

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

**SAP America, Inc.,
Petitioner**

**IPR2025-00420
U.S. Patent No. 6,889,244**

**DECLARATION OF KEVIN ALMEROTH, PH.D. UNDER 37 C.F.R. § 1.68
IN SUPPORT OF PETITION FOR *INTER PARTES* REVIEW**

TABLE OF CONTENTS

	Page
EXHIBIT LIST	9
I. INTRODUCTION	11
II. QUALIFICATIONS AND PROFESSIONAL EXPERIENCE	12
III. LEVEL OF ORDINARY SKILL IN THE ART	25
IV. RELEVANT LEGAL STANDARDS	26
V. OVERVIEW OF THE '244 PATENT.....	28
VI. CLAIM CONSTRUCTION	31
VII. IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE.....	32
VIII. THE PRINCIPAL PRIOR ART	33
A. Summary of <u>Bowman</u>	33
B. Summary of <u>Vahalia</u>	36
C. Summary of <u>Tuxedo</u>	37
D. Motivation to Combine	40
1. <u>Bowman</u> / <u>Vahalia</u> combination	41
2. <u>Bowman</u> / <u>Vahalia</u> / <u>Tuxedo</u> combination	51
IX. CLAIMS 1-10 ARE UNPATENTABLE	55
A. Ground 1: Claim 1 is Obvious Based on <u>Bowman</u> in view of <u>Vahalia</u>	55
1. Claim 1	55
i. [1.0] A method of transmitting messages between a first node and a second node, wherein the first and second nodes are each coupled to a fault tolerant storage system (FTSS), the method comprising:.....	55
ii. [1.1] transmitting a message from the first node to a communication agent in the FTSS;.....	63
iii. [1.2] storing the message in a data structure in highly reliable fault-tolerant storage media of the FTSS;	66
iv. [1.3] processing the message at the FTSS in accordance with a messaging paradigm; and.....	71
v. [1.4] transmitting the message from the FTSS to the second node.	75

B. Ground 2: Claims 2–10 are Obvious Based on <u>Bowman</u> in view of <u>Vahalia</u> and in Further View of <u>Tuxedo</u>	79
1. Claim 2	79
i. [2.0] The method of claim 1 wherein the messaging paradigm is a request/reply communication paradigm, and:.....	79
ii. [2.1] transmitting a message from the first node to a communication agent in the FTSS comprises:	84
iii. [2.2] transmitting a request message from the first node to a request/reply communication agent in the FTSS;.....	84
iv. [2.3] storing the message in a data structure in highly reliable fault-tolerant storage media of the FTSS comprises:	85
v. [2.4] storing the request message in a request queue in highly reliable fault-tolerant storage media of the FTSS;.....	85
vi. [2.5] processing the message at the FTSS in accordance with a messaging paradigm comprises:	89
vii. [2.6] retrieving the request message from the request queue; and	89
viii. [2.7] transmitting the message from the FTSS to the second node comprises:	90
ix. [2.8] transmitting the request message from the FTSS to a service on the second node;.....	90
x. [2.9a] and the method further comprises: executing the service on the second node using the request message as an input and providing a reply message as an output;.....	94
xi. [2.9b] transmitting the reply message from the second node to the request/reply communication agent in the FTSS;.....	97
xii. [2.9c] storing the reply message in a reply queue in highly reliable fault-tolerant storage media of the FTSS;.....	97
xiii. [2.9d] retrieving the reply message from the reply queue; and.....	98
xiv. [2.9e] transmitting the reply message from the FTSS to the first node.	99
2. Claim 3	100
i. [3.0] The method of claim 1 wherein the messaging paradigm is a conversational communication paradigm, and:	100

ii.	[3.1] transmitting a message from the first node to a communication agent in the FTSS comprises:	102
iii.	[3.2] transmitting a connect message from the first node to a conversational communication agent in the FTSS;	102
iv.	[3.3a] processing the message at the FTSS in accordance with a messaging paradigm comprises: opening a conversation queue in highly reliable fault-tolerant storage media of the FTSS;.....	104
v.	[3.3b] storing the message in a data structure in highly reliable fault-tolerant storage media of the FTSS comprises:	107
vi.	[3.3c] storing the connect message in the conversation queue; and	108
vii.	[3.4] transmitting the message from the FTSS to the second node comprises: transmitting the connect message from the FTSS to the second node.	109
3.	Claim 4	109
i.	[4.0] The method of claim 3 and further comprising:.....	109
ii.	[4.1] transmitting a send message from the first node to the conversational communication agent in the FTSS;	109
iii.	[4.2] storing the send message in the conversation queue;	110
iv.	[4.3] transmitting the send message to from the FTSS to a service on the second node;.....	111
v.	[4.4] executing the service in the second node using the send message as an input and providing a receive message as an output;	111
vi.	[4.5] transmitting the receive message from the second node to the conversational communication agent in the FTSS;	113
vii.	[4.6] storing the receive message in the conversation queue; and...	113
viii.	[4.7] transmitting the receive message to the first node.....	114
4.	Claim 5	114
i.	[5.0] The method of claim 3 and further comprising:.....	114
ii.	[5.1] transmitting a disconnect message from the first node to the conversational communication agent in the FTSS;	114
iii.	[5.2] closing the conversation queue; and	116
iv.	[5.3] transmitting the disconnect message to the second node.....	117
5.	Claim 6	117

- i. [6.0] The method of claim 1 wherein the messaging paradigm is an event-based communication paradigm, and: 117
- ii. [6.1] transmitting a message from the first node to a communication agent in the FTSS comprises: transmitting a post message from the first node to an event-based communication agent in the FTSS;.....118
- iii. [6.2] processing the message at the FTSS in accordance with a messaging paradigm comprises: accessing a subscription manager to determine whether any nodes have subscribed to receive the event represented by the post message; 120
- iv. [6.3] storing the message in a data structure in highly reliable fault-tolerant storage media of the FTSS comprises: storing the post message in an event queue in highly reliable fault-tolerant storage media of the FTSS; and 121
- v. [6.4] transmitting the message from the FTSS to the second node comprises: transmitting the post message from the FTSS to the second node if the second node has subscribed to receive the event represented by the post message. 122
- 6. Claim 7 123
 - i. [7.0] The method of claim 6 and further comprising:.....123
 - ii. [7.1] transmitting a subscribe message from the second node to the event-based communication manager in the FTSS, wherein the subscribe message includes an event; and 123
 - iii. [7.2] accessing the subscription manager to record that the second node has subscribed to receive post messages associated with the event. 124
- 7. Claim 8 125
 - i. [8.0] The method of claim 6 and further comprising:.....125
 - ii. [8.1] transmitting an unsubscribe message from the second node to the event-based communication manager in the FTSS, wherein the unsubscribe message includes an event; and 126
 - iii. [8.2] accessing the subscription manager to record that the second node should no longer receive post messages associated with the event. .127
- 8. Claim 9 127
 - i. [9.0] The method of claim 1 wherein the messaging paradigm is a queue-based communication paradigm, and: 127

- ii. [9.1] transmitting a message from the first node to a communication agent in the FTSS comprises: transmitting an enqueue message from the first node to a queue-based communication agent in the FTSS; 129
- iii. [9.2] storing the message in a data structure in highly reliable fault-tolerant storage media of the FTSS comprises: storing the enqueue message in a queue space in highly reliable fault tolerant storage media of the FTSS; 132
- iv. [9.3] processing the message at the FTSS in accordance with a messaging paradigm comprises: retrieving the enqueue message from the queue space only if the second node is available; and.....133
- v. [9.4] transmitting the message from the FTSS to the second node comprises: transmitting the enqueue message from the FTSS to a service on the second node only if the second node is available; 135
- vi. [9.5a] and the method further comprises: executing the service on the second node using the enqueue message as an input and providing a dequeue message as an output; 136
- vii. [9.5b] transmitting the dequeue message from the second node to the queue-based communication agent in the FTSS;.....137
- viii. [9.5c] storing the dequeue message in the queue space; 137
- ix. [9.5d] retrieving the dequeue message from the queue space; and .138
- x. [9.5e] transmitting the dequeue message from the FTSS to the first node. 138
- 9. Claim 10 139
 - i. [10.0] The method of claim 1 wherein the messaging paradigm is an unsolicited communication paradigm, and: 139
 - ii. [10.1] transmitting a message from the first node to a communication agent in the FTSS comprises: transmitting a broadcast message from the first node to an unsolicited communication agent in the FTSS;.....140
 - iii. [10.2] processing the message at the FTSS in accordance with a messaging paradigm comprises: accessing determining which nodes should receive the broadcast message based on the broadcast message;.....141
 - iv. [10.3] storing the message in a data structure in highly reliable fault-tolerant storage media of the FTSS comprises: storing the broadcast message in an unsolicited message queue in highly reliable fault-tolerant storage media of the FTSS; and.....142

v. [10.4] transmitting the message from the FTSS to the second node comprises: transmitting the post message from the FTSS to all nodes that should receive the broadcast message.	143
X. CONCLUSION.....	144

TABLE OF AUTHORITIES

	Page(s)
Statutes	
18 U.S.C. § 1101	133
35 U.S.C. § 102(b)	27
35 U.S.C. § 102(e)	23, 26
35 U.S.C. § 103	17

EXHIBIT LIST

No.	Exhibit Description
1101	U.S. Patent No. 6,889,244
1102	U.S. Patent No. 6,889,244 File History
1103	Declaration of Kevin Almeroth, Ph.D.
1104	CV of Kevin Almeroth, Ph.D.
1105	U.S. Patent No. 6,615,253 to Bowman-Amuah (<u>Bowman</u>)
1106	U.S. Patent 6,973,455 to Vahalia et al. (<u>Vahalia</u>)
1107	Juan M. Andrade, et al., THE TUXEDO SYSTEM (1996) (<u>Tuxedo</u>)
1108	Jesper Gullberg, Mariusz Gieparda, KyeongWon Jeong, IBM eserver Certification Study Guide - pSeries HACMP for AIX (2001) (CSG)
1109	Johnny Shieh, Quick Reference: AIX Logical Volume Manager and Veritas Volume Manager
1110	Declaration of Martin Farach-Colton, M.D., PH.D. In Support of Plaintiffs' Proposed Claim Constructions (Dec. 20, 2024)
1111	HP SureStore eDisk Array XP512, Configuration and Ordering Guide (Oct. 1, 2000)
1112	Random House Webster's Computer & Internet Dictionary (3d Ed. 1999)
1113	Newton's Telecom Dictionary (15th ed. 1999)
1114	Microsoft Computer Dictionary (4th ed. 1999)
1115	[RESERVED]
1116	Director Vidal's Memorandum Regarding Discretionary Denial Procedure, Dated June 21, 2022
1117	First Amended Docket Control Order in <i>Valtrus Innovations Ltd. v. SAP America, Inc. et al</i> , 2:24-cv-00021-JRG (E.D. Tex.)
1118	Motion to Dismiss for Lack of Standing (Public Version) in <i>Valtrus Innovations Ltd. v. SAP America, Inc. et al</i> , 2:24-cv-00021-JRG (E.D. Tex.)

