call parameters. The PTSP 206 continues the initiation process by transmitting call initiation information to the gateway 208 so that the gateway 208 may attempt to reach the target device 212 through the public switched telephone network 210. If the call initiation process is successful, voice communications may be transmitted back and forth between the user device 202 and the target device 212.

Nix, ¶33. A POSITA would have understood that Nix teaches *a mobile device gateway (the packet-switched telephony service provider) that handles voice calls, and a gateway to a PSTN.*

112. A POSITA would have been motivated to provide the function of the packet-switched telephony service provider described in Nix to the web server (and thus the virtualization server) of Yoon for the reasons I discuss in Section VII.B.2. In this combination, where Yoon's web server (and thus virtualization server) provides internet telephony, among other web services, the virtualization server also provides *voice calls and a gateway to a PSTN*.

- 4. Claim 14: The method of claim 11, further comprising: providing, by the server, a mobile device gateway for handling and providing internet access, voice calls and a gateway to a public switched telephone network (PSTN).**

113. The combination of Yoon and Nix renders obvious this claim for the reasons I discuss in Section VII.B.2.

- 5. Claim 22: The method of claim 20, further comprising: providing, by the server, a mobile device gateway for handling and providing internet access, voice calls and a gateway to a public switched telephone network (PSTN).**

114. The combination of Yoon and Nix renders obvious this claim for the reasons I discuss in Section VII.B.2.

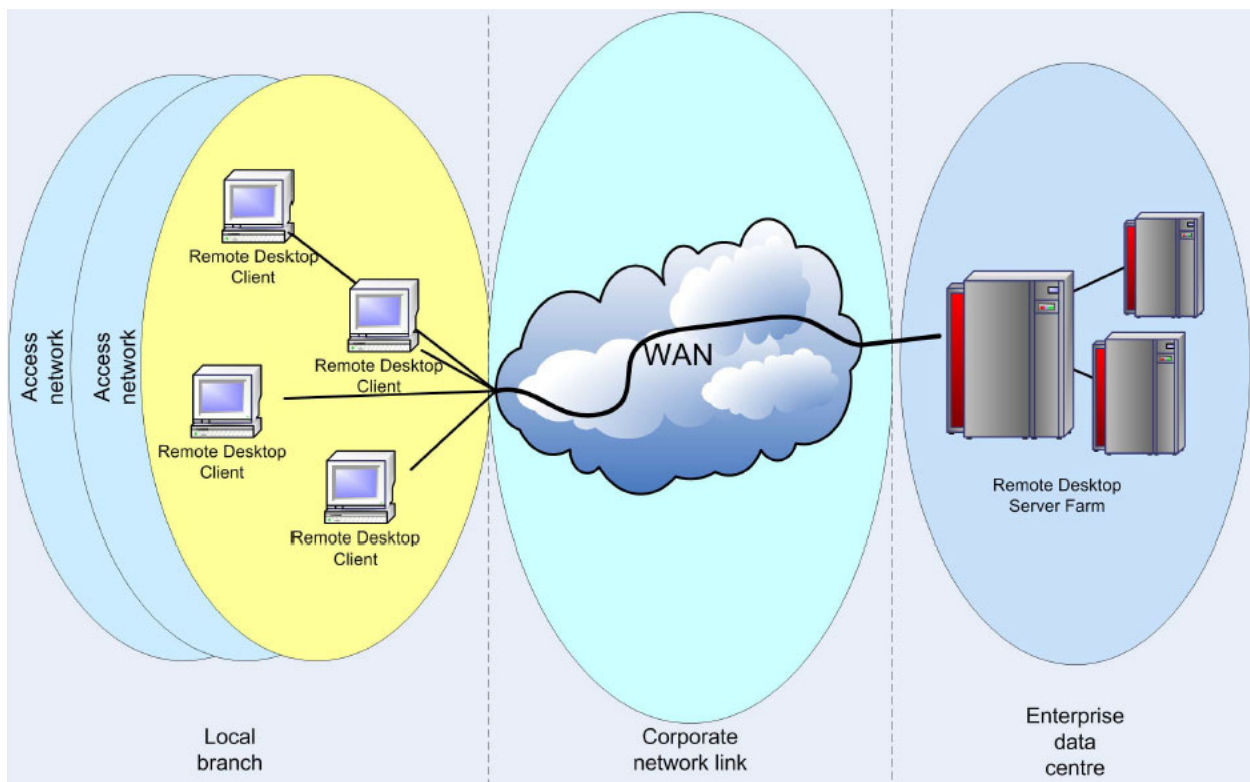
C. Ground 1C: Yoon + Lubonski + Richardson

1. Lubonski

115. Lubonski teaches a "server-side component of the architecture for [a] remote desktop environment aiming at improving user perception of the more demanding and network sensitive multimedia applications and services." Lubonski,

Abstract. Lubonski specifically aims to solve “significant challenges when working with more demanding data flows such as multimedia streams.” *Id.* The “component maps [] user-perceived Quality of Service (QoS) onto network-level performance parameters in order to improve overall end-to-end QoS.” *Id.* Lubonski explains that its component “can optimize the usage of the available network resources.” *Id.* In my opinion, Lubonski recognizes and proposes a solution to a problem that is not unique to remote desktop applications – namely optimization of network resources to optimize multi-media streaming.

116. Lubonski is directed to a remote desktop environment where an end-user terminal connects, via a network, to a server. Lubonski, 152; Figure 1.



Lubonski, Fig. 1

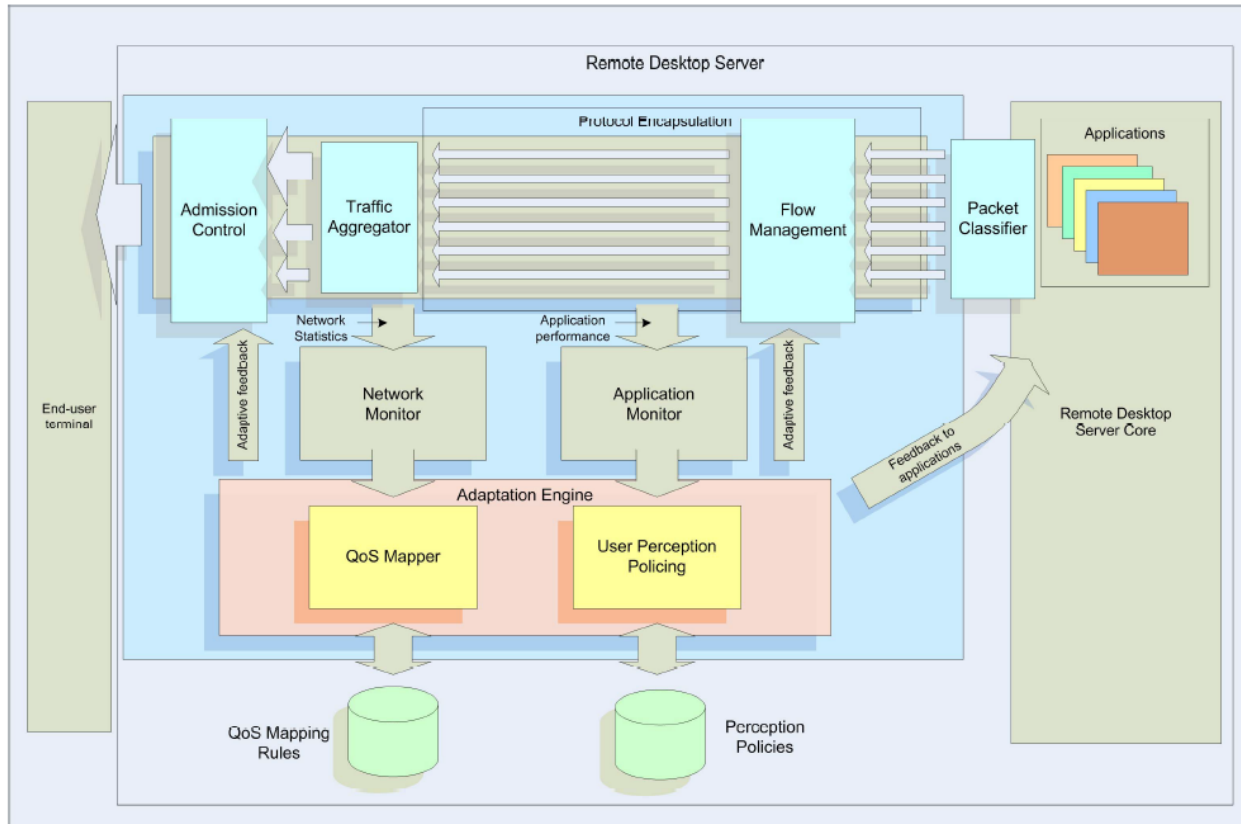
117. “The effect of a virtual desktop is achieved by capturing screen updates by the server and encapsulation into remote desktop protocol. The protocol sends the desktop updates to the client and transmits back user input.” Lubonski, 152. Lubonski recognizes challenges associated with streaming multi-media content between a server and a client:

Systems used nowadays in enterprise environment applications might combine many different types of content, (such as textual data, multimedia streams or print spooling information). This leads to a more complex situation where in the same connection from client to server there is a multitude of flows, each with its specific network characteristics and requirements (for instance, Citrix [22]). Those requirements can range from time- insensitive data transfers such as WWW or FTP to the real-time transmissions of voice or video. In result even application protocols having some internal adaptation to current network conditions combined with other type of traffic can experience a decrease and not an improvement of performance. In more complex interaction among different flows and heavily used network this can lead even to total decrease of quality.

Lubonski, 152.

118. Lubonski teaches “mapping of user-perception feedback to dynamically adapt resource parameters and[,] in result[,] improve the overall system

performance.” Lubonski, 154. Lubonski accomplishes this with a server extension, depicted below in Figure 2. Lubonski, 154, Figure 2:



Lubonski, Fig. 2

119. Lubonski’s extension “adjust[s] resource allocation in such a way that user-perceived QoS levels for all flows [between a server and end-user terminal] are preserved.” Lubonski, 155. More specifically:

In the first step the parameters of the user connection are modified; for instance allocating more bandwidth to particular data flow if possible. Failing that, the admission control would try to modify parameters of the flow aggregate to which this data flow pertains in order to meet requested requirements. Finally, if that fails the feedback would be

provided to the application and if any adaptation mechanisms within application itself are available, they would have a chance to compensate for with the current network state. Within this procedure the adaptation component would verify influence of possible changes on other data flows and assures that none of the critical applications are disadvantaged.

Lubonski, 155. To do this, Lubonski teaches a “Packet Classifier

To perform per-flow operations there has to be means to distinguish between flows. In one of the possible implementations the traffic shaping mechanism could use information provided by Packet Classifier – responsible for the classification and marking of each application data flow – to reflect its traffic characteristics and user-perceived QoS requirements.

Lubonski, 155.

2. Richardson

120. Richardson is a textbook that describes itself as a “practical, accessible and unbiased guide to the H.264 video compression standard.” Richardson, Preface. Relevant here, Richardson describes the transport of H.264 data. *Id.*, 244-248. Specifically, while H.264 itself “does not define a transport mechanism for coded visual data” (*id.*, 245), a well-known transport protocol used for H.264 data is Real-Time Protocol (“RTP”). *Id.*, 246. Richardson explains:

RTP defines a packet structure for real-time data (Figure 8.31) that includes a type identifier to signal the type of CODEC used to generate

the data, a sequence number, essential for re-ordering packets that are received out of order, and a time stamp, necessary to determine the correct presentation time for the decoded data. Transporting a coded audio-visual stream via RTP involves packetizing each elementary stream into a series of RTP packets, interleaving these and transmitting them across an IP network.

Id., 246.

121. A simplified RTP packet structure is provided in Figure 8.31:

	Payload Type	Sequence Number	Timestamp	Unique Identifier	Payload (e.g. Video Packet)
--	--------------	-----------------	-----------	-------------------	-----------------------------

Figure 8.31 RTP packet structure (simplified)

Richardson, Fig. 8.31

3. Motivation to Combine

122. Yoon’s virtualization server encodes a “plurality of virtual environment images into a specific video format such as H.264 and may transmit the same to the terminal 150 in real time.” Yoon, 15. Yoon’s virtualization server also encodes a “plurality of time-based sounds into a specific video format such as H.264 and may transmit the same to the terminal 150 in real time.” *Id.* While Yoon notes that transmitting images and sound as video may be more “efficient” (*id.*), Yoon does not provide further teaching regarding efficient data transmission between the virtualization server and terminals, nor does Yoon address the QoS of such transmissions.

123. A POSITA would have understood that QoS is vital to user experience when streaming multimedia such as video and audio. For example, Rinne recognizes that “QoS is of particular concern for the continuous transmission of high-bandwidth video and multimedia information, i.e. normally a high QoS is needed for real-time transmissions.” Rinne, ¶ 15. Similarly, Talaat recognizes that “audio/video streaming applications are becoming very popular” and discusses various network-level and application-level QoS solutions. Talaat, 266.

124. A POSITA would have further understood that users may value different QoS for different media types, depending on the application. For example, Talaat explains that in some cases, video is more important than audio, and in others, the reverse is true:

For example in distant learning applications, the content being streamed can be educational lectures given by a lecturer in a lecture theater. The video stream of this content will convey only the lecturers face, mouth motion, and facial gestures, while the audio stream content is the real semantically relevant stream for the students receiving this session. The audio stream of such audio/video session conveys the scientific material intended to be delivered to students. Another example would be if this lecturer decided to show his students a surgical operation he is making, while the students are watching it remotely through video streaming receiving applications, in this case the streams importance is reversed with respect to the previous one.

Talaat, 266.

125. A POSITA implementing Yoon's teachings would therefore have found it worthwhile to improve QoS for the user and would have looked to the teachings of Lubonski for this improvement.

126. Specifically, a POSITA would have been motivated to implement Lubonski's server extension that dynamically adapts resource parameters to maximize user-perceived QoS for multimedia transmitted from Yoon's virtualization server. A POSITA would have recognized that Lubonski's extension, which improves user-perceived QoS for video and audio streaming between a server and client, would work for a virtualization server and terminals connected over a network as taught by Yoon. A POSITA would have also understood that users perceive some changes in quality to be more significant than others (Lubonski, 155, *see also* Talaat, 266), and thus would have been motivated to use Lubonski's extension to improve user experience for any given set of network resources by reserving resources, where possible, to QoS elements most valued by the user.