1119	Motion to Reconsider Ruling on Motion to Dismiss for Lack of Standing (Public Version) in <i>Valtrus Innovations Ltd. v. SAP America, Inc. et al</i> , 2:24-cv-00021-JRG (E.D. Tex.)
1120	Original Complaint for Patent Infringement in <i>Valtrus Innovations Ltd. v. SAP America, Inc. et al</i> , 2:24-cv-00533-JRG (E.D. Tex.)
1121	Plaintiffs' Notice Regarding <i>Valtrus II</i> in <i>Valtrus Innovations Ltd. v. SAP America, Inc. et al</i> , 2:24-cv-00021-JRG (E.D. Tex.)
1122	SAP's Invalidity Contentions (Cover Pleading) in <i>Valtrus Innovations Ltd. v. SAP America, Inc. et al</i> , 2:24-cv-00021-JRG (E.D. Tex.)
1123	Declaration of Sylvia Hall-Ellis, Ph.D.

I, Kevin Almeroth, Ph.D., do hereby declare as follows:

I. INTRODUCTION

1. I am making this declaration at the request of Petitioner in the matter of the Inter Partes Review of U.S. Patent No. 6,889,244 (“the ’244 Patent”) to Gaither et al.

2. I am being compensated for my work in this matter at my standard hourly rate. I am also being reimbursed for reasonable and customary expenses associated with my work and testimony in this proceeding. My compensation is not contingent on the outcome of this matter or the specifics of my testimony.

3. I have been asked to provide my opinions regarding whether the subject matter of claims 1–10 (“the Challenged Claims”) of the ’244 Patent are anticipated or otherwise would have been obvious to a person having ordinary skill in the art (“POSITA”) at the time of the alleged invention, in light of the prior art. It is my opinion that the Challenged Claims would have been obvious to a POSITA.

4. In the preparation of this declaration, I have studied:

- a. the ’244 Patent, Ex-1101.
- b. the prosecution history of the ’244 Patent (“’244 File History”), Ex-1102.
- c. U.S. Patent No. 6,615,253 to Bowman-Amuah (Bowman) (Ex-1105).

d. U.S. Patent 6,973,455 to Vahalia et al. (Vahalia) (Ex-1106).

e. Juan M. Andrade, et al., THE TUXEDO SYSTEM (1996) (Tuxedo) (Ex-1107).

5. The additional references I refer to below and/or are identified in the exhibit list.

6. In forming the opinions expressed below, I have considered: the documents listed above; the relevant legal standards, including the standard for obviousness; and my own knowledge and experience based upon my work in the field of network communications and security as described below.

7. Unless otherwise noted, all emphasis in any quoted material has been added. Claim terms are italicized.

II. QUALIFICATIONS AND PROFESSIONAL EXPERIENCE

8. My complete qualifications and professional experience are described in my Curriculum Vitae, a copy of which can be found in Ex-1104. The following is a brief summary of my relevant qualifications and professional experience.

9. I am currently a Professor Emeritus in the Department of Computer Science at the University of California, Santa Barbara (UCSB). While at UCSB, I held faculty appointments and was a founding member of the Computer Engineering (CE) Program, Media Arts and Technology (MAT) Program, and the Technology Management Program (TMP). I also served as the Associate Director of the Center

for Information Technology and Society (CITS) from 1999 to 2012. I have been a faculty member at UCSB since July 1997.

10. I hold three degrees from the Georgia Institute of Technology: (1) a Bachelor of Science degree in Information and Computer Science (with minors in Economics, Technical Communication, and American Literature) earned in June 1992; (2) a Master of Science degree in Computer Science (with specialization in Networking and Systems) earned in June 1994; and (3) a Doctor of Philosophy (Ph.D.) degree in Computer Science (Dissertation Title: Networking and System Support for the Efficient, Scalable Delivery of Services in Interactive Multimedia System, minor in Telecommunications Public Policy) earned in June 1997. During my education, I took a wide variety of courses as demonstrated by my minors. My undergraduate degree also included a number of courses more typical of a degree in electrical engineering including digital logic, signal processing, and telecommunications theory.

11. One of the major concentrations of my research over the past 30+ years has been the delivery of multimedia content and data between computing devices, including through various network architectures. In my research, I have studied large-scale content delivery systems, and the use of servers located in a variety of geographic locations to provide scalable delivery to hundreds or thousands of users simultaneously. I have also studied smaller-scale content delivery systems in which

content is exchanged between individual computers and portable devices. My work has emphasized the exchange of content more efficiently across computer networks, including the scalable delivery of content to many users, mobile computing, satellite networking, delivering content to mobile devices, and network support for data delivery in wireless networks.

12. In 1992, the initial focus of my research was on the provision of interactive functions (e.g., VCR-style functions like pause, rewind, and fast-forward) for near video-on-demand systems in cable systems; in particular, how to aggregate requests for movies at a cable head-end and then how to satisfy a multitude of requests using one audio/video stream broadcast to multiple receivers simultaneously. This research has continually evolved and resulted in the development of techniques to scalably deliver on-demand content, including audio, video, web documents, and other types of data, through the Internet and over other types of networks, including over cable systems, broadband telephone lines, and satellite links.

13. An important component of my research has been investigating the challenges of communicating multimedia content, including video, between computers and across networks including the Internet. Although the early Internet was used mostly for text-based, non-real time applications, the interest in sharing multimedia content, such as video, quickly developed. Multimedia-based

applications ranged from downloading content to a device to streaming multimedia content to be instantly used. One of the challenges was that multimedia content is typically larger than text-only content, but there are also opportunities to use different delivery techniques since multimedia content is more resilient to errors. I have worked on a variety of research problems and used a number of systems that were developed to deliver multimedia content to users. One content-delivery method I have researched is the one-to-many communication facility called “multicast,” first deployed as the Multicast Backbone, a virtual overlay network supporting one-to-many communication. Multicast is one technique that can be used on the Internet to provide streaming media support for complex applications like video-on-demand, distance learning, distributed collaboration, distributed games, and large-scale wireless communication. The delivery of media through multicast often involves using Internet infrastructure, devices and protocols, including protocols for routing and TCP/IP.

14. Starting in 1997, I worked on a project to integrate the streaming media capabilities of the Internet together with the interactivity of the web. I developed a project called the Interactive Multimedia Jukebox (IMJ). Users would visit a web page and select content to view. The content would then be scheduled on one of a number of channels, including delivery to students in Georgia Tech dorms delivered

via the campus cable plant. The content of each channel was delivered using multicast communication.

15. In the IMJ, the number of channels varied depending on the capabilities of the server including the available bandwidth of its connection to the Internet. If one of the channels was idle, the requesting user would be able to watch their selection immediately. If all channels were streaming previously selected content, the user's selection would be queued on the channel with the shortest wait time. In the meantime, the user would see what content was currently playing on other channels, and because of the use of multicast, would be able to join one of the existing channels and watch the content at the point it was currently being transmitted.

16. The IMJ service combined the interactivity of the web with the streaming capabilities of the Internet to create a jukebox-like service. It supported true Video-on-Demand when capacity allowed, but scaled to any number of users based on queuing requested programs. As part of the project, we obtained permission from Turner Broadcasting to transmit cartoons and other short-subject content. We also connected the IMJ into the Georgia Tech campus cable television network so that students in their dorms could use the web to request content and then view that content on one of the campus's public access channels.

17. More recently, I have also studied issues concerning how users choose content, especially when considering the price of that content. My research has examined how dynamic content pricing can be used to control system load. By raising prices when systems start to become overloaded (i.e., when all available resources are fully utilized) and reducing prices when system capacity is readily available, users' capacity to pay as well as their willingness can be used as factors in stabilizing the response time of a system. This capability is particularly useful in systems where content is downloaded or streamed on-demand to users.

18. As a parallel research theme, starting in 1997, I began researching issues related to wireless devices and sensors. In particular, I was interested in showing how to provide greater communication capability to "lightweight devices," i.e., small form-factor, resource-constrained (e.g., CPU, memory, networking, and power) devices. Starting in 1998, I published several papers on my work to develop a flexible, lightweight, battery-aware network protocol stack. The lightweight protocols we envisioned were similar in nature to protocols like Bluetooth, Universal Plug and Play (UPnP) and Digital Living Network Alliance (DLNA).

19. From this initial work, I have made wireless networking—including ad hoc, mesh networks and wireless devices—one of the major themes of my research. My work in wireless networks spans the protocol stack from applications through to the encoding and exchange of data at the data link and physical layers.

20. At the application layer, even before the large-scale “app stores” were available, my research looked at building, installing, and using apps for a variety of purposes, from network monitoring to support for traditional computer-based applications (e.g., content retrieval) to new applications enabled by ubiquitous, mobile devices. For example, my research has looked at developing applications for virtually exchanging and tracking “coupons” through “opportunistic contact” among mobile wireless devices (i.e., communication among devices moving into communication range with each other). In many of the courses I have taught there is a project component. Through these projects I have supervised numerous efforts to develop new “apps” for download and use across a variety of mobile platforms.

21. Toward the middle of the protocol stack, my research has also looked to build wireless infrastructure support to enable communication among a set of mobile devices unaided by any other kind of network infrastructure. These kinds of networks are useful either in challenged network environments (e.g., when a natural disaster has destroyed existing infrastructure) or when suitable support for network communication never existed. The deployment of such networks (or even the use of traditional network support) are critical to support services like disaster relief, catastrophic event coordination, and emergency services deployment.

22. Yet another theme is monitoring wireless networks, in particular different variants of IEEE 802.11 compliant networks, to (1) understand the

operation of the various protocols used in real-world deployments, (2) use these measurements to characterize use of the networks and identify protocol limitations and weaknesses, and (3) propose and evaluate solutions to these problems. I have successfully used monitoring techniques to study wireless data link layer protocol operation and to improve performance by enhancing the operation of such protocols. For wireless protocols, this research includes functions like network acquisition and channel bonding.

23. One theme in my wireless network research has been cross-layer solutions and innovations. As mentioned above, with greater wireless device use and network support, we envisioned new application paradigms and services, for example, when mobile devices come into contact with each other. Instead of relying on existing infrastructure to relay communication, the devices are able to discover each other and communicate directly. Other examples include discovering and using location information to enhance users' experiences. Network support and novel applications span use a variety of network architectures supporting users on foot, in vehicles, and across varying terrains and environments. Finally, we studied how communication efficiency can be supported through intelligent handoffs as well as location and movement prediction.

24. Protecting networks, including their operation and content, has been an underlying theme of my research almost since the beginning of my research career.

Starting in 2000, I have been involved in several projects that specifically address security, network protection, and firewalls. After significant background work, a team on which I was a member successfully submitted a \$4.3M grant proposal to the Army Research Office (ARO) at the Department of Defense to propose and develop a high-speed intrusion detection system. Key aspects of the system included associating streams of packets and analyzing them for viruses and other malware. Once the grant was awarded, we spent several years developing and meeting the milestones of the project. A number of my students worked on related projects and published papers on topics ranging from intrusion detection to developing advanced techniques to be incorporated into firewalls. I have also used firewalls, including their associated malware detection features, in developing techniques for the classroom to ensure that students are not distracted by online content.

25. My recent work ties some of the various threads of my past research together. I have investigated content delivery in online social networks and proposed reputation management systems in large-scale social networks and marketplaces. On the content delivery side, I have looked at issues of caching and cache placement, especially when content being shared and the cache has geographical relevance. We were able to show that effective caching strategies can greatly improve performance and reduce deployment costs. Our work on reputation systems showed that reputations have economic value, and as such, creates a motivation to manipulate

reputations. In response, we developed a variety of solutions to protect the integrity of reputations in online social networks. The techniques we developed for content delivery and reputation management were particularly relevant in peer-to-peer communication and recommendations for downloadable “apps.”

26. As an important component of my research program, I have been involved in the development of academic research into available technology in the market place. One aspect of this work is my involvement in the Internet Engineering Task Force (IETF). The IETF is a large and open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. I have been involved in various IETF groups including many content delivery-related working groups like the Audio Video Transport (AVT) group, the MBone Deployment (MBONED) group, Source Specific Multicast (SSM) group, the Inter-Domain Multicast Routing (IDMR) group, the Reliable Multicast Transport (RMT) group, the Protocol Independent Multicast (PIM) group, etc. I have also served as a member of the Multicast Directorate (MADDOGS), which oversaw the standardization of all things related to multicast in the IETF. Finally, I was the Chair of the Internet2 Multicast Working Group for seven years.

27. My involvement in the research community extends to leadership positions for several academic journals and conferences. I am the co-chair of the

Steering Committee for the ACM Network and System Support for Digital Audio and Video (NOSSDAV) workshop and on the Steering Committees for the International Conference on Network Protocols (ICNP), ACM Sigcomm Workshop on Challenged Networks (CHANTS), and IEEE Global Internet (GI) Symposium. I have served or am serving on the Editorial Boards of IEEE/ACM Transactions on Networking, IEEE Transactions on Mobile Computing, IEEE Network, ACM Computers in Entertainment, AACE Journal of Interactive Learning Research (JILR), and ACM Computer Communications Review. I have co-chaired a number of conferences and workshops including the IEEE International Conference on Network Protocols (ICNP), IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), International Conference on Communication Systems and Networks (COMSNETS), IFIP/IEEE International Conference on Management of Multimedia Networks and Services (MMNS), the International Workshop On Wireless Network Measurement (WiNMee), ACM Sigcomm Workshop on Challenged Networks (CHANTS), the Network Group Communication (NGC) workshop, and the Global Internet Symposium, and I have served on the program committees for numerous conferences.

28. Furthermore, in the courses I taught at UCSB, a significant portion of my curriculum covered aspects of the Internet and network communication including the physical and data link layers of the Open System Interconnect (OSI)

protocol stack, and standardized protocols for communicating across a variety of physical media such as cable systems, telephone lines, wireless, and high-speed Local Area Networks (LANs). The courses I have taught also cover most major topics in Internet communication, including data communication, multimedia encoding, and mobile application design. My research and courses have covered a range of physical infrastructures for delivering content over networks, including cable, Integrated Services Digital Network (ISDN), Ethernet, Asynchronous Transfer Mode (ATM), fiber, and Digital Subscriber Line (DSL). For a complete list of courses I have taught, see my curriculum vitae (CV).

29. In addition, I co-founded a technology company called Santa Barbara Labs that was working under a sub-contract from the U.S. Air Force to develop very accurate emulation systems for the military's next generation internetwork. Santa Barbara Labs' focus was in developing an emulation platform to test the performance characteristics of the network architecture in the variety of environments in which it was expected to operate, and, in particular, for network services including IPv6, multicast, Quality of Service (QoS), satellite-based communication, and security. Applications for this emulation program included communication of a variety of multimedia-based services, including video conferencing and video-on-demand.

30. In addition to having co-founded a technology company myself, I have worked for, consulted with, and collaborated with companies for nearly 30 years.

These companies range from well-established companies to start-ups and include IBM, Hitachi Telecom, Turner Broadcasting System (TBS), Bell South, Digital Fountain, RealNetworks, Intel Research, Cisco Systems, and Lockheed Martin.

31. Through my graduate education, leadership with CITS, involvement in TMP, role in the development of the Internet2 infrastructure, and consulting with ISPs, I have gained a strong understanding in the role of the Internet in our society and the challenges of deploying large-scale production networking infrastructure. CITS, since its inception, has looked at the role of the Internet in society, including how the evolution of technology have created communication opportunities and challenges, including, for example through disruptive technologies like P2P. TMP looks to focus on non-purely technical issues, including, for example, state-of-the-art business methods, strategies for successful technology commercialization, new venture creation, and best practices for fostering innovation. Through my industry collaborations and Internet2 work, I have developed significant experience in the challenges of deploying, monitoring, managing, and scaling communication infrastructure to support evolving Internet services like streaming media, conferencing, content exchange, social networking, and e-commerce.

32. I am a Member of the Association of Computing Machinery (ACM) and a Fellow of the Institute of Electrical and Electronics Engineers (IEEE).

III. LEVEL OF ORDINARY SKILL IN THE ART

33. I understand there are multiple factors relevant to determining the level of ordinary skill in the pertinent art, including (1) the levels of education and experience of persons working in the field at the time of the invention; (2) the sophistication of the technology; (3) the types of problems encountered in the field; and (4) the prior art solutions to those problems.

34. A Person of Ordinary Skill in The Art (“POSITA”) in October 2000 would have had a working knowledge of the telecommunications and networking art that is pertinent to the ’244 Patent. A POSITA would have had a bachelor’s degree or graduate degree in computer science, computer engineering, or an equivalent, and two years of relevant experience in networking and network security, and related fields. Lack of professional experience can be remedied by additional education, and vice versa.

35. For purposes of this Declaration, in general, and unless otherwise noted, my statements and opinions, such as those regarding my own experience and what a POSITA would have understood or known generally (and specifically related to the references I consulted herein), reflect the knowledge that existed in the relevant field as of the priority date of the ’244 Patent.

IV. RELEVANT LEGAL STANDARDS

36. I am not an attorney. In preparing and expressing my opinions and considering the subject matter of the '244 Patent, I am relying on certain basic legal principles that counsel has explained to me.

37. I understand that prior art to the '244 Patent includes patents and printed publications in the relevant art that predate the priority date of the '244 Patent. For purposes of this Declaration, I am applying October 31, 2000, as the priority date of the '244 Patent, but I take no position on that issue.

38. I have been informed by counsel that a claimed invention is invalid if it is “anticipated” by the prior art. I understand that, for a patent claim to be “anticipated” by the prior art, each and every requirement of the claim must be found, expressly or inherently, in a single prior art reference as recited in the claim. I understand that claim limitations that are not expressly described in a prior art reference may still be there if they are “inherent” to the thing or process being described in the prior art.

39. I understand that if a reference incorporates other documents by reference, the incorporating reference and the incorporated reference(s) should be treated as a single prior art reference for purposes of analyzing anticipation.

40. I understand that it is acceptable to consider evidence other than the information in a particular prior art document to determine if a feature is necessarily present in or inherently described by that reference.

41. I have been informed by counsel that a claimed invention is unpatentable under 35 U.S.C. § 103 if the differences between the claimed invention and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a POSITA. I have also been informed by counsel that the obviousness analysis considers factual inquiries, including the level of ordinary skill in the art, the scope and content of the prior art, and the differences between the prior art and the claimed subject matter.

42. I have been further informed by counsel that there are several recognized rationales for combining references or modifying a reference to show obviousness. These rationales include: (a) combining prior art elements according to known methods to yield predictable results; (b) simple substitution of one known element for another to obtain predictable results; (c) use of a known technique to improve a similar device (method, or product) in the same way; (d) applying a known technique to a known device (method, or product) ready for improvement to yield predictable results; (e) choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success; and (f) some teaching,

suggestion, or motivation in the prior art that would have led a POSITA to modify the prior art or to combine prior art teachings to arrive at the claimed invention.