127. To implement Lubonski's extension in Yoon's server, a POSITA would also have needed to understand how multimedia data is transported so that Lubonski's Packet Classifier could correctly distinguish between media types. Because Yoon teaches encoding video and audio using H.264 (Yoon, 15) but provides no detail on how H.264 data is transported, a POSITA would have looked

to an authoritative source like Richardson for that information. In doing so, a POSITA would have seen that RTP is one of two commonly used transport protocols used for H.264 data. One would have been motivated to use RTP as one of a finite number of identified, predictable solutions, with a reasonable expectation of success. Richardson, 246. Indeed, in my experience RTP is among the most commonly used, if not the most commonly used transport protocol for low-latency video and audio streaming.

128. When H.264 data is transmitted with RTP, each data packet includes a type identifier, which I have annotated in Richardson's figure 8.31 below. Richardson, 246, Fig. 8.31.

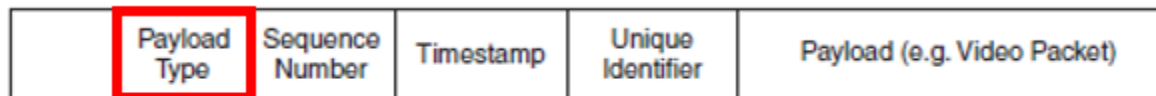


Figure 8.31 RTP packet structure (simplified)

Richardson, Fig. 8.31 (annotated)

129. A POSITA would have understood that this type identifier could be used by Lubonski's Packet Classifier to distinguish between, e.g., video and audio flows because it distinguishes the type of codec used to generate the data in a packet, and thus type of data in said packet. Richardson, 246; Lubonski, 155, Figure 2. A POSITA would have therefore been motivated to use RTP (as taught by Richardson) as the protocol for H.264 multimedia transport between Yoon's virtualization

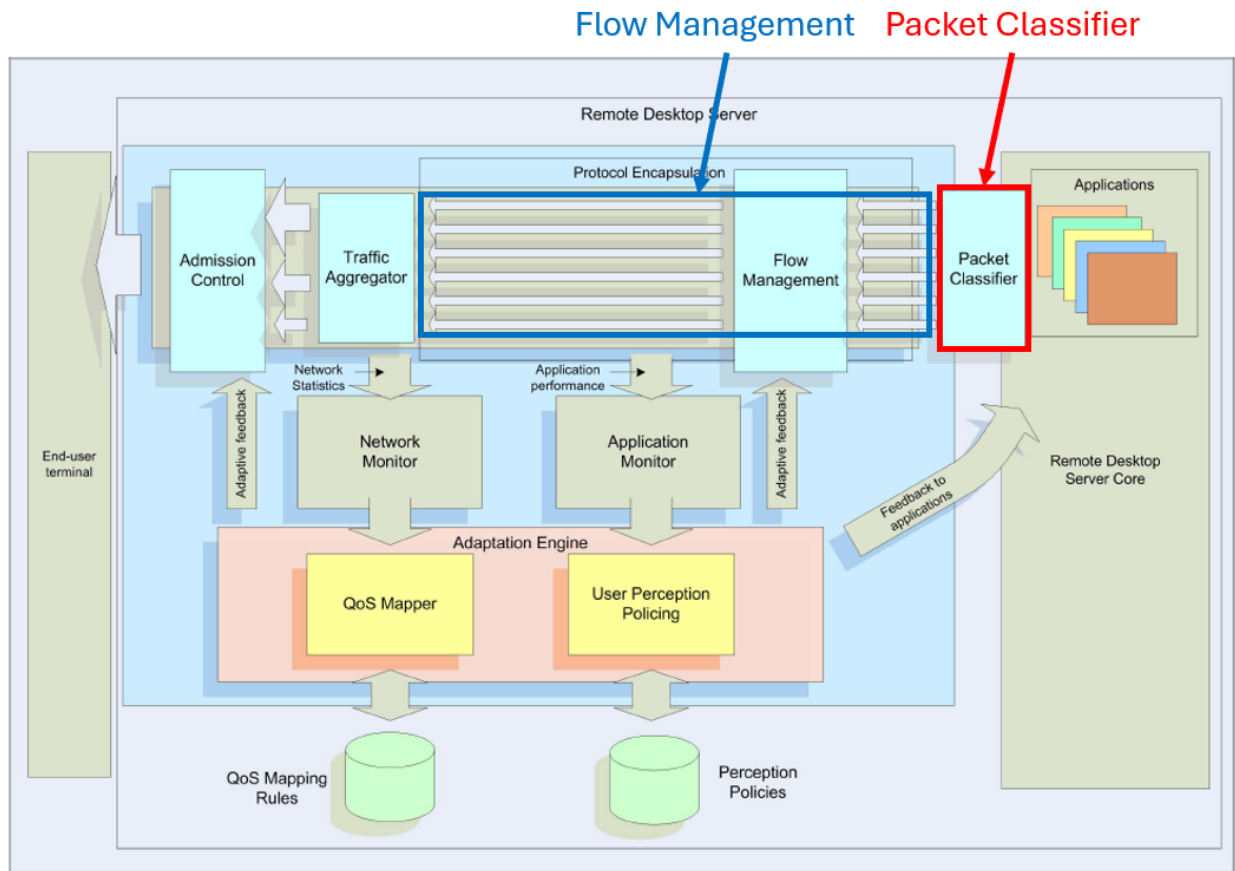
server—with Lubonski’s extension for improving user user-perceived QoS—and Yoon’s terminals. Doing so would have been no more than use of a known technique (RTP transport of H.264 data and Lubonski’s server extension component) to improve a similar method (multimedia transport between a server and terminals) that was ready for improvement (Yoon’s mention, without further teaching, of transmitting H.264 video between its virtualization server and user terminals) to yield predictable results (better user-perceived QoS).

4. Claim 6: The method of claim 1, wherein the stream of user interface information comprises a plurality of different types of data and the server sends each different type of data to the mobile device in a separate data stream.

130. The combination of Yoon, Lubonski, and Richardson renders obvious this claim.

131. Yoon discloses sending a plurality of different data types (e.g., images and audio) in a media stream from the virtualization server to terminal(s). Yoon, 14-15.

132. Lubonski’s server extension, depicted in Figure 2, which I have annotated below, comprises a Packet Classifier [red] that “classif[ies] and mark[s] [] each application data flow - to reflect its traffic characteristics and user-perceived QoS requirements.” Lubonski, 155, Figure 2. I have annotated the separate data flows in blue:



Lubonski, Fig. 2 (annotated)

133. Lubonski's extension then adjusts resource allocation among the data flows discerned by the Packet Classifier based on user-perceived QoS parameters. Lubonski, 155.

134. Where RTP is used for multimedia transport (as explained in Richardson), the video and audio data produced by Yoon would be transported via packets indicating a different payload type. Richardson, 246, Figure 8.31. Specifically, Richardson explains that the RTP payload format "signal[s] the type of CODEC used to generate the data" and that "RTP payload formats are defined for

various standard video and audio CODECs, including H.264.” Richardson, 246. Accordingly, Lubonski's Packet Classifier would classify video and audio packets differently and separate them into different data flows so that user-perceived QoS of each data flow could be adjusted separately. For example, in many applications streamed video may be more important to users than the accompanying audio, and network resources for the video data flow would be prioritized. *See also*, Lubonski, 155; Talaat, 266.

135. Finally, a POSITA would have understood each of these individually adjusted data flows for video and audio to be *a separate data stream* because each flow represents an independent sequence of data that is handled separately from the others. Lubonski, 150 (“In our approach, the data flows of different characteristics - corresponding to different types of applications or services - within one client connection are being dynamically shaped to the current network conditions and user-perceived performance of remote applications and services.”).

5. Claim 7: The method of claim 6, wherein at least one of said separate data streams comprises a quality of service different from a quality of service of another of said separate data streams.

136. The combination of Yoon, Lubonski, and Richardson renders obvious this claim. Lubonski teaches adjusting resource allocation among data flows based on user-perceived QoS. Lubonski, 155. Lubonski further teaches that, e.g., the QoS

of video is particularly important to users. *Id.* (“analysis of temporal and spatial parameters of video streaming applications...showed significant deterioration in user-perceived performance when the number of displayed frames per second or color depth was reduced below some threshold.”); Talaat, 266 (a video stream is more important than the associated audio stream in some cases, e.g, demonstrating a procedure via video). Thus, a POSITA would have found it obvious to provide a different, higher, QoS for Yoon’s video data flow than for Yoon’s audio data flow.

6. Claim 8: The method of claim 6, wherein at least one of said separate data streams comprises an encoding method and a bandwidth different from an encoding method and a bandwidth of another of said separate data streams.

137. The combination of Yoon, Lubonski, and Richardson renders obvious this claim. As I explain in Sections VII.C.3 and VII.C.4, Yoon discloses encoding and transmitting video and audio from the virtualization server to terminal(s). While Yoon provides an example of encoding both images and sounds using the H.264 encoding standard (Yoon, 14-15), a POSITA would have understood that H.264 does not provide an encoding method for audio, making it impossible to use the same encoding method for both video and audio. Richardson, 246 (explaining that there are defined RTP payload formats for various standard video and audio CODECS, meaning that audio and video are encoded using different CODECs). A POSITA would therefore have understood that video and audio sent from Yoon’s

virtualization server would comprise different CODECs (and thus different encoding methods).

138. Further, Lubonski teaches that adjusting parameters of a connection to improve user-perceived QoS comprises “allocating more bandwidth to [a] particular data flow if possible.” Lubonski, 155. And as explained in Section VII.C.5, a POSITA would have found it obvious to allocate more resources (and thus bandwidth) to the video data flow than the audio data flow.

7. Claim 11

a. 11[pre]: A method for emulating a mobile device at a server, the method comprising:

139. Yoon discloses or suggests the preamble for the reasons I discuss in Section VII.A.2.a.

b. 11[a]: receiving, by the server, input information from at least one mobile device;

140. Yoon discloses or suggests this limitation for the reasons I discuss in Section VII.A.2.b.

- c. **11[b]: processing, by a mobile device operating system (OS) executed at the server, the input information to produce a stream of user interface information for use in reproducing at the mobile device at least one of a display, audio and a hardware (HW) action by a HW component of the mobile device based on a same instance of the mobile device OS; and**

141. Yoon discloses or suggests this limitation for the reasons I discuss in Section VII.A.2.c.

- d. **11[c]: sending, by the server, the stream of user interface information to the mobile device, wherein the stream of user interface information from the same instance of the mobile device OS comprises a plurality of different types of data and the server sends each different type of data to the mobile device in a separate data stream for reproduction on the mobile device of at least one of the display, audio and the HW action by the HW component of the mobile device.**

142. The combination of Yoon, Lubonski, and Richardson renders obvious this limitation. As I discuss in Section VII.A.2.d, Yoon discloses “sending, by the server, the stream of user interface information to the mobile device...to the mobile device...for reproduction....” As I discuss in Section VII.C.4, the Yoon-Lubonski-Richardson combination teaches that the “user interface information...comprises a plurality of different types of data and the server sends each different type of data to the mobile device in a separate data stream.”

8. **Claim 12: The method of claim 11, further comprising: emulating, by the server, mobile device HW and executing the mobile device OS on the emulated mobile device HW.**

143. The combination of Yoon, Lubonski, and Richardson renders obvious this claim for the reasons I discuss in Section VII.A.3.

9. **Claim 13**

- a. **13[a]: The method of claim 11, further comprising: executing, by the server, at least one application on the mobile device OS;**

144. The combination of Yoon, Lubonski, and Richardson renders obvious this claim for the reasons I discuss in Section VII.A.4.a.

- b. **13[b]: wherein the at least one application processes the input information to produce the stream of user interface information.**

145. The combination of Yoon, Lubonski, and Richardson renders obvious this claim for the reasons I discuss in Section VII.A.4.b.

10. **Claim 15: The method of claim 11, further comprising: receiving the input information from a plurality of different mobile devices, wherein a same instance of the mobile device OS processes the input information from the plurality of different mobile devices to produce the stream of user interface information.**

146. The combination of Yoon, Lubonski, and Richardson renders obvious this claim for the reasons I discuss in Section VII.A.5.

- 11. Claim 16: The method of claim 11, wherein at least one of said separate data streams comprises a quality of service different from a quality of service of another of said separate data streams.**

147. The combination of Yoon, Lubonski, and Richardson renders obvious this claim for the reasons I discuss in Section VII.C.5.

- 12. Claim 17: The method of claim 11, wherein at least one of said separate data streams comprises an encoding method and a bandwidth different from an encoding method and a bandwidth of another of said separate data streams.**

148. The combination of Yoon, Lubonski, and Richardson renders obvious this claim for the reasons I discuss in Section VII.C.6.

- 13. Claim 23: The method of claim 20, wherein the stream of user interface information comprises a plurality of different types of data and the server sends each different type of data to the mobile device in a separate data stream.**

149. The combination of Yoon, Lubonski, and Richardson renders obvious this claim for the reasons I discuss in Section VII.C.4.

- 14. Claim 24: The method of claim 23, wherein at least one of said separate data streams comprises a quality of service different a quality of service of another of said separate data streams.**

150. The combination of Yoon, Lubonski, and Richardson renders obvious this claim for the reasons I discuss in Section VII.C.5.

- 15. Claim 25: The method of claim 23, wherein at least one of said separate data streams comprises an encoding method and a bandwidth different from an encoding method and a bandwidth of another of said separate data streams.**

151. The combination of Yoon, Lubonski, and Richardson renders obvious this claim for the reasons I discuss in Section VII.C.6.