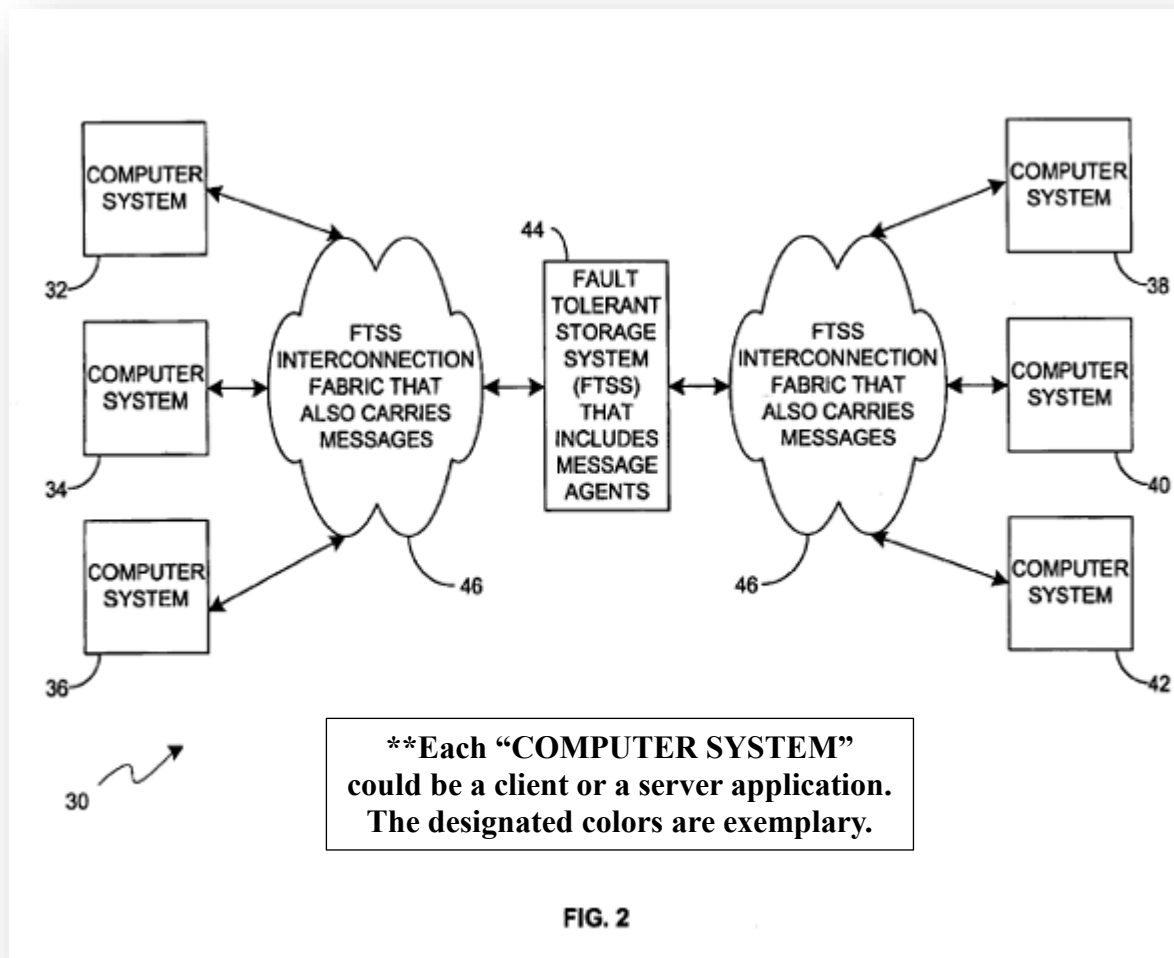
V. OVERVIEW OF THE '244 PATENT

43. The '244 Patent is directed to messaging between computer systems. The '244 Patent describes “a messaging architecture wherein messages are transmitted over the interconnection fabric of a fault tolerant storage system and are stored within the fault tolerant storage system [(FTSS)].” Ex-1101, 1:21-24. “The interconnection fabric [] couples the FTSS to the computer systems that host the client and server applications.” Ex-1101, 5:11-14. The FTSS both “processes file-related I/O transactions” and “carries message traffic.” Ex-1101, 5:17-19. The FTSS also “includes several message agents to facilitate message transfer in a reliable and fault tolerant manner.” Ex-1101, 5:19-21. These message agents comprise “a conversational communication agent, a queue-based communication agent, a request/reply communication agent, and an unsolicited communication agent.” 5:21-25. Once a message is received by the FTSS, “the message is committed to a nonvolatile write cache” and then “written to a redundant array of independent disks (RAID) of the FTSS, and processed by one of the message agents.” Ex-1101, 5:29-35.

44. The '244 Patent further explains that the present invention “enhances the reliability and fault tolerance of messaging” by (a) using the FTSS

interconnection fabric to transmit messages between nodes so as to “increase[] availability of the distributed application without the customer having to incur additional hardware costs”; (b) moving messaging paradigms that traditionally would operate on separate servers to the FTSS, again increasing reliability and “eliminat[ing] the need to route the messages to a separate server, thereby allowing the messages to be processed faster”; and (c) passing interprocess communication messages that would traditionally pass between nodes through the FTSS so that those messages are “stored in and retained by the FTSS, thereby allowing any application to ‘rollback’ to a known point and retrieve lost messages from the FTSS.” Ex-1101, 5:35-58.

45. Figure 2, annotated below, depicts an embodiment of the invention described in the '244 Patent:



Ex-1101, Fig. 2 (annotated).

46. Figure 2 includes a “Networked system 30,” which “includes computer systems 32, 34, 36, 38, 40, and 42, fault tolerant storage system (FTSS) 44, and FTSS interconnection fabric 46.” Ex-1101, 6:28-33. The ’244 Patent notes that “[a]lthough only computer systems are shown in FIG. 2...other network nodes may be coupled to FTSS 44 via fabric 46, such as printers, scanners, backup systems, and the like,” and the computer systems “may be coupled to each other and to other

computers via another interconnection fabric, such as an Ethernet network, which in turn may be coupled to the Internet via a gateway.” Ex-1101, 6:34-41.

47. Finally, the ‘244 Patent states that the messages passed between these various “network nodes are processed...using a middle ware product.” Ex-1101, 7:3-4. The middleware supports the messaging paradigms listed above. The communication between nodes is facilitated by communication agents, which are responsible for “retain[ing the] old message transactions for some time after the message transactions are complete.” Ex-1101, 8:34-36.

VI. CLAIM CONSTRUCTION

48. It is my understanding that to properly evaluate the ’244 Patent, the terms of the claims must first be interpreted. It is my understanding that for the purposes of this *inter partes* review, the claims are to be given their “ordinary and customary meaning” as understood by a POSITA in light of the specification and prosecution history, unless the inventor has set forth a special meaning for a term. I have been informed that Patent Owner has taken two positions through its claim-construction expert in a parallel proceeding that Petitioner has adopted in this Petition without agreeing those positions are correct. First, it is my understanding that Patent Owner has asserted that “*coupled to*” as used in the claims to describe connectivity between nodes and the fault tolerant storage system (e.g., “*wherein the first and second nodes are each coupled to a fault tolerant storage system (FTSS)*”

in claim 1) means that the “nodes are each communicatively connected to the fault tolerant storage system.” Ex-1110, ¶ 50. Second, it is my understanding that Patent Owner has asserted that “*highly reliable*” and “*fault tolerant*” as used in the claims (e.g., “*highly reliable fault-tolerant storage media*” in claim 1) “would not refer to any specific value or measurement of reliability, but rather as itself a quality that a system could provide” (*id.*, ¶ 67), such that a POSITA would have understood “‘highly reliable’ and ‘fault tolerant’ to generally describe systems with additional indications or guarantees of reliability beyond non-redundant, single point of failure systems” (*id.*, ¶¶ 68-69). I have also been informed that claim terms only need to be construed to the extent necessary to resolve the unpatentability inquiry.

VII. IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE

49. A POSITA would find the challenged claims unpatentable under any interpretation consistent with their plain and ordinary meaning. The following discussion in this Declaration provides a detailed analysis of how the asserted prior art teaches each limitation of the Challenged Claims.

50. As part of my analysis, I have considered, and discuss in detail, the scope and content of the prior art and any differences between the alleged invention and the prior art.

51. It is my opinion that the alleged invention recited in the Challenged Claims would have been obvious in view of the teachings of the asserted prior art and the knowledge of a POSITA.

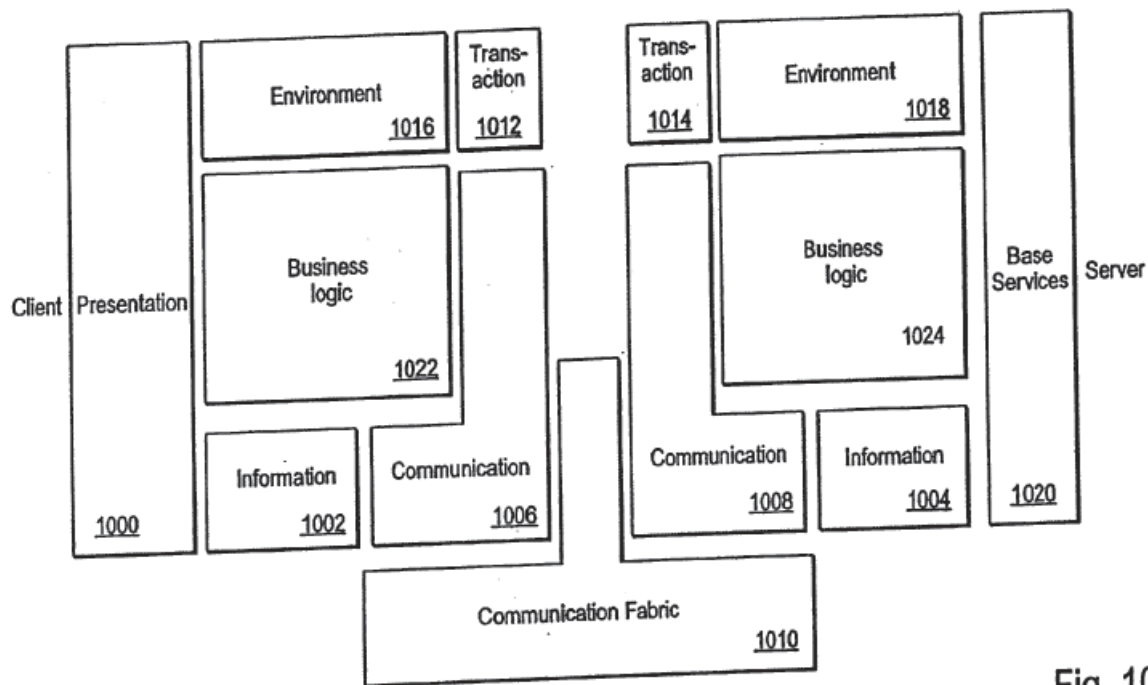
VIII. THE PRINCIPAL PRIOR ART

A. Summary of Bowman

52. U.S. Patent No. 6,615,253 to Bowman-Amuah (“Bowman”) (Ex-1105) was filed on August 31, 1999, and issued on September 2, 2003. Bowman is prior art to the ’244 Patent at least under pre-AIA 35 U.S.C. § 102(e).

53. Bowman is directed to “updating a server while minimizing network traffic and maintaining an acceptable level of system performance.” Ex-1105, 1:20-24.

54. Bowman teaches a system for efficiently retrieving data in which data from a server is requested over a network, bundled into a data structure by the server, sent to the client over the network, and then cached on the client. *Id.*, Abstract. A high-level overview illustrating the architecture is depicted below with respect to Figure 10:

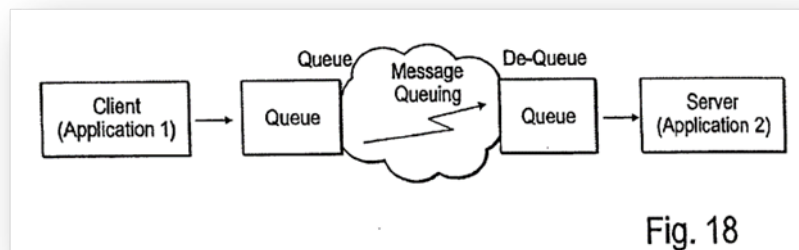


Id., Fig. 10.

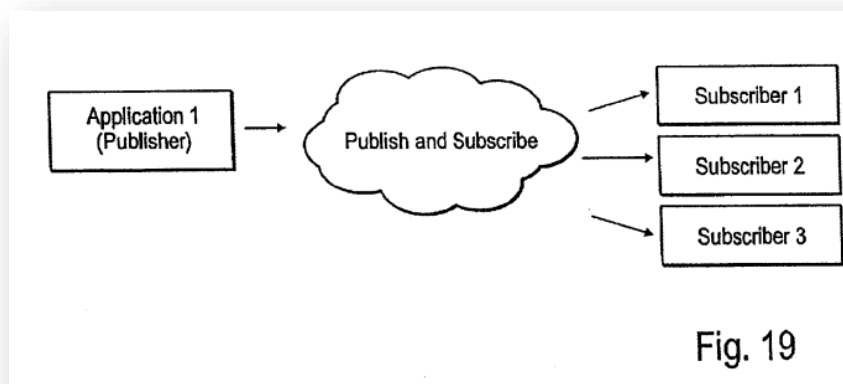
55. Bowman utilizes an “enhanced” client/server model. *Id.*, 34:1-2. Bowman discloses that in this model (shown in Fig. 10), “the services can be broken down into logical areas” including “Communication Services 1006, 1008” and “Communication Fabric Services 1010.” *Id.*, 31:54-57. As such, “all presentation and control logic resides on the client, all application logic resides on multiple back-end application servers, and all data management logic resides on multiple back-end database servers.” *Id.*, 34:5-9.

56. Bowman teaches that a so-called “Message Transport service is responsible for the end-to-end delivery of messages.” *Id.*, 84:50-51. Bowman further

teaches that a “communication fabric is aware of the different message-oriented information streams in order to help the client and server communicate regardless of the different network functions implemented on each platform.” *Id.*, 84:5-9. Messages may be queued for later processing (*see* Figure 18 below) or sent via a “publish and subscribe” method “(also known as Push messaging),” (*see* Figure 19 below). *Id.*, 70:22-36. This flow of information is actively managed by an “intelligent network” that “integrates heterogeneous clients, servers, and other resources by resolving incompatible protocols and standards.” *Id.*, 84:22-24.



Id., Fig. 18.



Id., Fig. 19.

57. Bowman teaches that it is desirable for a system and method to include safeguards that ensure the reliable delivery of messages. For example, it discloses interconnection fabrics in the form of its Message-Oriented Middleware that is “responsible for managing the interface to the underlying communications architecture via the communication protocol APIs and ensuring the delivery of the information to the remote process.” *Id.*, 69:53-58.

B. Summary of Vahalia

58. U.S. Patent No. 6,973,455 to Vahalia et al. (“Vahalia”) (Ex-1106) was filed on March 3, 1999, and issued on December 6, 2005. Vahalia is prior art to the ’244 Patent at least under pre-AIA 35 U.S.C. § 102(e).

59. Vahalia is directed to “data storage systems, ... particularly network file servers.” Ex-1106, 1:9-10. Vahalia discloses that a “cached disk array has been interfaced to a data network via at least one data mover computer. The data mover computer receives data access commands from clients in the data network in accordance with a network file access protocol.” *Id.*, 1:20-24. This type of architecture is needed, since “[m]ainframe data processing, and more recently distributed computing, have required increasingly large amounts of data storage. This data storage is most economically provided by an array of low-cost disk drives integrated with a large semiconductor cache memory.” *Id.*, 1:12-16.

60. An exemplary embodiment of the Vahalia invention is shown in Figure 3 (below):

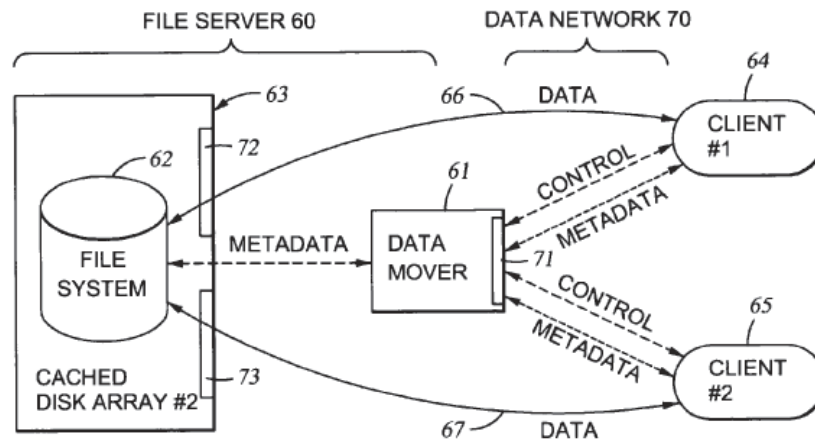


Fig. 3

Id., Figure 3. In one aspect of the invention, the client sends requests to the file server through the data mover where it responds via the data mover with metadata. The file server has a cached disk array and data mover. One of the key features of the Vahalia invention is that the “file server 60 including a data mover 61 and a cached disk array 63 provides direct data sharing between network clients 64, 65 by arbitrating and coordinating data access requests.” *Id.*, 35:25-27.

C. Summary of Tuxedo

61. The Tuxedo™ System by Juan M. Andrade, Mark T. Carges, Terence J. Dwyer, and Stephen D. Felts (“Tuxedo”) (Ex-1107) was published by Addison-Wesley Publishing Co. in 1996. Tuxedo is prior art to the '244 Patent at least under 35 U.S.C. § 102(b).

62. Tuxedo is a textbook that describes the TUXEDO System, a “set of software modules that enables the construction, execution, and administration of high performance, distributed business applications.” Ex-1107, xxxiii. It offers a number of communication paradigms including “Request/Response,” “Conversational,” “Event” Driven, “Subscription” and “Queue[s],” shown in the figures below:

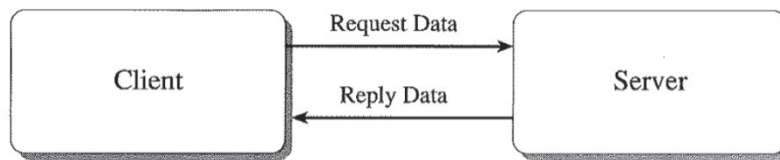


Figure 3.2 Request/Response Interaction

Id., Figure 3.2.

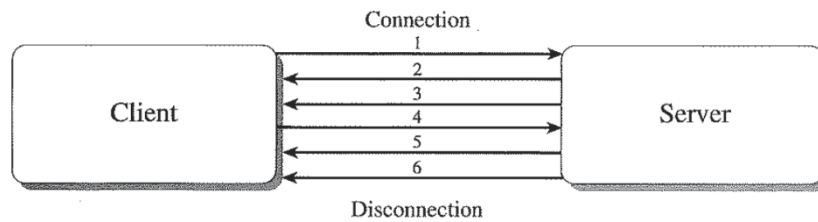


Figure 3.6 Conversational Programming Paradigm

Id., Figure 3.6.

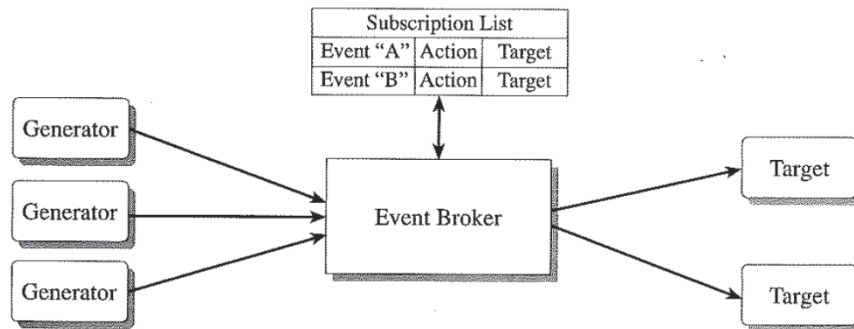


Figure 3.8 Delivery of Events via a Broker

Id., Figure 3.8.

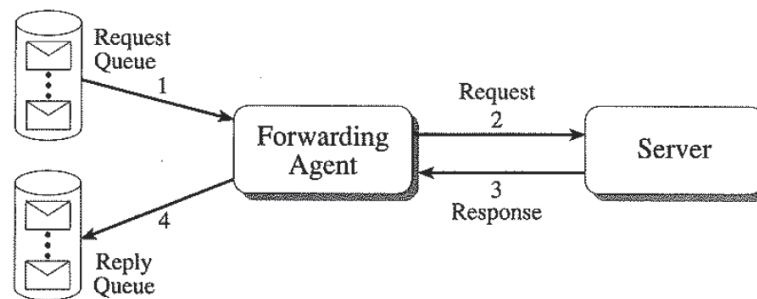


Figure 3.10 Forwarding Agent Converts Queue Operations to Request/Response

Id., Figure 3.10.

63. Note that for multiple clients and servers, as shown in Figure 3.8 and 3.10, there is an intermediary facilitating the communication. Tuxedo shows (in Figure 6.1) the “Conceptual Architecture of a Distributed TUXEDO Application.” *Id.*, 108.

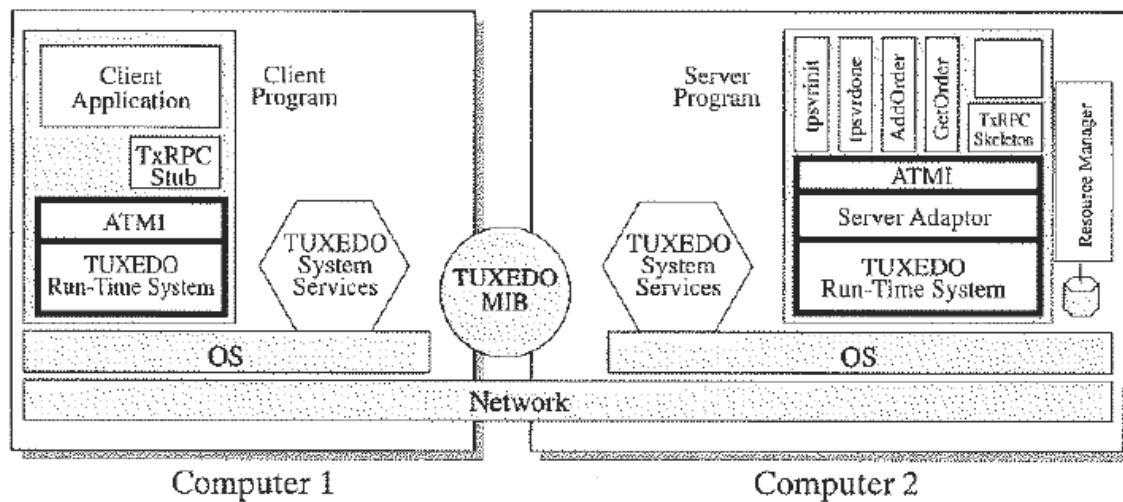


Figure 6.1 Conceptual Architecture of a Distributed TUXEDO Application

Id., Figure 6.1.