D. Ground 2A: Overton

1. Overton

152. Overton teaches “emulating computing devices over a network” to allow “customers to use applications running remotely on emulated computing devices” because customers “may wish to try the application before buying it.” Overton, 1:54-65. As illustrated in Overton’s Figure 1, which I reproduce below, Overton teaches a computing device 103 (e.g., a server) communicating with one or more clients 106. Overton, 2:24-36, Fig. 1. The computing device 103 can execute an emulation server application 115, and a plurality of wrappers 118, applications 119, and media encoders 120. Overton, 2:57-63. “The emulation server application 115 is executed to launch applications 119, which are executed within the wrappers 118. The emulation server application 115 is also executed to obtain application input data 122 from the clients 106 and provide the application input data 122 to the respective wrapper 118.” Overton, 2:63-3:3. The emulation server application 115 also “send[s] application output data 123 that is captured from the application 119 to the clients 106.” Overton, 3:4-6.

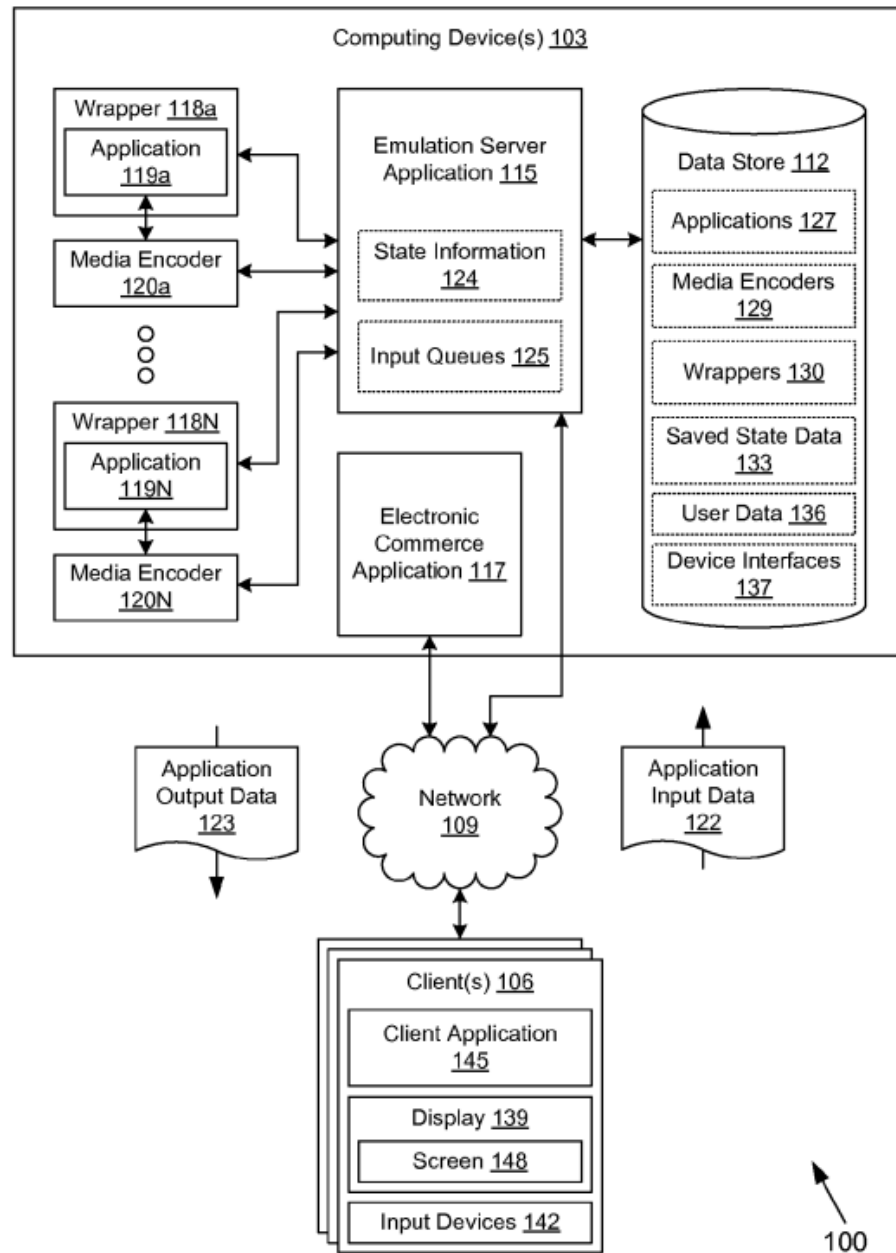


FIG. 1

153. Overton further discloses associated methods for emulating an application at its computing device 103. E.g., Overton, 11:52-12:46, Fig. 3.

2. Claim 1

a. 1[pre]: A method for emulating a mobile device at a server, the method comprising:

154. Overton teaches a method by which its computing device 103 (*a server*) *emulates* a computing device at the request of a client 106 (*a mobile device*). Overton, 11:52-12:46, *see also, id.*, 2:34-36 (“The computing device 103 may comprise, for example, a server computer or any other system providing computing capability.”), 4:45-49 (“where the application 119 is configured for execution in a specialized device or another type of computing device, the wrapper 118 may include an emulation application that emulates the device.”). Overton also provides examples of mobile client devices, which include “a laptop computer, personal digital assistants, cellular telephones, [and] smartphones.” Overton, 6:8-10.

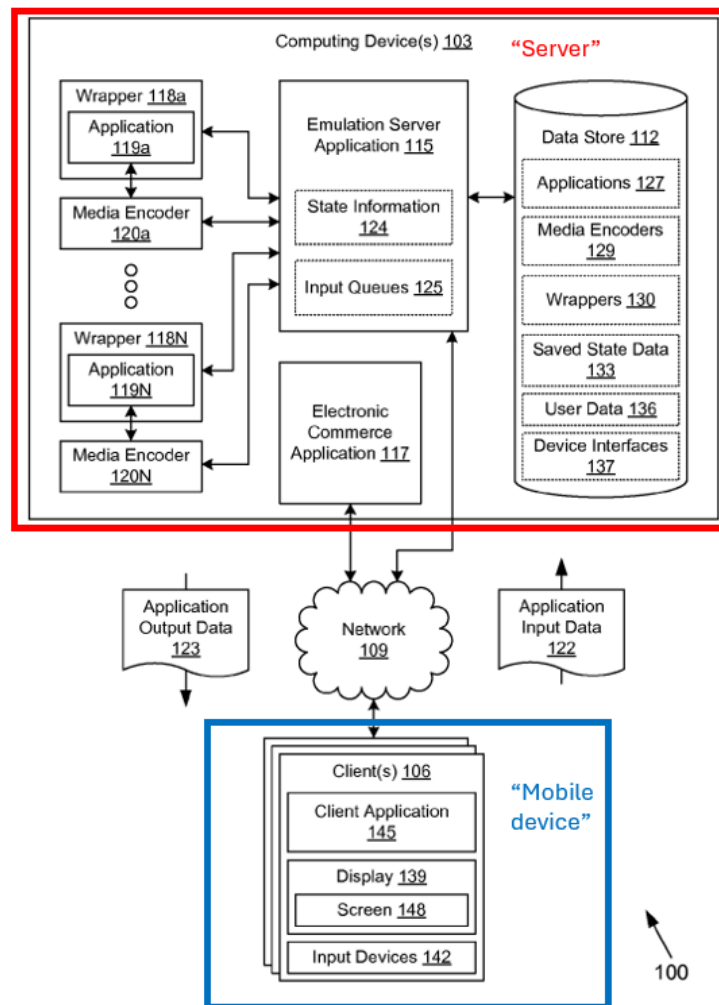


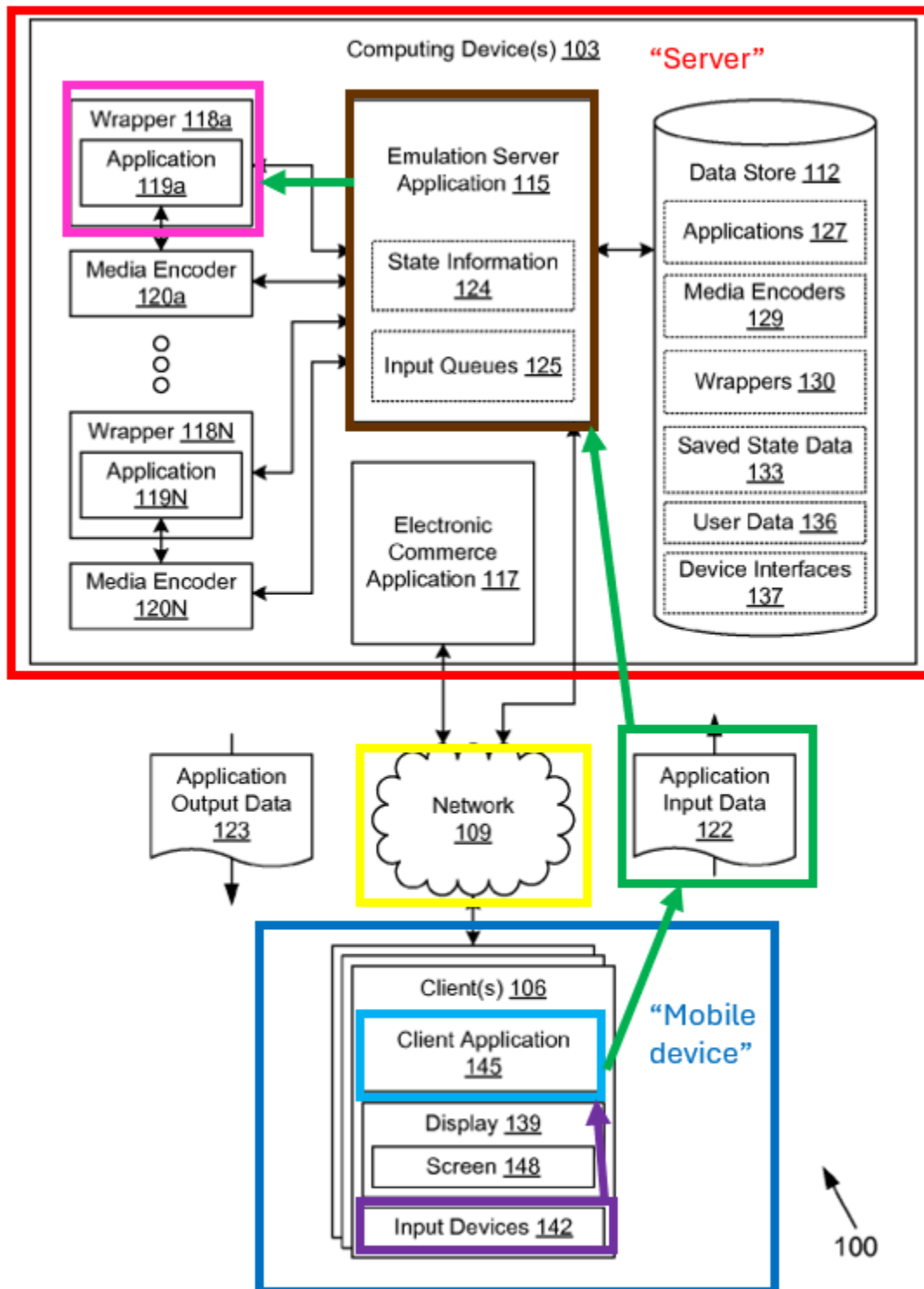
FIG. 1

Overton, Fig. 1 (annotated)

- b. 1[a]: receiving, by the server, input information from at least one mobile device, wherein the input information has not been processed to produce a display, audio and a hardware (HW) action by a HW component of the mobile device;

155. As illustrated in Figure 1, which I have annotated below, Overton discloses that client 106 [blue] executes a client application 145 [teal] that is “configured to capture input commands provided by the user through one or more

of the input devices 142 [purple] and send this input over the network 109 [yellow] to the computing device 103 [red] as application input data 122 [green].” Overton, 6:28-36. The computing device 103’s [red] (*the server*) emulation server application 115 [brown] “obtain[s] application input data 122 from clients 106 and provide[s] the application input data 122 to the respective wrapper 118 [pink].” Overton, 2:67-3:3. In other words, Overton’s computing device receives input information from clients.



Overton, Fig. 1 (annotated)

156. Overton's client device 106 may be, e.g., a laptop, PDA, cell phone, or smart phone, each of which is be *a mobile device* because it can easily be moved and used in transit. Overton, 6:4-14. Thus, Overton discloses *receiving, by the server, input information from at least one mobile device*.

157. Additionally, Overton discloses a sequence in which a user provides an input command at a client device, which then encodes²⁹ the command and transmits it to the server, where the server processes the input command to perform a hardware action at the emulated mobile device.

The user at the client 106 enters input commands for the application 119 by use of the input devices 142 of the client 106. As a non-limiting example, the user may depress a left mouse button. Accordingly, the client application 145 functions to encode the input command into a format that may be transmitted over the network 109 within the application input data 122. The emulation server application 115 receives the input command, adds it to the input queue 125 for the application 119, and ultimately passes it to the wrapper 118. The wrapper 118 then provides a left mouse button depression to the application 119 by way of a virtualized mouse.

²⁹ Encoding does not encompass processing inputs “to produce a display, audio and a hardware (HW) action.” ’887Pat, 3:25-36 (mobile device encodes inputs for transmittal to server).

Overton, 8:64-9:7; *see also, id.*, 4:8-20. Because the user input is processed at the server to produce an action at the emulated application, a POSITA would have understood that the user input is not processed by the client device to produce a display, audio, or HW action (*the input information has not been processed to produce a display, audio and a hardware (HW) action by a HW component of the mobile device*).

- c. **1[b]: processing, by a mobile device operating system (OS) executed at the server, the input information to produce a stream of user interface information for use in reproducing at the mobile device at least one of the display, the audio and the hardware (HW) action by the HW component of the mobile device based on a same instance of the mobile device operating system; and**

158. Overton discloses that application 119 executed at computing device 103 (*the server*) corresponds to an operating system and associated applications.

The applications 119 may also correspond to mobile phone applications, computer-aided design (CAD) applications, computer-aided manufacturing (CAM) applications, photo manipulation applications, video editing applications, office productivity applications, **operating systems and associated applications**, emulators for operating systems, architectures, and capabilities not present on a consumer device, and other applications and combinations of applications.