64. Tuxedo also introduces the notion of fault-tolerance, where “[t]he administrator can also configure another machine, the Backup machine, that can act as the Master machine. If the Master machine goes down, the administrator can...activate[]...the Backup machine.” *Id.*, 120. One of the goals in the system design, “is that there be no single point of failure...through the use of redundant hardware and software.” *Id.*, 313. For “applications that require high availability, hardware support is needed. Mirroring of disks...can be used used to avoid failure due to the loss of a single disk.” *Id.*, 332.

D. Motivation to Combine

65. In this section, I will describe motivations to combine three prior-art references: Bowman, Vahalia, and Tuxedo. Among Bowman’s teachings are

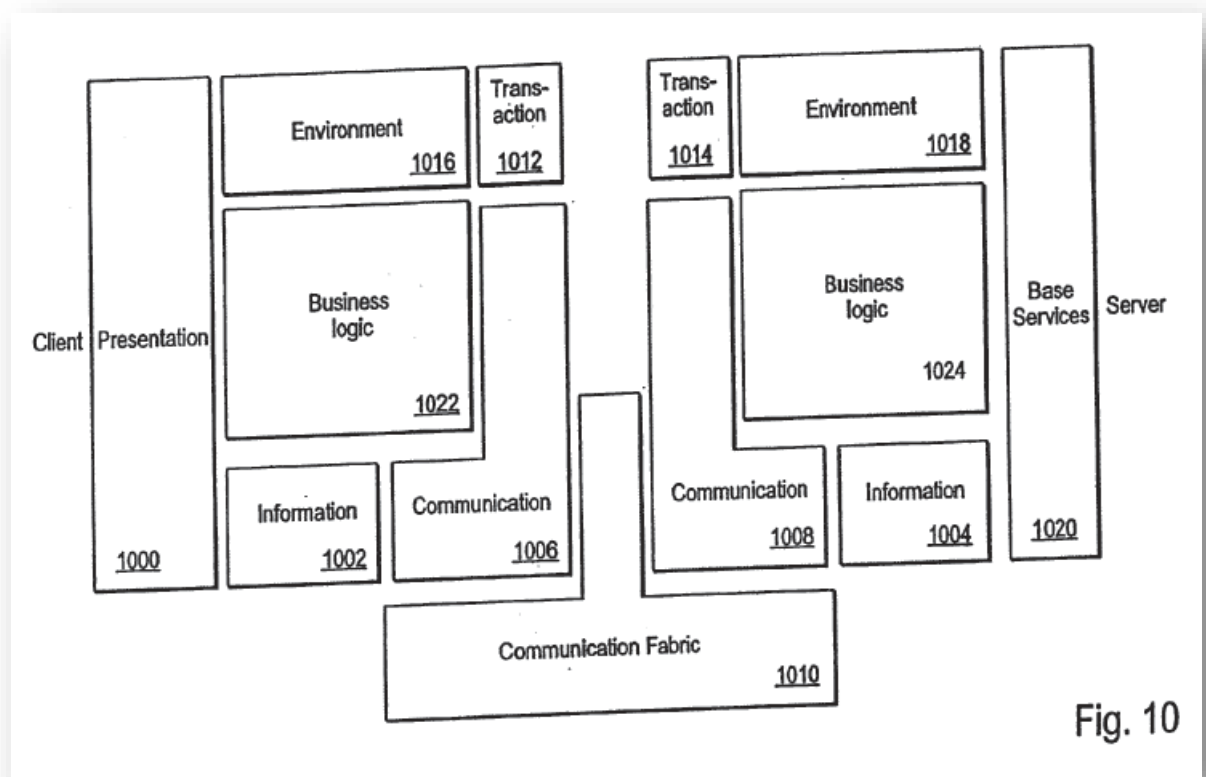
methods and systems related to transferring data efficiently between clients and servers on networks. Ex-1105, Abstract. It teaches or suggests several approaches to achieving this efficiency, two of which are relevant: using middleware agents to implement communication paradigms and using disk arrays to improve the fault tolerance of network file systems. Ex-1105, 69:53-58, 58:58-69:10. Vahalia is similarly concerned with increasing the fault tolerance of network file systems and provides additional implementation details for the disk arrays taught or suggested in Bowman. Ex-1106, 1:9-15, 1:20-24, Fig. 3. Vahalia likewise teaches implementing middleware agents for data transfer and teaches implementing the middleware agents on a file server comprising a disk array. *Id.* As to implementation details for the communication paradigms, Bowman explicitly discloses using Tuxedo. Ex-1105, 57:9-18, 34:18-20, 95:15-16. Tuxedo augments Bowman's teachings with more implementation details for various communication paradigms, including request/response, conversations, application queuing, and events. Ex. 1007, 131, 147, 190. For the further reasons below, it would have been obvious to a POSITA to combine Bowman, Vahalia, and Tuxedo.

1. Bowman / Vahalia combination

66. A POSITA would have found it obvious to use disk arrays in its fault tolerant file sharing, as taught by Vahalia. Ex-1105; Ex-1107.

67. Bowman expressly teaches a system for efficiently retrieving data in which data from a server is requested over a network, bundled into a data structure by the server, and sent to the client over the network. Ex-1105, Abstract.

68. A high-level overview is depicted below with respect to annotated Figure 10:



Ex-1105, Fig. 10.

69. Bowman utilizes a client/server model. Ex-1105, 34:1-2. Bowman discloses that in this model, and as shown in Fig. 10, “the services can be broken down into logical areas” including “Communication Services 1006, 1008” and “Communication Fabric Services 1010.” Ex-1105, 31:54-57.

70. Bowman teaches that a so-called “Message Transport service is responsible for the end-to-end delivery of messages.” Ex-1105, 84:50-51. Bowman further teaches that a “communication fabric is aware of the different message-oriented information streams in order to help the client and server communicate regardless of the different network functions implemented on each platform.” Ex-1105, 84:5-9. To facilitate communication between clients and servers, Bowman describes “Message-Oriented Middleware (MOM),” which “refers to the process of distributing data and control throughout the exchange of records known as messages.” Ex-1105, 69:47-56. And “Message-Oriented Middleware is responsible for managing the interface to the underlying communications architecture via the communications protocol APIs and ensuring the delivery of the information to the remote process,” including “[t]ransferring data and control messages during the session.” Ex-1105, 69:53-65.

71. Bowman teaches that it is desirable for a system and method to include safeguards that ensure the reliable delivery of messages. Bowman teaches that the Netcentric Architecture Framework has fault-tolerant functionality. . For example, Bowman explains that Communications Fabric 1010 “provides common network services to the platform-specific network services residing on the client and server nodes.” Ex-1105, 83:66-84:1. Among the common network services that Bowman describes are File Sharing Services 1512. *See, e.g.*, Ex-1105, 58:50-59:50, Fig. 16.

72. Bowman explains that Netcentric Architecture Framework has or can integrate with fault-tolerant hardware. Bowman explains that “File Sharing services allow users to view, manage, read, and write files that may be located on a variety of platforms in a variety of locations.” Ex-1105, 58:50-54. The “File Sharing services can provide the following capabilities,” which includes “Fault tolerance,” such as the “use of primary and replica file systems to ensure high availability of the file system.” Ex-1105, 58:58-69:10.

73. A POSITA would have also understood Bowman’s reference to primary and replica file systems to teach or suggest disk arrays (e.g., RAID (Redundant Arrays of Independent Disks)) that protects against a single point of failure. Ex-1108, 28 (“Disk arrays are groups of disk drives that work together to achieve higher data-transfer and I/O rates than those provided by single large drives. Arrays can also provide data redundancy so that no data is lost if a single drive (physical disk) in the array should fail. Depending on the RAID level, data is either mirrored or striped.”); Ex-1114, 144; Ex-1112, 464 (defining “RAID”). A POSITA would have also understood RAID (Redundant Arrays of Independent Disks) to describe a disk array arrangement (like that in Vahalia) that protects against a single point of failure—e.g., using disk arrays for disk mirroring as provided in RAID provides an increased indication of reliability by providing for redundancy, such that if one drive in the array fails, data is not lost. Ex-1108, 28 (“Disk arrays are groups of disk drives that

work together to achieve higher data-transfer and I/O rates than those provided by single large drives. Arrays can also provide data redundancy so that no data is lost if a single drive (physical disk) in the array should fail. Depending on the RAID level, data is either mirrored or striped.”); Ex-1114, 144; Ex-1112, 464 (defining “RAID” as “a category of disk drives that employ two or more drives in combination for fault tolerance” and noting it “[p]rovides disk mirroring”). A POSITA would have recognized that systems and methods (such as use of the cached disk array in Vahalia) offer safeguards (e.g., replication) contemplated by Bowman.

74. Accordingly, a POSITA would have been motivated to look to Vahalia to provide more details for a fault tolerant system using disc arrays.

75. A POSITA would have been motivated to rely on Vahalia’s use of disc arrays, exemplified by the File Server 60 below, to satisfy Bowman’s explicit interest in increasing fault tolerance. Vahalia includes a block diagram of a preferred embodiment:

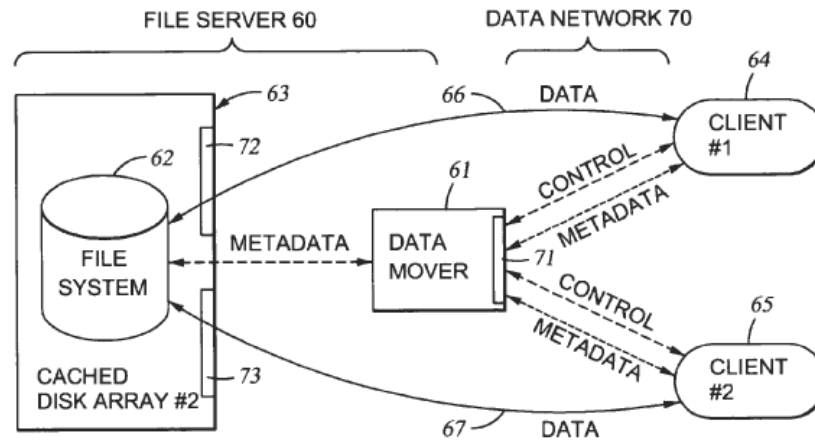


Fig. 3

Id., Fig. 3.