Overton, 3:59-67 (emphasis added). The application 119 is executed by wrapper 118 (Overton 2:65-67), which, as discussed in Section VII.D.2.b, receives the application input data 122 and processes it. And as I explain in Section VII.D.2.a, Overton’s server emulates a mobile device, so a POSITA would have understood that Overton’s *mobile device operating system (OS) executed at the server* processes the input information.

159. Overton further discloses that the application 119 generates a video and/or audio signal—i.e., application output data 123 [orange]. Overton, 3:4-6, 4:21-24 (“Further, the wrapper 118 is able to obtain a video signal and/or audio signal generated by the application 119 as if the wrapper 118 emulates a display device, an audio device, or another type of output device.”). This application output data is sent by the server to client 106 (*mobile device*). Overton, 3:4-6. A POSITA would have understood that this application output data is produced, at least in part, by processing the application input data received from the client device. This is how a user may try an application on the server before buying it—i.e., Overton’s stated goal. Overton, 1:54-58 (“The present disclosure relates to emulating computing devices over a network. Customers may wish to purchase applications for various computing devices over a network. In order to make an informed purchasing decision, a customer may wish to try the application before buying it.”).

160. The video and/or audio generated at the server is encoded into a media stream (Overton 4:28-32) and sent to client device 106 (*mobile device*) via a network (Overton, 6:41-44). The client device then renders the video on display 139 [black] and plays the audio (*reproducing ... at least one of the display, the audio and the hardware (HW) action by the HW component of the mobile device*). Overton, 6:41-46, Fig. 1 (annotated);

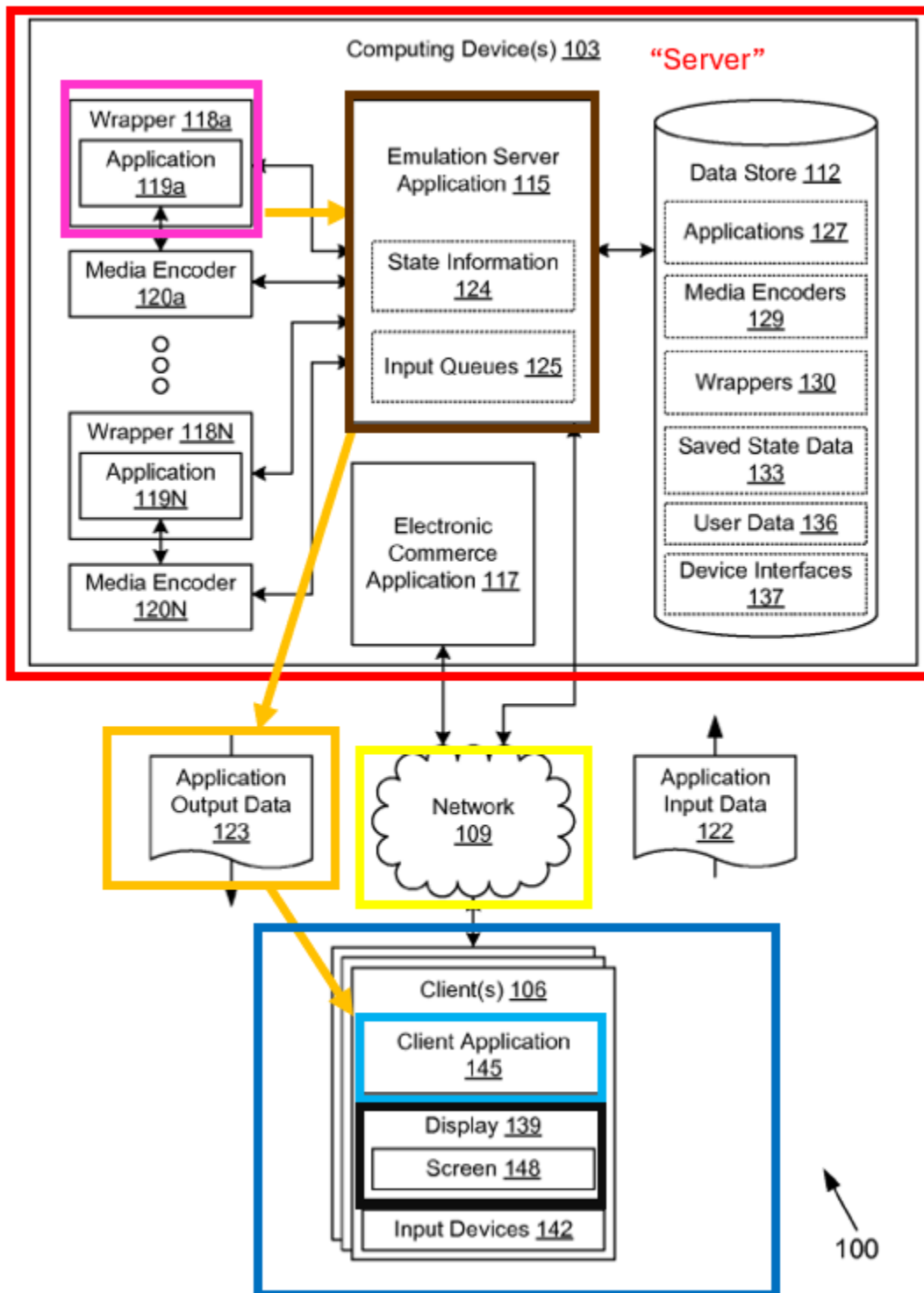


FIG. 1

Overton, Fig. 1 (annotated)

161. Finally, because the same wrapper 118 and application 119 in Overton provide both the video and audio stream, a POSITA would have understood that the streams are *based on a same instance of the mobile device operating system*. As the Applicant argued during prosecution to overcome a rejection, this limitation requires that multiple data streams be provided by a single emulation of the mobile device. '887FH, 198 (“Put differently, applicants’ claimed invention provides multiple data streams (e.g., audio, video, compressed text, voice and/or sensors) arriving from a single mobile device emulation.”).

- d. 1[c]: sending, by the server, the stream of user interface information to the mobile device to reproduce at the mobile device at least one of the display, the audio and the HW action by the HW component of the mobile device.**

162. Overton discloses that computing device 103 (*the server*) sends the media stream (*user interface information*) to a client device 106 (*mobile device*). Overton, 4:21-39 (“wrappers 118 may communicate directly with the clients 106 to...serve up the application output data 123.”). The client device receives the media stream and renders video and audio from the stream on the device (*reproduce at the mobile device at least one of the display, the audio and the HW action by the HW component of the mobile device*). Overton, 6:41-46 (“The client application 145 is also configured to obtain application output data 123 over the network 109 from the computing device 103 and render a screen 148 on the display 139. To this end, the

client application 145 may include one or more video and audio players to play out a media stream generated by an application 119.”).

**3. Claim 2: The method of claim 1, further comprising:
emulating, by the server, mobile device HW and executing
the mobile device OS on the emulated mobile device HW.**

163. Overton discloses that the server emulates mobile device hardware—e.g., “the wrapper 118 may include an emulation application that emulates the device.” Overton, 4:46-49, 12:16-19 (“the emulation server application 115 loads the application 127 and execute an instance of an application 119 (FIG.1) within a wrapper 118 (FIG. 1) that corresponds to the emulated computing device.”). And as I explain in Section VII.D.2.c, Overton teaches executing a mobile device OS in the wrapper 118 (*executing the mobile device on the emulated mobile device HW*).

4. Claim 3

**a. 3[a]: The method of claim 1, further comprising:
executing, by the server, at least one application on
the mobile OS;**

164. As I explain in Section VII.D.2.c, Overton’s computing device 103 (*the server*) executes a mobile device OS and associated applications. Overton, 3:59-67.

**b. 3[b]: wherein the at least one application processes
the input information to produce the stream of user
interface information.**

165. As I explain in Section VII.D.2.c, Overton discloses that application 119 executed by computing device 103 (*the server*) processes user input received

from the client device 106 (*the input information*) and *produces the stream of user interface information*. To the extent not explicitly disclosed, a POSITA would have found it obvious that the application processes the input information to produce the stream of user interface information because Overton teaches allowing a user to try applications on the server before purchasing them (Overton, 1:56-59), which would entail the user providing input to the application and receiving a stream of resulting user interface information.

- 5. Claim 5: The method of claim 1, further comprising: receiving, by the server, the input information from a plurality of different mobile devices, wherein a same instance of the mobile device OS processes the input information from each of the plurality of different mobile devices to produce the stream of user interface information.**

166. Overton discloses that “multiple clients 106 may be employed for one or more users to interact with the application 119.” Overton, 6:52-54 (emphasis added). In other words, a singular application 119 at the computer device 103 (*the server*) receives user input from a plurality of different clients 106 (*mobile devices*). And a POSITA would have further understood that the same application 119 (*same instance of the mobile OS*) processes the input information received from the client devices (*see* Section VII.D.2.c) because Overton discloses that computer device 103 may run multiple applications 119, but refers to a single application 119 as interacting with multiple clients. Overton, 6:52-54 (“multiple clients 106 may be

employed for one or more users to interact with the application 119”). Here, “interact” would be understood as sending input and receiving output to/from the application.

6. Claim 20

a. 20[pre]: A method for emulating a mobile device at a server, the method comprising:

167. Overton discloses or suggests the preamble for the same reasons discussed in Section VII.D.2.a.

b. 20[a]: receiving, by the server, input information from a plurality of different mobile devices, wherein the input information has not been processed to produce a display, audio and a hardware (HW) action by a corresponding HW component of the mobile devices;

168. Overton discloses or suggests this limitation for the same reasons discussed in Sections VII.D.2.b and VII.D.5. Overton teaches its computer device 106 receiving input information that has not been processed from a client device (Section VII.D.2.b), and multiple client devices may interact with the same application 119 at the computer device (Section VII.D.5).

- c. **20[b]: processing, by a same instance of a mobile device operating system (OS) executed at the server, the input information from each of the plurality of different mobile devices to produce user interface information for use in reproducing at the different mobile devices at least one of the display, audio and the HW action by the HW components of the different mobile device based on the same instance of the mobile device OS; and**

169. Overton discloses or suggests this limitation for the same reasons discussed in Sections VII.D.2.c and VII.D.5. Overton teaches that the same application 119 processes input information from multiple clients 106 and produces a media stream to send to each of the clients, where the stream will be rendered. Sections VII.D.2.c, VII.D.5.

- d. **20[c]: sending, by the server, the user interface information to at least one of the plurality of different mobile devices.**

170. Overton discloses or suggests this limitation for the same reasons discussed in Sections VII.D.2.d and VII.D.5 because Overton discloses sending the media stream from the computer device 103 to one or more client devices 106 (*at least one of the plurality of different mobile devices*).

7. Claim 21

- a. 21[a]: The method of claim 20, further comprising: executing, by the server, at least one application on the mobile device OS;**

171. Overton discloses or suggests this limitation for the same reasons discussed in Section VII.D.4.a.

- b. 21[b]: wherein the at least one application processes the input information to produce the user interface information.**

172. Overton discloses or suggests this limitation for the same reasons discussed in Section VII.D.4.b.

8. Claim 28

- a. 28[pre]: A method for emulating a mobile device at a server, the method comprising:**

173. Overton discloses or suggests the preamble for the same reasons discussed in Section VII.D.2.a.

- b. 28[a]: receiving, by server, first input information from a mobile device;**

174. As I discuss in Section VII.D.2.b, Overton discloses its computer device 103 (*server*) receiving input information from client device 106 (*mobile device*). Overton further teaches receiving multiple input commands (i.e., at least first and second input information) from a client device. Overton, 6:32-36 (“the client application 145 is configured to capture input **commands** provided by the user

through **one or more of the input devices 142** and send this input over the network 190 to the computing device 103 as application input data 122.”) (emphasis added).

- c. **28[b]: processing, by a first mobile device operating system (OS) executed at the server, the first input information to produce a first stream of user interface information for use in reproducing at the mobile device at least one of a display, audio and a hardware (HW) action by a HW component of the mobile device; and**

175. Overton discloses or suggests this limitation for the same reasons discussed in Section VII.D.2.c.

176. Alternatively, to the extent PO argues that this claim requires “different kinds of mobile device OS’s are executed and not merely multiple instances of the same mobile device OS” (as PO argued during prosecution, Section V.B), Overton discloses or suggests this limitation. Overton discloses or suggests that a user may choose to emulate different devices to test an application before purchasing it. For example, Overton’s Figure 2, which I have annotated below, depicts an application available for purchase that is compatible with two different devices, Device 1 and Device 2:

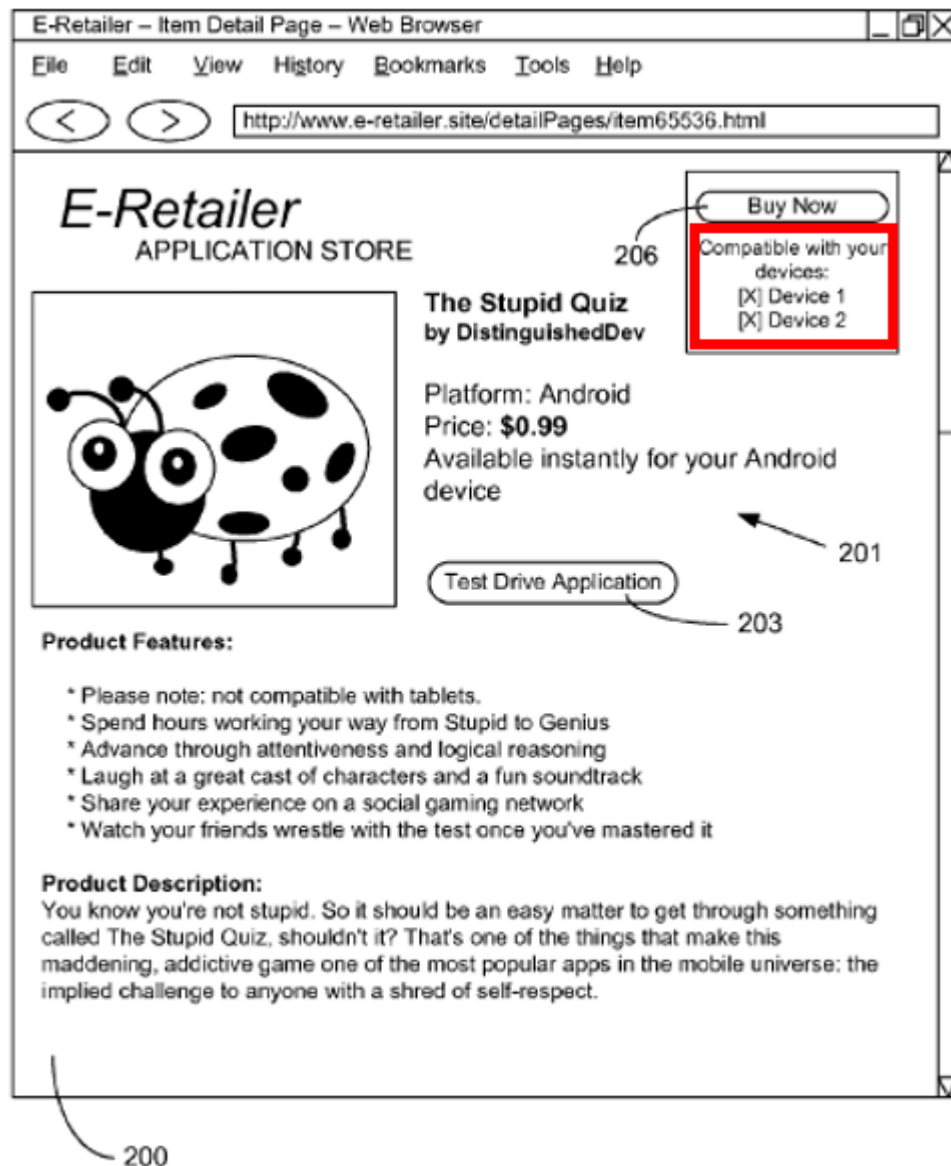


FIG. 2A

Overton, Fig. 2A (annotated)

177. Overton explains that “when a customer activates the emulation request tool 203, the customer is able to use the application 127 in a virtualized environment that resembles the computing device on which it is intended to execute” and provides the “non-limiting example [where] the customer is associated with two computing

devices that are compatible with the application.” Overton, 7:11-22. Overton further discloses that “[t]he customer may specify one or more preferred models of an emulated computing device for which the customer is seeking applications” and that the server emulates a device “specified by the customer at the client 106.” Overton, 8:31-33, 11:66-12:12 (“In box 306, the emulation server application 115 determines a preferred model of the emulated computing device either specified by the customer at the client 106 or otherwise associated with the customer.”). Overton further discloses that the applications executed by computing device 103 may have been originally designed for different types of devices, such as, e.g., a general purpose computer, smartphone, or a handheld game device. Overton, 3:55-59.

178. A POSITA would have understood that different types of devices would likely have different kinds of mobile OS’s. For example, a smartphone and a laptop would typically have different kinds of mobile OS’s. Accordingly, a POSITA would have understood that Overton teaches allowing a user to emulate different kinds of mobile OS’s to try an application on different kinds of devices. For example, a user could try an application on “device 1” as depicted in Figure 2A and afterwards try the application on “device 2,” where these are two different kinds of devices, e.g., a smartphone and a laptop, with different kinds of mobile OS’s emulated by Overton’s server. Overton thus discloses or suggests operating a first kind of mobile OS followed by a second kind of mobile OS at its server.

- d. **28[c]: sending, by the server, the first stream of user interface information to the mobile device to reproduce on the mobile device at least one of the display, audio and the HW action by the HW component of the mobile device;**

179. Overton discloses or suggests this limitation for the same reasons discussed in Section VII.D.2.d.

- e. **28[d]: receiving, by the server, second input information from the mobile device;**

180. Overton discloses or suggests this limitation for the same reasons discussed in Sections VII.A.2.b (the server receiving an input) and VII.D.8.b (the user selecting to emulate two different devices, with different kinds of mobile OS's, one after the other).

- f. **28[e]: processing, by a second mobile device operating system (OS) executed by the server, the second input information to produce a second stream of user interface information for reproducing at least one of a display, audio and a HW action by a HW component of the mobile device, wherein the second mobile device OS has at least one of a different OS type, a different OS version and a different OS instance than the first mobile device OS; and**

181. Overton teaches a client device 106 interacting with multiple (i.e., first and second) applications (*mobile device operating systems*) at the computing device 103, and Overton explains that users can try applications executed on the server before purchasing them. Overton, 1:55-58 (“Customers may wish to purchase **applications** for various computing devices over a network. In order to make an

informed purchasing decision, a customer may wish to try the application before buying it.”) (emphasis added). Overton further teaches that “a plurality of applications 119” are executed by computing device 103. Overton, 2:57-63, Fig. 1. A POSITA would have found it obvious that a user shopping for applications may wish to try out multiple applications before choosing to purchase one (or more) because the first application a user tries may not be to their liking.

182. Overton further teaches that each application emulation session is eventually terminated, either by a timer, user choice, or inactivity.

In box 324, the emulation server application 115 determines whether the session is over. As a non-limiting example, the session may be associated with a timer, and the timer may have expired. As another non-limiting example, the session may have been explicitly ended through user interaction at the client 106. As yet another non-limiting example, the session may have been implicitly ended through user inaction at the client 106.

Overton, 12:32-39; Fig. 3.

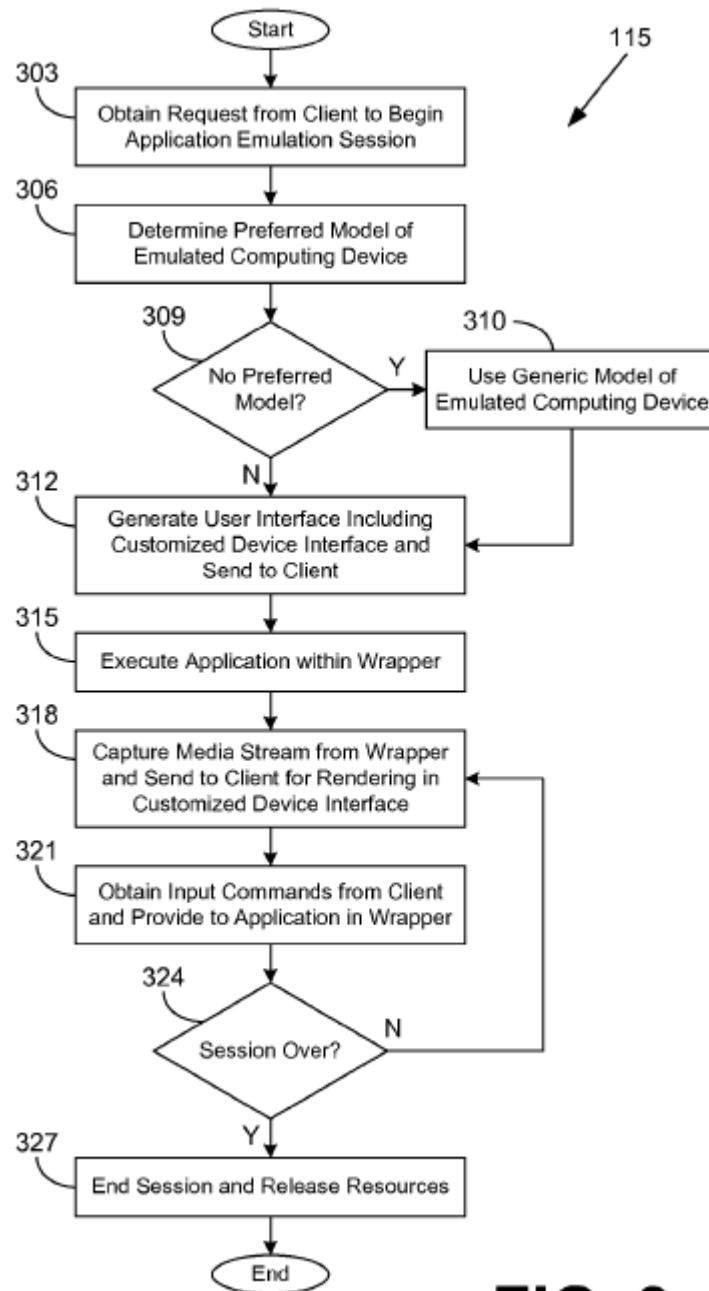


FIG. 3

Overton, Fig. 3

Accordingly, a POSITA would have understood that when a user tries multiple applications, each application would be tried in its own emulation session that is terminated by the user before trying out the next because Figure 3 does not provide

to trying multiple applications in the same emulation session. The second application tried by the user would therefore be “in a different OS instance.”

183. To the extent that PO argues that this claim requires that this claim requires that “different kinds of mobile device OS’s are executed and not merely multiple instances of the same mobile device OS” as it did during prosecution, Overton discloses or suggests this limitation as I discuss in Section VII.D.8.c.

184. Overton discloses the remaining limitations of the claim for the reasons I discuss in Section VII.D.2.c.

- g. 28[f]: sending, by the server, the second stream of user interface information to the mobile device to reproduce on the mobile device at least one of the display, audio and the HW action by the HW component of the mobile device.**

185. Overton discloses or suggests this limitation for the same reasons discussed in Section VII.D.2.d.

E. Ground 2B: Overton + Nix

1. Motivation to Combine

186. A POSITA would have been motivated to connect a gateway to a PSTN, as taught by Nix, to Overton’s computing device 103, acting as a server, to allow users to made voice calls and/or try applications for internet telephony.

187. Nix explains that “[n]umerous benefits may be realized through the use of packet-switched telephony,” including cheaper long-distance calls and

“convenient interfaces and features [such as]...volume control, a video session, or an address book application.” Nix, ¶ 3. Nix further discloses several known applications that provide packet-switched telephony services. Nix, ¶¶ 3 (“each ITSP [internet telephony server provider]...may require a download of client software”), 4 (“The download requirements vary by ITSP, but in general they require an application...”).

188. Overton explains that a goal of its invention is to allow a customer to try an application before purchasing it. Overton, 1:56-58. A POSITA would have understood that users may want to try different applications for packet-switched telephony and would have been motivated to combine the teachings of Nix and Overton to allow users to try them. Specifically, because Overton already teaches a user device communicating with a computing device (server) over a network such as the internet (Overton, 2:24-32, Fig. 1), a POSITA would have been motivated to connect the computing device to a gateway to a PSTN, as taught by Nix, to allow users to try applications for packet-switched telephony. Doing so is no more than combining prior art elements (Overton’s computing device acting as a server and communicating with a mobile device over a network, and Nix’s server communicating with a mobile device over a network and a gateway to a PSTN) according to known methods (providing internet telephony via a server

communicating with a gateway to a PSTN) to yield predictable results (allowing users to try internet telephony applications before purchasing them).

189. Alternatively, a POSITA would have understood that in cases where client devices 106 are cellular telephones or smartphones (together, “phones”), Overton, 6:4-10, a core functionality of phones is to allow the user to place telephone calls. Because Overton’s Figure 1 shows that client devices are only connected to the computing device 103, a POSITA would have been motivated to modify computing device 103 to provide this core functionality for phones that connect to it. Nix teaches one way of providing this functionality to devices connected to a server via a packet-switched network, such as client devices 106 connected to computing device 103.

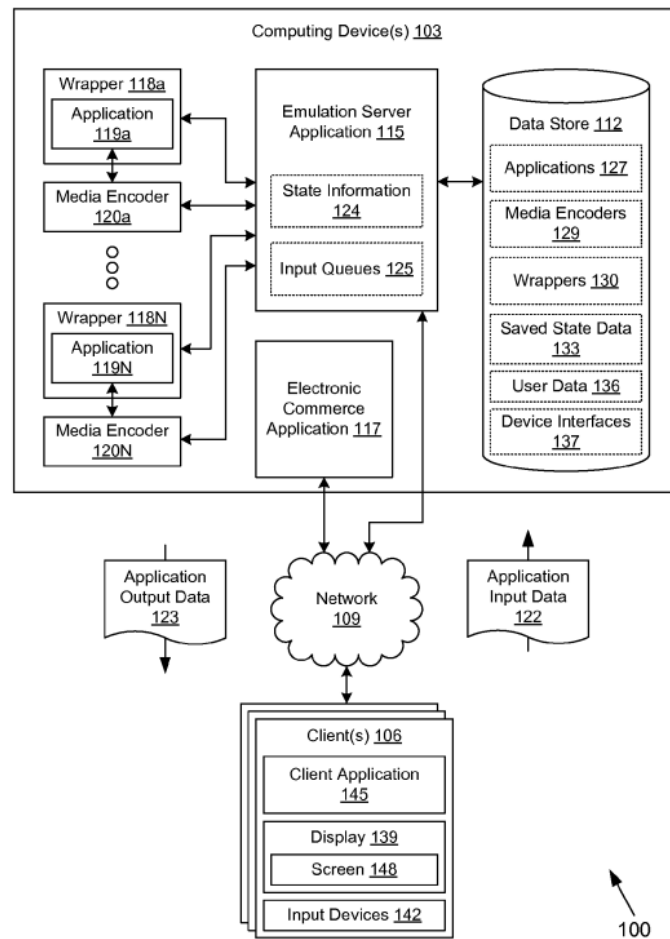


FIG. 1

Overton, Fig. 1

190. A POSITA would have therefore connected a gateway to a PSTN, as taught by Nix, to Overton's computing device 103, in order to provide telephone functionality to phones that connect to computing device 103. Doing so would have amounted to nothing more than combining prior art elements (Nix's gateway to a PSTN and Overton's computing device 103 connected via a network to client devices 106) according to known methods (connecting a gateway to a PSTN to a

server, as taught in Nix) to yield predictable results (providing the ability for Overton's phone client devices to place telephone calls).

**2. Claim 4: The method of claim 1, further comprising:
providing, by the server, a mobile device gateway for
handling and providing internet access, voice calls and a
gateway to a public switched telephone network (PSTN).**

191. Overton teaches allowing users to try applications before purchasing them. Overton, 1:56-58. A POSITA would have understood that at least some such applications (e.g., internet browser applications) would provide internet access to the user. A POSITA would have thus understood that Overton's computing device 103 is a *mobile device gateway for handling and providing internet access* by virtue of allowing the user to try applications that provide internet access.