76. Vahalia teaches a data storage system that uses “an array of low-cost disk drives integrated with a large semiconductor cache memory.” Ex-1106, 1:9-15. Vahalia also teaches that its disk array is coupled over a network (e.g., “interfaced to a data network via at least one data mover computer”). Ex-1106, 1:20-21. Annotated Figure 3 of Vahalia shows “Cached Disk Array #2” (red) as part of File Server 60 (yellow):

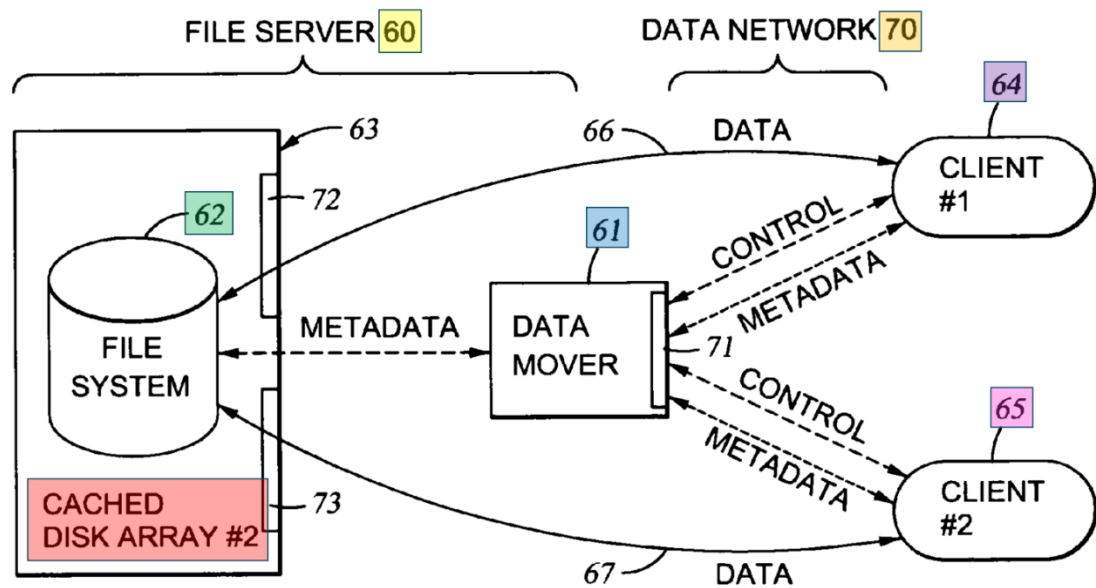


Fig. 3

77. Vahalia further teaches that client devices (e.g., Client #1, 64 (purple) and Client # 2, 65 (pink)) move data to and from Cached Disk Array #2 over Data Network 70 via a Data Mover 61 (blue) that is part of File Server 60:

In the preferred implementation of the system of FIG. 3, the clients 64, 65 may mount file systems on the cached disk array 63 by sending NFS commands to the data mover 61. The clients also have a configuration file associating volumes with file systems. The clients move file data directly to the cached disk array 63 using a high-speed data protocol such as is commonly used to read or write data directly to a disk drive over an SCSI or Fibre Channel link.

Ex-1106, 35:47-55.

78. Moreover, a POSITA would have understood that Vahalia's Data Network 70 could be implemented over Bowman's Communications Fabric 1010. Ex-1105, 83:56-84:3 (describing Communications Fabric 1010 as providing for “common network services”); Ex-1106, 8:51-9:17 (describing Data Network 70 as enabling common client-server network communications). This would result in the combined system of Bowman / Vahalia using the Data Mover 61 as the middleware described in Bowman.

79. It would have thus been obvious for a POSITA to try implementing Vahalia's teaching of a file server comprising disk arrays and data mover to satisfy Bowman's explicit interest in increasing fault tolerance through primary and replica file systems. It would also have been obvious to include a known element (Vahalia's teaching of a file server comprising a disk array and data mover) into another mechanism (Bowman's teaching of a fault tolerant file sharing system) that yield predictable results (providing fault tolerance to Bowman's Filesharing Service using Vahalia's File Server comprising a disk array). Ex-1105; Ex-1106.

80. Both Bowman and Vahalia are analogous art to the '244 Patent. Bowman and Vahalia are both directed to a similar field of endeavor of the '244 Patent— fault tolerant client-server communication. Ex-1101, Abstract; Ex-1105, Abstract; Ex-1106, Abstract. Bowman is directed to “updating a server while minimizing network traffic and maintaining an acceptable level of system

performance.” Ex-1105, 1:20-24. Vahalia is directed to data storage systems in relation to network file servers. Ex-1106 at 1:9-10.

81. Second, both Bowman and Vahalia are reasonably pertinent to a problem described by the ’244 Patent—the need for reliable message delivery. Ex-1101, Abstract, 3:65-4:8.

82. Vahalia expressly teaches that “recently, the cached disk array has been interfaced to a data network via at least one data mover computer. The data mover computer receives data access commands from clients in the data network in accordance with a network file access protocol.” *Id.* at 1:20-24. Vahalia also teaches that “[m]ainframe data processing, and more recently distributed computing, have required increasingly large amounts of data storage. This data Storage is most economically provided by an array of low-cost disk drives integrated with a large semiconductor cache memory.” *Id.* at 1:12-16. A POSITA would have recognized that including Vahalia’s cached disk array within Bowman’s communication fabric would increase the reliability of Bowman’s messaging system in the same way Bowman expresses is desirable and Vahalia explains would be accomplished.

83. A POSITA would have expected to succeed in combining Bowman and Vahalia due to similarity in structure and solutions to perceived problems with the prior art. Immediate similarities to Bowman are apparent when Figure 10 of Bowman is compared to Figure 3 of Vahalia.

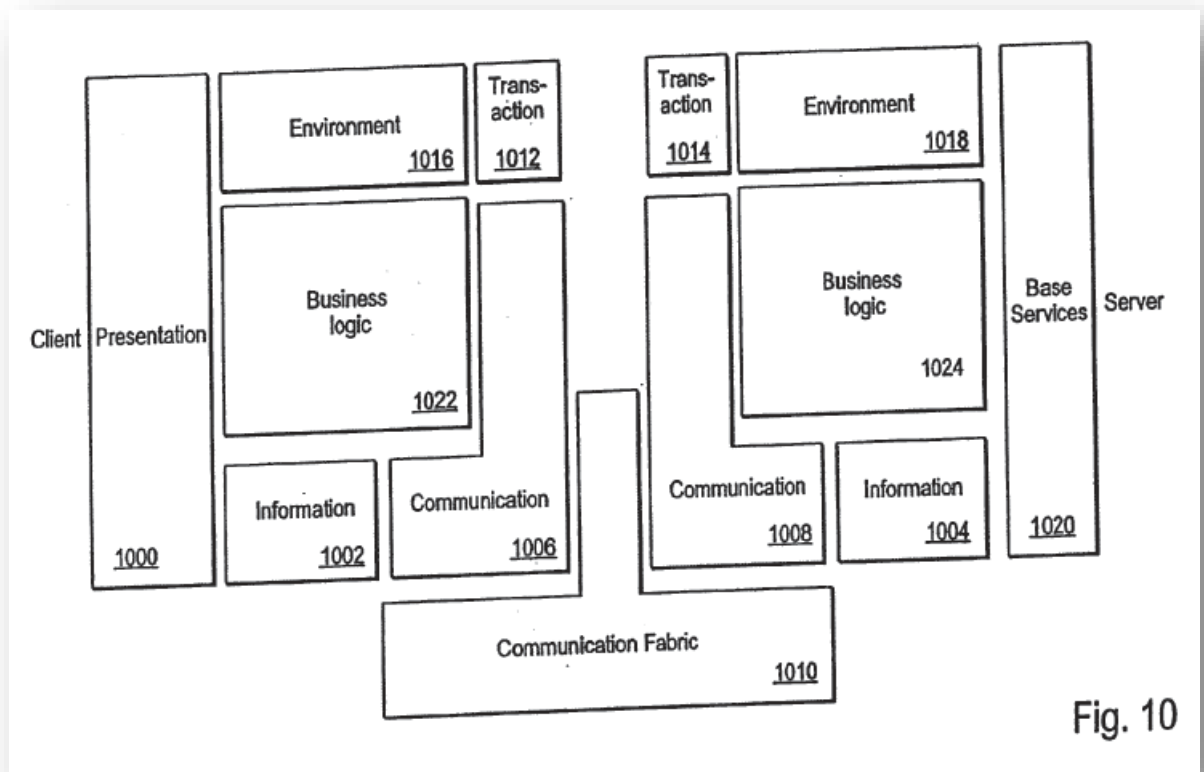


Fig. 10

See Ex-1105 at Fig. 10.

84. Each system has a first computer (“Client” in Bowman, “Client 1” in Vahalia), and second computer (“Server” and “Client 2” respectively), and an interconnection fabric (“Communication Fabric” and “Data Network 70” respectively). And, as discussed above, Bowman explicitly teaches implementing a disc array to create a fault tolerant file sharing system and a POSITA would have expected to succeed in accomplishing fault tolerance through the use of Vahalia.

2. Bowman / Vahalia / Tuxedo combination

85. A POSITA implementing the Bowman / Vahalia system, which implements Vahalia's Data Mover 61 as middleware, would have been motivated to use Tuxedo's messaging paradigms.

86. Bowman teaches that:

Broadly defined, Messaging services enable information or commands to be sent between two or more recipients. Recipients may be computers, people, or processes within a computer. Messaging Services are based on specific protocols. A protocol is a set of rules describing, in technical terms, how something should be done. Protocols facilitate transport of the message stream. For example, there is a protocol describing exactly what format should be used for sending specific types of mail messages.

Ex-1105, 65:16-25.