192. Nix teaches providing a gateway to a PSTN at a packet-switched telephony service provider. Nix, Fig. 2. Nix discloses transmitting voice data between a user device connected to a packet switched network and a target device connected to a PSTN via a gateway as follows:

According to the exemplary system 200, a user located at the user device 202 may initiate a call to a target user located at the target device 212 by participating in a call initiation process involving the user device 202 and the PTSP 206. For example, the user device 202 may access a web-site maintained by the PTSP 206, using a web-browser, such as Microsoft Internet Explorer or Netscape Navigator. As another example, the user device 202 may execute a calling application to

contact the PTSP 206 through the packet-switched network 204. The call initiation process may include choosing a phone number or address book entry to call, and may include specifying other call parameters. The PTSP 206 continues the initiation process by transmitting call initiation information to the gateway 208 so that the gateway 208 may attempt to reach the target device 212 through the public switched telephone network 210. If the call initiation process is successful, voice communications may be transmitted back and forth between the user device 202 and the target device 212.

Nix, ¶33. A POSITA would have understood that Nix teaches *a mobile device gateway* (the packet-switched telephony service provider) *that handles voice calls, and a gateway to a PSTN*.

193. In combining the teachings of Overton and Nix, a POSITA would have been motivated to incorporate Nix's teaching of a packet-switched telephony service provider into one of the applications 119 taught by Overton because, as is disclosed by Nix and was also well-known at the time, various applications that provided packet-switched telephony existed (e.g., "Go2Call.com, Net2Phone, DialPad, Maxcall, AccessPower, and others" such as Skype or Viber). Nix, ¶ 3. A POSITA would have further been motivated to provide, at the computing device 103 (*server*), a gateway to a PSTN, as taught by Nix, to allow users to try these telephony applications.

194. Alternatively, a POSITA would have been motivated to provide the core functionality of making telephone calls to Overton's client devices that are phones, and would have thus connected Nix's gateway to a PSTN to Overton's computing device 103 for the reasons I discuss in Section VII.E.1.

195. Accordingly, Overton and Nix teach providing, by Overton's computing device 103 (*server*) applications for internet access and packet-switched telephony (*mobile device gateway for handling and providing internet access, voice calls*) and a gateway to a PSTN to allow the applications for patch-switched telephony to place said voice calls.

**3. Claim 14: The method of claim 11, further comprising:
providing, by the server, a mobile device gateway for
handling and providing internet access, voice calls and a
gateway to a public switched telephone network (PSTN).**

196. The combination of Overton and Nix renders obvious this claim for the reasons I discuss in Section VII.E.2.

**4. Claim 22: The method of claim 20, further comprising:
providing, by the server, a mobile device gateway for
handling and providing internet access, voice calls and a
gateway to a public switched telephone network (PSTN).**

197. The combination of Overton and Nix renders obvious this claim for the reasons I discuss in Section VII.E.2.

F. Ground 2C: Overton + Lubonski + Richardson

1. Motivation to Combine

198. A POSITA would have been motivated to improve user-perceived QoS for media transmitted from Overton's computing device 103 to client device(s) 106 by implementing the teachings of Lubonski's server extension and Richardson's use of RTP for transmitting H.264 data.

199. Overton discloses various encoders for the video and/or audio stream sent from its computing device 103 to client device(s) 106. Overton, 4:28-39 ("The wrapper 118 is able to encode the video signal and/or audio signal by way of one or more media encoders 120 into a media stream. To this end, the wrapper 118 may include various types of media encoders 120, such as, for example, Moving Pictures Experts Group (MPEG) encoders, H.264 encoders, Flash® video encoders, etc."). Overton explains that "[s]uch media encoders 120 may be selected according to factors such as, for example, data reduction, encoding quality, latency, etc." Overton, 4:34-36. Overton thus recognizes quality concerns associated with transmitting a media stream and suggests varying encoders to deal with quality issues. Overton further teaches that its computing device 103 may include, e.g., the H.264 encoder (Overton, 4:31-34), and that computing device 103 and client 106 may communicate using RTP (Overton, 3:6-11).

200. A POSITA would have understood that, in addition to selecting a suitable encoder, adjusting the available resources to improve user perceived QoS (as taught by Lubonski) would improve the user experience by prioritizing network resources to elements of the media stream deemed most important for the user. A POSITA would have been motivated to do so when implementing Overton's teachings because, e.g., a poor user experience when trying an application might make the user less likely to purchase the application. *See also*, Section VII.C.3 (discussing importance of QoS to user experience). A POSITA implementing Overton's teachings would have thus found it worthwhile to improve QoS from the user's point of view, and in doing so, one would have looked to the teachings of Lubonski.

201. Specifically, a POSITA would have been motivated to implement Lubonski's server extension that dynamically adapts resource parameters to maximize user-perceived QoS for multimedia sent from Overton's computing device to client devices. A POSITA would have recognized that Lubonski's extension, which improves user-perceived QoS for video and audio streaming between a server and client connected by a network, would work at Overton's computing device 103, which is connected to client(s) 106 by a network because both Lubonski and Overton stream multimedia from server to client(s). A POSITA would have also understood that users perceive some changes in quality, such as

number of displayed frames per second or color depth, to be more significant than others. Lubonski, 155. These qualities are attributed to video, and so a POSITA would have been motivated to use Lubonski's extension to improve a user's experience for any given set of network resources by prioritizing resources for transmitting the video portion of the multi-media stream sent to clients 106.

202. And while Overton discloses encoding multimedia via H.264 and transmitting data via RTP, it does not provide additional detail regarding either standard. A POSITA would have therefore looked to Richardson for additional detail to implement the encoding and transport. In doing so, a POSITA would have seen that when H.264 data is transmitted with RTP, each data packet includes a type identifier. Richardson, 246. A POSITA would have understood that this type identifier could be used by Lubonski's Packet Classifier to distinguish between, e.g., video and audio flows because the type identifier indicates the type of data in the packet's payload. Lubonski, 155, Figure 2; Richardson, 246.

203. A POSITA would have thus been motivated to use RTP, as suggested by Overton and taught by Richardson, as the protocol for multimedia transport between Overton's computing device 103, with Lubonski's component for improving user user-perceived QoS, and Overton's client device(s). Doing so would have been no more than use of a known technique (RTP transport of H.264 data and Lubonski's server extension component) to improve similar methods (multimedia

transport between Overton's computing device and client device(s)) using a known approach that was ready for improvement (Overton's brief references to H.264 encoding and RTP transport) to yield predictable results (improved user-perceived QoS).

2. Claim 6: The method of claim 1, wherein the stream of user interface information comprises a plurality of different types of data and the server sends each different type of data to the mobile device in a separate data stream.

204. The combination of Overton, Lubonski, and Richardson renders obvious this claim.

205. Overton discloses sending a plurality of different types of data (e.g., video and audio) in a media stream from the computing device 103 (server) to client device(s) 106.

Meanwhile, the graphical output of the application 119 is captured by the wrapper 118 and encoded into a media stream. Additionally, the audio output of the application 119 may be captured and multiplexed into the media stream. The graphical output and/or audio output of the application 119 may be captured by hardware devices of the computing device 103 in some embodiments. The media stream is transmitted by the emulation server application 115 to the client 106 over the network 109 as the application output data 123.

Overton, 10:38-46.

206. A POSITA looking to improve Overton's quality of service would have looked to Lubonski and Richardson's teachings for the reasons explained in Section VII.F.1. Lubonski's server extension comprises a Packet Classifier that classifies and marks each application data flow. Lubonski, 155, Figure 2. The extension then adjusts resource allocation among the data flows discerned by the Packet Classifier based on user-perceived QoS. Lubonski, 155. Where RTP is used for multimedia transport (as suggested by Overton and described by Richardson), Overton's video and audio data (*the stream of user interface information comprises a plurality of different types of data*) would be transported via packets indicating different payload types. Richardson, 246, Figure 8.31. Specifically, Richardson explains that the RTP payload format "signal[s] the type of CODEC used to generate the data" and that "RTP payload formats are defined for various standard video and audio CODECs, including H.264." Richardson, 246. Because H.264 does not encode audio, a different CODEC would be used for the audio portion of the media stream. Accordingly, Lubonski's Packet Classifier would classify video and audio packets differently and separate them into different data flows so that user-perceived QoS of each data flow could be adjusted separately. A POSITA would have understood each of these individually adjusted data flows for video and audio to be *a separate data stream*.

- 3. Claim 7: The method of claim 6, wherein at least one of said separate data streams comprises a quality of service different from a quality of service of another of said separate data streams.**

207. The combination of Overton, Lubonski, and Richardson renders obvious this claim. Lubonski teaches adjusting resource allocation among data flows based on user-perceived QoS. Lubonski, 155. Lubonski further teaches that, e.g., the QoS of video is particularly important to users, even more so than the QoS of audio. *Id.* So a POSITA would have found it obvious to provide a *different QoS* for Overton's video data flow (*at least one of said separate data streams*) as compared to Overton's audio data flow (*another of said separate data streams*).

- 4. Claim 8: The method of claim 6, wherein at least one of said separate data streams comprises an encoding method and a bandwidth different from an encoding method and a bandwidth of another of said separate data streams.**

208. The combination of Overton, Lubonski, and Richardson renders obvious this claim.

209. As I explain in Sections VII.F.1 and VII.F.2, Overton discloses encoding and transmitting video and audio from the computing device 103 to client device(s). While Overton provides an example of encoding H.264 video, a POSITA would have known that H.264 does not provide an encoding method for audio. Richardson, 246 (defined RTP payload formats for various standard video and audio CODECS). Thus, a POSITA would have understood that Overton's video and audio

sent from the computing device (*separate data streams*) would comprise different CODECs (and thus *different ... encoding methods*).

210. Further, Lubonski teaches that adjusting parameters of a connection to improve user-perceived QoS comprises “allocating more bandwidth to [a] particular data flow if possible.” Lubonski, 155. And as explained in Section VII.F.3, a POSITA would have found it obvious to allocate resources differently (and thus bandwidth) for the video data flow and the audio data flow.

5. Claim 11

a. 11[pre]: A method for emulating a mobile device at a server, the method comprising:

211. Overton discloses or suggests the preamble for the same reasons discussed in Section VII.D.2.a.

b. 11[a]: receiving, by the server, input information from at least one mobile device;

212. Overton discloses or suggests this limitation for the reasons I discuss in Section VII.D.2.b.

- c. **11[b]: processing, by a mobile device operating system (OS) executed at the server, the input information to produce a stream of user interface information for use in reproducing at the mobile device at least one of a display, audio and a hardware (HW) action by a HW component of the mobile device based on a same instance of the mobile device OS; and**

213. Overton discloses or suggests this limitation for the reasons I discuss in Section VII.D.2.c.

- d. **11[c]: sending, by the server, the stream of user interface information to the mobile device, wherein the stream of user interface information from the same instance of the mobile device OS comprises a plurality of different types of data and the server sends each different type of data to the mobile device in a separate data stream for reproduction on the mobile device of at least one of the display, audio and the HW action by the HW component of the mobile device.**

214. The combination of Overton, Lubonski, and Richardson renders obvious this limitation. As I discuss in Section VII.D.2.d, Overton discloses *sending, by the server, the stream of user interface information to the mobile device...to the mobile device...for reproduction....* As I discuss in Section VII.F.2, the Overton-Lubonski-Richardson combination teaches that the *user interface information...comprises a plurality of different types of data and the server sends each different type of data to the mobile device in a separate data stream.*

6. **Claim 12: The method of claim 11, further comprising: emulating, by the server, mobile device HW and executing the mobile device OS on the emulated mobile device HW.**

215. The combination of Overton, Lubonski, and Richardson renders obvious this claim for the reasons I discuss in Section VII.D.3.

7. **Claim 13**

- a. **13[a]: The method of claim 11, further comprising: executing, by the server, at least one application on the mobile device OS;**

216. The combination of Overton, Lubonski, and Richardson renders obvious this limitation for the reasons I discuss in Section VII.D.4.a.

- b. **13[b]: wherein the at least one application processes the input information to produce the stream of user interface information.**

217. The combination of Overton, Lubonski, and Richardson renders obvious this limitation for the reasons I discuss in Section VII.D.4.b.

8. **Claim 15: The method of claim 11, further comprising: receiving the input information from a plurality of different mobile devices, wherein a same instance of the mobile device OS processes the input information from the plurality of different mobile devices to produce the stream of user interface information.**

218. The combination of Overton, Lubonski, and Richardson renders obvious this claim for the reasons I discuss in Section VII.D.5.

9. **Claim 16: The method of claim 11, wherein at least one of said separate data streams comprises a quality of service different from a quality of service of another of said separate data streams.**

219. The combination of Overton, Lubonski, and Richardson renders obvious this claim for the reasons I discuss in Section VII.F.3.

10. **Claim 17: The method of claim 11, wherein at least one of said separate data streams comprises an encoding method and a bandwidth different from an encoding method and a bandwidth of another of said separate data streams.**

220. The combination of Overton, Lubonski, and Richardson renders obvious this claim for the reasons I discuss in Section VII.F.4.

11. **Claim 23: The method of claim 20, wherein the stream of user interface information comprises a plurality of different types of data and the server sends each different type of data to the mobile device in a separate data stream.**

221. The combination of Overton, Lubonski, and Richardson renders obvious this claim for the reasons I discuss in Section VII.F.2.

12. **Claim 24: The method of claim 23, wherein at least one of said separate data streams comprises a quality of service different a quality of service of another of said separate data streams.**

222. The combination of Overton, Lubonski, and Richardson renders obvious this claim for the reasons I discuss in Section VII.F.3.

- 13. Claim 25: The method of claim 23, wherein at least one of said separate data streams comprises an encoding method and a bandwidth different from an encoding method and a bandwidth of another of said separate data streams.**

223. The combination of Overton, Lubonski, and Richardson renders obvious this claim for the reasons I discuss in Section VII.F.4.

G. Ground 3A: Lu

1. Lu

224. Lu discloses methods and apparatuses to allow users to remotely play games emulated on a server by using their client devices to send input to the server and receive a media stream from the server in response. Specifically, Lu discloses “a game appliance system can start a game appliance for a client system and said client system can send input to the game appliance system via a network to control game play of the game appliance and in return receive media stream of the game appliance via a network.” Lu, Abstract, Claim 1. Lu explains that a game appliance “comprises a package of at least one game software application and/or game software appliance.” Lu, ¶ 19. The “game application contained in a game appliance includes but [sic] not limited to, computer game, mobile game, console game, online game client, virtual world client, video game, etc.” Lu, ¶ 22. The game appliance “can also comprise emulation environment required by a game application (e.g., console emulation environment, handset emulation environment, portal gaming emulation environment, etc).” Lu, ¶ 21. Lu’s game appliance system comprises a

game appliance apparatus that can execute a game appliance by “emulate[ing] a target platform or a target operating system along with the game appliance running over the target platform or the target operating system.” Lu, ¶ 45, *see also, id.*, ¶¶ 43 (“a game appliance can use virtualization (e.g., platform emulation, machine emulator such as console emulator...”), 44 (“For example, using OS emulation or simulation, Windows game can be executed on Linux system.”), 99 (“A game appliance execution unit (4110) can execute a game appliance using emulation or virtualization (e.g., platform emulation, machine emulator such as console emulator, virtual machine, hypervisor, operating system emulation, hardware assisted virtualization, etc)”), 113 (“The kind of emulation or virtualization includes but not limited to [sic], full machine emulation, OS stack emulation....”), 120 (“game appliance (4112) can be executed by a game appliance execution unit (4110) of a game appliance apparatus (4000) using emulation...”).

225. Lu’s game appliance apparatus provides “interactive game play” for a client system by using “input data sent from a client system.” Lu, ¶¶ 67-70. Specifically, a “client system can send input to the game appliance apparatus via network to control game play of the game appliance and in return receive media stream of the game appliance via a local network.” Lu, ¶ 87, *see also, id.*, ¶ 99 (“A game appliance execution unit (4110) [a component of game appliance apparatus 4100] can execute a game appliance using emulation...receive input from one or

multiple client systems, capture graphics rendering command of the executed game appliance using virtual graphics driver, and transmit the captured graphics rendering command to one or multiple graphics rendering agents”). Lu’s Figure 1 depicts its game appliance system [red], comprising the game appliance apparatus [orange], and a client system [blue].

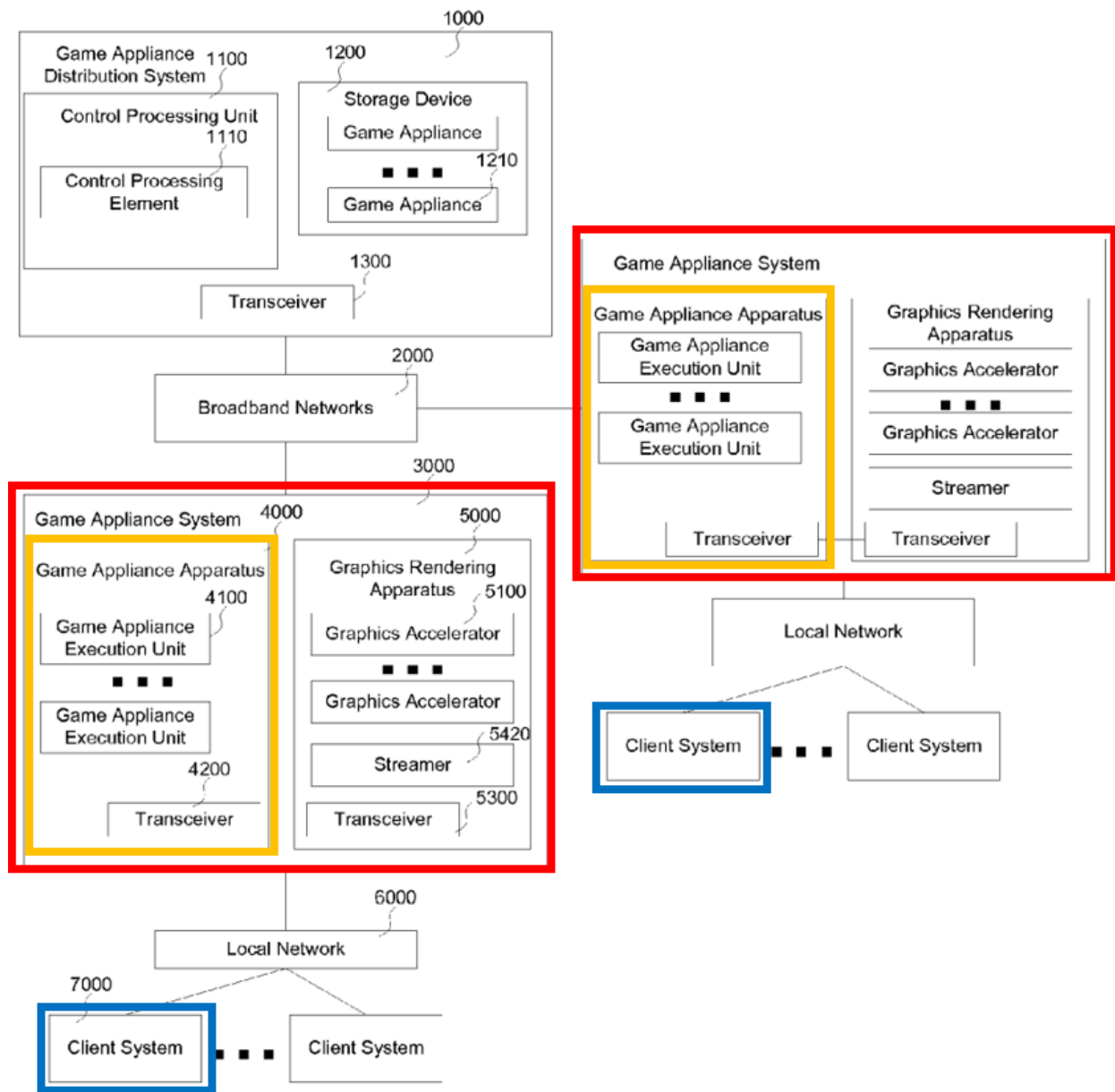


Fig. 1

Lu, Fig. 1 (annotated)

2. Claim 1

a. 1[pre]: A method for emulating a mobile device at a server, the method comprising:

226. Lu discloses that its game appliance system (“*a server*”) comprises a game appliance apparatus that can use emulation to execute a game appliance “originally developed for a different operating system.” Lu, ¶¶ 43, 46 (game appliance apparatus can emulate a target platform), 99, 113, 120. Lu further discloses that “the kind of game application contained in a game appliance includes...[a] mobile game.” Lu, ¶ 22. Because Lu discloses emulating a target platform for a game appliance, and that a game appliance can include a mobile game, a POSITA would have understood that Lu discloses emulating a device used to play a mobile game, i.e., a “*mobile device*.” Indeed, Lu provides several examples mobile devices that could be emulated by game appliance apparatus, including smart phones and Gameboys. Lu, ¶ 46.

b. 1[a]: receiving, by the server, input information from at least one mobile device, wherein the input information has not been processed to produce a display, audio and a hardware (HW) action by a HW component of the mobile device;

227. Lu discloses that a client device (“*mobile device*”) sends an input to the game appliance apparatus (which is within the game appliance system (“*the server*”), and that the game appliance apparatus uses the input data sent from the client system to control a game appliance it executed. Lu, ¶¶ 69-70, 99 (“A game appliance

execution unit (4110) [a component of game appliance apparatus 4100] can execute a game appliance using emulation...receive input from one or multiple client systems, capture graphics rendering command of the executed game appliance using virtual graphics driver, and transmit the captured graphics rendering command to one or multiple graphics rendering agents”). Because the client sends inputs to the claim appliance apparatus, which processes them, the game appliance apparatus (and thus, “*the server*”) “*receiv[es]*” the input from the client device. Lu further discloses that “a client system can transmit the original input event. Then, the game appliance apparatus that receives the event does any required translation.” Lu, ¶77. Lu thus discloses that “*the input information has not been processed to produce a display, audio and a hardware (HW) action by a HW component of the mobile device.*”

228. I note that the only processing Lu contemplates may be performed by the client system is translation of the input for processing by the game appliance apparatus. Lu, 76 (“In an additional embodiment, a client system may first translate an input event to a different **input event that can be processed by a destination game appliance apparatus....**”) (emphasis added). Lu thus explicitly states that the game appliance apparatus “processes” the input. To the extent that “translation” is considered “processing” as recited in this limitation, Lu also discloses an embodiment where the “client system can transmit the original input even. Then, the

game appliance apparatus that receives the event does any required translation.” Lu, ¶ 77.

- c. **1[b]: processing, by a mobile device operating system (OS) executed at the server, the input information to produce a stream of user interface information for use in reproducing at the mobile device at least one of the display, the audio and the hardware (HW) action by the HW component of the mobile device based on a same instance of the mobile device operating system; and**

229. In addition to emulating a mobile device, as discussed in Section VII.G.2.a, Lu discloses its game appliance apparatus emulating a “*mobile device operating system (OS)*.” Lu, ¶¶ 20 (game appliance comprises operating system required for running game applications), 43 (game appliance apparatus can use operating system emulation to execute a game appliance), 45 (game appliance apparatus can emulate a target operating system). Because Lu discloses the game appliance apparatus executing mobile games, a POSITA would have understood that the game appliance apparatus therefore also executes “*a mobile device operating system (OS)*.”

230. Lu further discloses that input sent from the client system to the game appliance apparatus is used “to control game play of the game appliance and in return receive [sic] media stream of the game appliance.” Lu, ¶ 87, Abstract, claim 1. Lu thus discloses that its game appliance (and thus the mobile device OS executed at

the game appliance apparatus) processes the input to create “*a stream of user interface information.*” Lu’s “client system...supports rendering of the media streams transmitted to it from a game appliance apparatus or a graphics rendering apparatus over a local network. It can decompress the received media stream and display the frames on an attached display device.” Lu, ¶ 61. Lu thus discloses that the media stream is used in “*reproducing at the mobile device at least one of the display, the audio and the hardware (HW) action by the HW component of the mobile device.*”

231. I note that Lu teaches that “[t]he actual graphics rendering can be carried out by a game appliance apparatus or a graphics rendering apparatus.” Lu, ¶ 105. Thus, while Lu discusses at length the operation of a graphics rendering apparatus separate from the game appliance apparatus, a POSITA would have understood that Lu also discloses embodiments where graphics rendering is performed by the same game appliance apparatus that receives input from the client devices, i.e., “*based on a same instance of the mobile device operating system.*”

232. To the extent that PO argues, as the Applicant argued during prosecution to overcome a rejection, that this limitation requires that multiple data streams be provided by a single emulation of the mobile device, Lu renders obvious this limitation.’887FH, 198 (“Put differently, applicants’ claimed invention provides multiple data streams (e.g., audio, video, compressed text, voice and/or sensors)

arriving from a single mobile device emulation.”). While Lu focuses exclusively on the graphics rendering aspect of executing a game application at the game appliance apparatus, a POSITA would have understood that, or at the very least found it obvious to modify Lu’s invention to, also execute audio associated with game play and to include said audio in the media stream sent from the game appliance apparatus to the client system. Specifically, a POSITA would have known that most games include audio, and would have understood that the game appliance apparatus would generate said audio when executing the game. Lu does not disclose anything that would suggest deviation from this expectation. For example, it does not disclose rendering audio separately, nor does it disclose how the game applications could have been modified to separately render audio. Accordingly, a POSITA would have understood that in executing game applications, both video and audio would be produced at Lu’s game appliance apparatus.

233. Even if it would not have been obvious that Lu’s game appliance apparatus provides both video and audio in the media stream it sends back to the client system, a POSITA would have been motivated to modify Lu to perform audio processing at the game appliance apparatus and to include audio in the media stream sent back to the client devices. Doing so would have simplified the system by requiring that only one instance of the game application needs to be run, at the game appliance apparatus, to provide both video and audio associated with gameplay.

This is consistent with Lu's disclosure, which does not provide any teaching for executing the game application at the client system (or any other way of obtaining audio associated with a game application) in a way that syncs the audio and video from the game. For example, in a fighting game a visual, such as a player striking another player, would typically be accompanied audio, such as a smacking sound. If video and audio are both provided, e.g., at the game appliance apparatus, the video and audio would be processed together as normal. Separating the video and audio processing by the game would not be straightforward, not the least because it would require somehow syncing user input and the corresponding video and audio output from the game. Lu does not provide any teaching or suggestion about how, or indeed why, a POSITA would undertake doing such a thing.

234. Finally, as mentioned above, Lu discloses that the game appliance apparatus alone can execute a game application. *See also*, Lu, ¶ 99 (single game appliance execution unit with game appliance apparatus can execute a game appliance using emulation). Thus, to the extent that this limitation requires multiple data streams (e.g., video and audio) to be provided from a single emulation, Lu discloses or at the very least renders obvious this limitation.

- d. 1[c]: sending, by the server, the stream of user interface information to the mobile device to reproduce at the mobile device at least one of the display, the audio and the HW action by the HW component of the mobile device.**

235. Lu discloses sending the game appliance apparatus (part of the game appliance system (“*the server*”)) sending a media stream back to the client system. Lu, Abstract, ¶¶ 49, 61, 64, 81, 87. The client system, which can be a “mobile device such as PDA or cellphone or handset, netbook, laptop computer, etc) supports rendering of the media streams transmitted to it from a game appliance apparatus.” Lu, ¶ 61. The client system “decompress[es] the received media stream and display[s] the frames on an attached display device.” Lu, ¶¶ 61, 64, 66.

- 3. Claim 2: The method of claim 1, further comprising: emulating, by the server, mobile device HW and executing the mobile device OS on the emulated mobile device HW.**

236. Lu discloses that the game appliance apparatus can emulate an entire target platform in order to execute a game application.

Furthermore, when emulation is used, the host operating system running on a game appliance apparatus can create emulation tasks that emulate a target platform or a target operating system along with the game appliance running over the target platform or the target operating system.

Lu, ¶ 45, ¶¶ 43-44, 46. A POSITA would have understood emulating a target platform to comprise emulating the target platform hardware and executing the

associated OS on the emulated hardware because Lu explains that its game appliance apparatus emulates different hardware and software because “[a] game appliance may be developed for hardware or operating system different from the hardware or operating system of the game appliance apparatus”), *see also*, Lu, ¶ 70 (“game appliance apparatus can use device emulation to emulate one or multiple input devices....A game appliance executed by a game appliance apparatus can receive input data from the emulated input device. A game appliance apparatus can use the input data sent from a client system to drive the emulated input device.”).

4. Claim 3

- a. 3[a]: The method of claim 1, further comprising:
executing, by the server, at least one application on
the mobile OS;**

237. Lu discloses or suggests this limitation. Lu discloses the game appliance apparatus (part of the game appliance system (“*the server*”)) executing a game application. Lu, ¶¶ 30, 43-46.

- b. 3[b]: wherein the at least one application processes
the input information to produce the stream of user
interface information.**

238. Lu discloses or suggests this limitation. Lu discloses that a game appliance (which comprises a game application) receives input data from the client system. Lu, ¶¶ 69-72, 99. The executed game appliance then provides a graphics

rendering command³⁰, Lu, ¶ 99, used to create the media stream sent back to the client system, Lu, Abstract, ¶¶ 49, 61, 64, 81, 87.

**5. Claim 5: The method of claim 1, further comprising:
receiving, by the server, the input information from a
plurality of different mobile devices, wherein a same
instance of the mobile device OS processes the input
information from each of the plurality of different mobile
devices to produce the stream of user interface information.**

239. Lu discloses or suggests this claim. Lu discloses that “one...game appliance apparatus[] can retrieve game appliance and execute game appliance for one or a collection of client systems.” Lu, ¶26. Thus, a POSITA would have understood that Lu discloses an embodiment where a single game appliance apparatus, and thus a single “*mobile device OS*” processes the input from multiple different client systems “*mobile devices*” to produce the stream of user interface information.

240. Such an embodiment would also have been obvious to a POSITA. Lu discloses emulating systems such as a Gamecube, a Playstation, and an Xbox. Lu, ¶ 46. These systems were well known for providing multiplayer games, including games with local multiplayer such as Super Smash Bros Melee and Mario Party, and a POSITA would have understood that these emulated systems thus support, without

³⁰ As I noted earlier in Section VII.G.2.c, Lu discloses the game appliance apparatus rendering the graphics.

any modification, receiving input from multiple controllers. And because Lu discloses that one game appliance apparatus can execute a game appliance for a collection of client systems, a POSITA would have found it obvious that Lu's game appliance apparatus can process input from multiple client systems, acting as controllers for an emulated game with local multiplayer, and produce a stream of user interface back to each of said client systems.

6. Claim 20

a. 20[pre]: A method for emulating a mobile device at a server, the method comprising:

241. Lu discloses or suggests the preamble for the same reasons discussed in Section VII.G.2.a.

b. 20[a]: receiving, by the server, input information from a plurality of different mobile devices, wherein the input information has not been processed to produce a display, audio and a hardware (HW) action by a corresponding HW component of the mobile devices;

242. Lu discloses or suggests this limitation for the same reasons discussed in Sections VII.G.2.b (receiving input information that has not been processed) and VII.G.5 (a single game appliance apparatus executes game appliances for, and thus receives input from, multiple client systems).

- c. **20[b]: processing, by a same instance of a mobile device operating system (OS) executed at the server, the input information from each of the plurality of different mobile devices to produce user interface information for use in reproducing at the different mobile devices at least one of the display, audio and the HW action by the HW components of the different mobile device based on the same instance of the mobile device OS; and**

243. Lu discloses or suggests this limitation for the reasons I discuss in Sections VII.G.2.c (“*processing...*” for a single client system) and VII.G.5 (multiple client systems to a single game appliance apparatus).

- d. **20[c]: sending, by the server, the user interface information to at least one of the plurality of different mobile devices.**

244. Lu discloses or suggests this limitation for the same reasons discussed in Sections VII.G.2.d (“*sending...*”) and VII.G.5 (multiple client systems to a single game appliance apparatus).

7. Claim 21

- a. **The method of claim 20, further comprising: executing, by the server, at least one application on the mobile device OS;**

245. Lu discloses or suggests this limitation for the same reasons discussed in Section VII.G.4.a.

- b. 21[b]: wherein the at least one application processes the input information to produce the user interface information.**

246. Lu discloses or suggests this limitation for the same reasons discussed in Section VII.G.4.b.

H. Ground 3B: Lu + Lubonski + Richardson

1. Motivation to Combine

247. A POSITA would have been motivated to improve user-perceived QoS for media transmitted from Lu's game appliance apparatus (within the game appliance system ("*the server*") to client systems by implementing the teachings of Lubonski's server extension and Richardson's use of RTP for transmitting H.264 data.

248. Lu recognizes the need for, and discloses some methods for ensuring, quality of service for the media stream sent back to the client system. For example, Lu discloses based on a QoS requirement:

In another embodiment, a graphics rendering apparatus (5000) can select the next graphics rendering context out of a collection of graphics rendering contexts based on pre-determined frame rate constraints or frame rendering deadlines. For example, in an embodiment, a graphics rendering context can have a quality of service requirement that specifies a desired rendering frame rate or frame rendering period.

Lu, ¶ 160, *see also*, Lu, ¶¶ 182, 183 ("Furthermore, transcoding can also be performed according to achievable or pre-determined bitrate. In an embodiment,

bitrate can be determined based on the available or assigned bandwidth between a client system and a game appliance system.”).

249. A POSITA would have understood that, in addition to varying graphics rendering to a desired frame rate or bitrate, as mentioned by Lu, adjusting the utilization of available resources to improve user-perceived QoS (as taught by Lubonski) would improve the user experience by prioritizing network resources to elements of the media stream deemed most important for the user. Lubonski suggests improving user-perceived QoS with a server-side component that adjusts resource allocation among packet flows from a server to a client to preserve user-perceived QoS. Lubonski, 155. For example, Lubonski notes that a decrease in displayed frames per second, a parameter that Lu suggests adjusting to meet QoS requirements, can create a “significant deterioration in user-perceived performance.” Lubonski, 155, Lu, ¶160. A POSITA would have thus recognized that Lu’s approach to maintaining QoS may have shortcomings to the extent that it could disproportionately affect user-perceived QoS. Recognizing the importance of the user experience, see also, Section VII.C.3 (discussing importance of QoS to user experience), particularly in gaming applications, to which Lu is directed, a POSITA would have been motivated to implement Lubonski’s approach to improving user-perceived QoS in order to improve the user experience. Specifically, a POSITA would have been motivated to implement Lubonski’s server extension that

dynamically adapts resource parameters to maximize user-perceived QoS for multimedia sent from Lu's game appliance system to client system(s).

250. Because Lubonski teaches use of a packet classifier to mark data flows sent from the server, Lubonski, 155, Fig. 2, a POSITA would have needed to understand the structure of packets sent from Lu's game appliance system to a client system. Lu discloses using RTP (Real-Time Protocol) to stream media to a client system. Lu, ¶ 187. A POSITA would have known that RTP packets include a type identifier that signals the type of CODEC used to generate the data in the packet's payload. Richardson, 246, Fig. 8.31. A POSITA would have therefore had a reasonable expectation of success in combining Lubonski's teachings with Lu's because Lubonski's packet classifier could use the type identifier in the RTP packets already used by Lu to transmit the media stream to the client system.

2. Claim 6: The method of claim 1, wherein the stream of user interface information comprises a plurality of different types of data and the server sends each different type of data to the mobile device in a separate data stream.

251. The combination of Lu and Lubonski renders obvious this claim.

252. As explained in Section VII.G.2.c, it would have been obvious to include both video and audio data in the media stream sent from Lu's game appliance system to a client system.

253. A POSITA looking to improve the Lu's quality of service would have looked to Lubonski's teachings for the reasons explained in Section VII.H.1. Lubonski's server extension comprises a Packet Classifier that classifies and marks each application data flow. Lubonski, 155, Figure 2. The extension then adjusts resource allocation among the data flows discerned by the Packet Classifier based on user-perceived QoS. Lubonski, 155. Because Lu discloses sending the media stream to the client system via RTP, which includes a type identifier with each packet, Lubonski's Packet Classifier would classify video and audio packets differently and separate them into different data flows so that user-perceived QoS of each data flow could be adjusted separately. A POSITA would have understood each of these individually adjusted data flows for video and audio to be *a separate data stream*.

3. Claim 7: The method of claim 6, wherein at least one of said separate data streams comprises a quality of service different from a quality of service of another of said separate data streams.

254. The combination of Lu and Lubonski renders obvious this claim. Lubonski teaches adjusting resource allocation among data flows based on user-perceived QoS. Lubonski, 155. Lubonski further teaches that, e.g., the QoS of video is particularly important to users, even more so than the QoS of audio. *Id.* So a POSITA would have found it obvious to provide a *different* QoS for Overton's video

data flow (*at least one of said separate data streams*) as compared to Overton's audio data flow (*another of said separate data streams*).

4. Claim 8: The method of claim 6, wherein at least one of said separate data streams comprises an encoding method and a bandwidth different from an encoding method and a bandwidth of another of said separate data streams.

255. The combination of Lu and Lubonski renders obvious this claim.

256. Lu discloses that video may be encoded using any video or image compression approach, e.g., H.264. Lu, ¶¶ 56, 182. Lu does not disclose a specific encoding method for audio, but a POSITA would have understood that audio would need to be encoded for efficient transmission and that H.264 does not encode audio. A POSITA would have further understood that none of the video or image encoding approaches disclosed by Lu can be used to encode audio, and thus would have known to use a different encoding method, e.g., AAC (Advanced Audio Codec). *See* Richardson, 246 (discussing separate video and audio codecs).

257. Further, Lubonski teaches that adjusting parameters of a connection to improve user-perceived QoS comprises “allocating more bandwidth to particular data flow if possible.” Lubonski, 155. And as explained in Section VII.H.3, a POSITA would have found it obvious to allocate resources differently (and thus bandwidth) for the video data flow and the audio data flow.

5. Claim 11

- a. 11[pre]: A method for emulating a mobile device at a server, the method comprising:**

258. Lu discloses or suggests the preamble for the same reasons discussed in Section VII.G.2.a.

- b. 11[a]: receiving, by the server, input information from at least one mobile device;**

259. Lu discloses or suggests this limitation for the same reasons discussed in Section VII.G.2.b.

- c. 11[b]: processing, by a mobile device operating system (OS) executed at the server, the input information to produce a stream of user interface information for use in reproducing at the mobile device at least one of a display, audio and a hardware (HW) action by a HW component of the mobile device based on a same instance of the mobile device OS; and**

260. Lu discloses or suggests this limitation for the same reasons discussed in Section VII.G.2.c.

- d. **11[c]: sending, by the server, the stream of user interface information to the mobile device, wherein the stream of user interface information from the same instance of the mobile device OS comprises a plurality of different types of data and the server sends each different type of data to the mobile device in a separate data stream for reproduction on the mobile device of at least one of the display, audio and the HW action by the HW component of the mobile device.**

261. The combination of Lu and Lubonski renders obvious this limitation for the same reasons discussed in Sections VII.G.2.d (*sending, by the server, the stream of user interface information to the mobile device...to the mobile device...for reproduction...*) and VII.H.2 (*user interface information...comprises a plurality of different types of data and the server sends each different type of data to the mobile device in a separate data stream*).

- 6. **Claim 12: The method of claim 11, further comprising: emulating, by the server, mobile device HW and executing the mobile device OS on the emulated mobile device HW.**

262. The combination of Lu and Lubonski renders obvious this claim for the same reasons discussed in Section VII.G.3.

7. **Claim 13**

- a. **13[a]: The method of claim 11, further comprising: executing, by the server, at least one application on the mobile device OS;**

263. The combination of Lu and Lubonski renders obvious this limitation for the same reasons discussed in Section VII.G.4.a.

- b. 13[b]: wherein the at least one application processes the input information to produce the stream of user interface information.**

264. The combination of Lu and Lubonski renders obvious this limitation for the same reasons discussed in Section VII.G.4.b.

- 8. Claim 15: The method of claim 11, further comprising: receiving the input information from a plurality of different mobile devices, wherein a same instance of the mobile device OS processes the input information from the plurality of different mobile devices to produce the stream of user interface information.**

265. The combination of Lu and Lubonski renders obvious this claim for the same reasons discussed in Section VII.G.5.

- 9. Claim 16: The method of claim 11, wherein at least one of said separate data streams comprises a quality of service different from a quality of service of another of said separate data streams.**

266. The combination of Lu and Lubonski renders obvious this claim for the same reasons discussed in Section VII.H.3.

- 10. Claim 17: The method of claim 11, wherein at least one of said separate data streams comprises an encoding method and a bandwidth different from an encoding method and a bandwidth of another of said separate data streams.**

267. The combination of Lu and Lubonski renders obvious this claim for the same reasons discussed in Section VII.H.4.

- 11. Claim 23: The method of claim 20, wherein the stream of user interface information comprises a plurality of different types of data and the server sends each different type of data to the mobile device in a separate data stream.**

268. The combination of Lu and Lubonski renders obvious this claim for the same reasons discussed in Section VII.H.2.

- 12. Claim 24: The method of claim 23, wherein at least one of said separate data streams comprises a quality of service different a quality of service of another of said separate data streams.**

269. The combination of Lu and Lubonski renders obvious this claim for the same reasons discussed in Section VII.H.3.

- 13. Claim 25: The method of claim 23, wherein at least one of said separate data streams comprises an encoding method and a bandwidth different from an encoding method and a bandwidth of another of said separate data streams.**

270. The combination of Lu and Lubonski renders obvious this claim for the same reasons discussed in Section VII.H.4.