87. Bowman goes on to describe various kinds of messaging formats, such as “Store and Forward Message Services,” “Synchronous Message Services,” and “Asynchronous Message Services.” *Id.*, 65:42-58. Bowman further teaches that message-based middleware could provide services such as “Guaranteed Delivery,” “Queuing,” “Priority Message Deliver,” “Dynamic Routing,” “Multicasting and Broadcasting,” and “Load Balancing.” *Id.*, 66:65-67:11. Bowman further teaches that users have a variety of specific needs and budgetary concerns surrounding their

selection of a messaging middleware, and that the large variety of commercially available middleware should be considered. *Id.*, 66:4-76:44.

88. A POSITA would have recognized that based on the open-ended nature of Bowman with respect to different messaging services, that various known messaging services could be used within the Bowman / Vahalia system. A POSITA would have further recognized the Tuxedo system and method as one such messaging middleware. Tuxedo offers each of these functionalities. It offers “Store and Forward” communications in which a “forwarding agent . . . may read a queue . . . perform a request . . . and store the reply in a reply queue.” Ex-1107, 43. A diagram of this process is below:

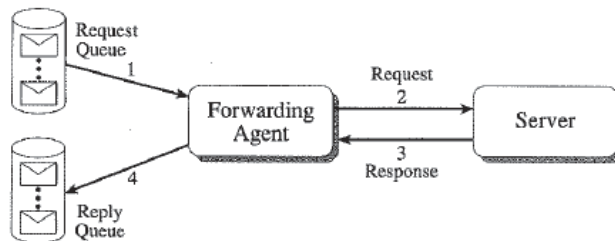


Figure 3.10 Forwarding Agent Converts Queue Operations to Request/Response

Ex-1107 at Fig. 3.10.

89. Tuxedo further states that such a system could be “synchronous to the client,” (*id.*, 31), but that in other instances it could be asynchronous (*id.*, 31-32).

90. A POSITA would have been motivated to incorporate the “Request/Response” or request/reply communication paradigm of Tuxedo in the

Bowman / Vahalia system. Bowman describes using messaging paradigms as part of the MOM functionality of its Filesharing System. Ex-1105, 70:8-12, 14-21, 47-56. In service of that use, Bowman details using Tuxedo's messaging paradigms to satisfy the functionality required within the Filesharing System's "communication agent"—MOM. Ex-1105, 34:18-20, 57:9-18, 95:65-96. Vahalia describes a Data Mover, providing the same service as Bowman's MOM, as part of the functionality of its File Server. Ex-1106, 8:52-58. A POSITA implementing the File Server of Vahalia in the Bowman system, as discussed above, would have been motivated to use the same messaging paradigms, those of Tuxedo, as recommended by Bowman.

91. A POSITA would have been motivated to create the Bowman / Vahalia / Tuxedo combination due to the independent improvement Vahalia and Tuxedo provide to the Bowman system. Accordingly, it would have been obvious to include a known element (Tuxedo's messaging paradigms) into another mechanism (Vahalia's Data Mover or Bowman's middleware) that yield predictable results (using Tuxedo's messaging paradigms with Vahalia's middleware in the Bowman system) for the reasons explained herein. Ex-1105; Ex-1106; Ex-1107.

92. As demonstrated above, Bowman and Vahalia are analogous art to the '244 Patent. And Tuxedo is analogous art to the '244 Patent. Tuxedo is directed to a similar field of endeavor of the '244 Patent— efficient, fault-tolerant client server communication in distributed applications. Ex-1101, Abstract; Ex-1107, 31. Tuxedo

is also reasonably pertinent to a problem described by the '244 Patent—the need for reliable, fault-tolerant communication. Ex-1101, Abstract, 3:65-4:8. Second, Tuxedo is reasonably pertinent to a problem described by the '409 Patent—the need for reliable communication. Ex-1101, Abstract, 3:65-4:8.

93. A POSITA would have expected to succeed in using Tuxedo's paradigms in the Bowman / Vahalia system due to Bowman's explicit suggestion of using the paradigms. Ex-1105, 57:9-18, 34:18-20, 95:15-16. Further, similarities between Tuxedo and Bowman and Vahalia are again immediately apparent just from an examination of their block diagrams. Above, in Tuxedo's Fig. 3.10, one can see the familiar “Computer/Application 1 – Interconnection Fabric – Computer/Application 2” format. A POSITA would have instantly recognized, both from Bowman's explicit teachings and Tuxedo's teachings, that Tuxedo was in the same field of endeavor as Bowman and Vahalia.

IX. CLAIMS 1-10 ARE UNPATENTABLE

A. Ground 1: Claim 1 is Obvious Based on Bowman in view of Vahalia

1. Claim 1

- i. *[1.0] A method of transmitting messages between a first node and a second node, wherein the first and second nodes are each coupled to a fault tolerant storage system (FTSS), the method comprising:*

(A) *A method of transmitting messages between a first node and a second node*

94. Bowman taught “a method of transmitting messages between a *first node* and a *second node*.” Bowman’s Figure 10 depicts a Netcentric Architecture Framework with a client (“*first node*”) and Server (“*second node*”) coupled over Communication Fabric 1010 (*green*):

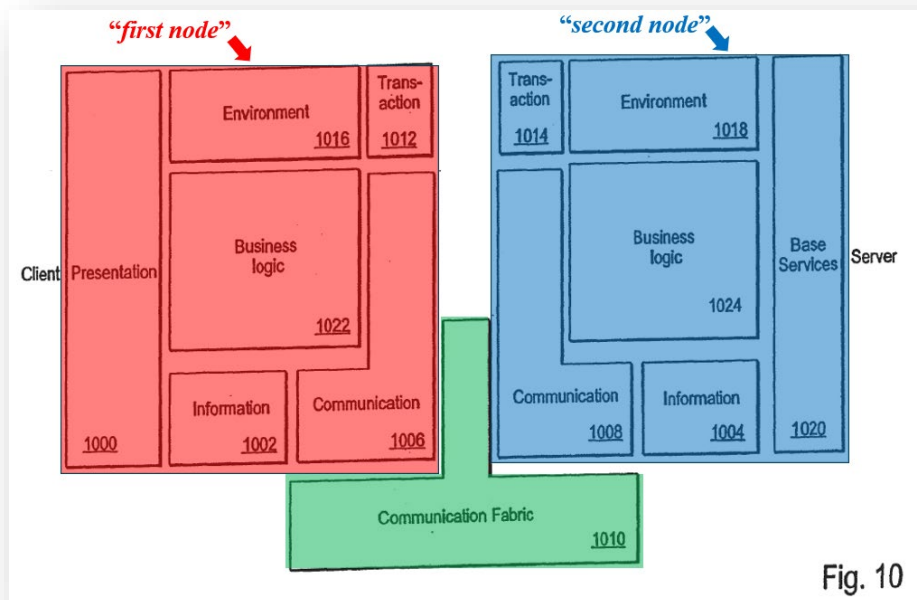


Fig. 10

Ex-1105 at Fig. 10 (annotated).

95. The client is a “*first node*” and the Server is a “*second node*,” consistent with how those terms are used in the ’244 Patent, because the client and server are each computer systems coupled to a network. Ex-1105, 33:5-8 (“Business logic can reside on both client and server. Clients are typically PCs or Workstations with a graphical user interface running in a Web browser. Servers are usually implemented on UNIX, NT or mainframe machines.”), 1:61-64 (“The architecture of the Web follows a conventional client server model. The terms ‘client’ and ‘server’ are used to refer to a computer's general role as a requester of data (the client) or provider of data (the server).”).

96. Bowman also taught “*transmitting messages between a first node and a second node*.” The Communications Fabric 1010 in Figure 10 comprises “communications networks” that facilitate network communications between the client (“*first node*”) and server (“*second node*”). Ex-1105, 83:56-84:3. The Communications Fabric 1010 also includes “Transport Services 2402” to “support the transferring of messages from a process running on one machine to a process running on another machine” (“*a method for transmitting messages*”). Ex-1105, 84:35-44, 84:49-16 (describing message transport). A POSITA would have understood “transferring messages” over the communications fabric would include

transmitting messages between the client (“*first node*”) and server (“*second node*”).

Ex-1105, 84:35-44, 84:49-16.

97. Accordingly, Bowman taught “*a method for transmitting messages*” (e.g., transferring messages over Communications Fabric 1010) “*between a first node*” (e.g., “Server” in Figure 10) and “*a second node*” (e.g., “Client” in Figure 10).

(B) *wherein the first and second nodes are each coupled to a fault tolerant storage system (FTSS)*

98. Bowman would have rendered obvious “*the first and second nodes*” (e.g., Bowman’s server and client) “*are each coupled to a fault tolerant storage system (FTSS)*.” As explained above in Part VI, Petitioner has adopted (for the limited purpose of this Petition) Patent Owner’s position in a parallel proceeding that the scope of “*coupled to*” in this limitation includes “communicatively connected” (e.g., over a conventional computer network). Ex-1110, ¶ 50.

99. Bowman teaches or suggests “*a fault tolerant storage system (FTSS)*.” First, Bowman teaches that the that Netcentric Architecture Framework has fault-tolerant functionality. For example, Bowman explains that Communications Fabric 1010 “provides common network services to the platform-specific network services residing on the client and server nodes.” Ex-1105, 83:66-84:1. Among the common network services that Bowman describes are File Sharing Services 1512. *See, e.g.*, Ex-1105, 58:50-59:50, Fig. 16.

100. Second, Bowman explains that Netcentric Architecture Framework has or can integrate with fault-tolerant hardware. Bowman explains that “File Sharing services allow users to view, manage, read, and write files that may be located on a variety of platforms in a variety of locations.” Ex-1105, 58:50-54. The “File Sharing services can provide the following capabilities,” which includes “Fault tolerance,” such as the “use of primary and replica file systems to ensure high availability of the file system.” Ex-1105, 58:58-69:10.

101. A POSITA would have understood this discussion of primary and replica file systems in Bowman to describe the disk mirroring functionality of RAID 1. Ex-1108, 29 (“**RAID 1 is also known as disk mirroring.** In this implementation, identical copies of each chunk of data are kept on separate disks, or more commonly, **each disk has a twin that contains an exact replica** (or mirror image) of the information. If any disk in the array fails, then the mirrored twin can take over.”); Ex-1114, 144 (defining “disk mirroring” as “[a] technique in which all or part of a hard disk is duplicated onto one or more other hard disks”). A POSITA would have understood RAID (Redundant Arrays of Independent Disks) to describe a disk array arrangement that protects against a single point of failure—e.g., using disk arrays for disk mirroring as provided in RAID provides an increased indication of reliability by providing for redundancy, such that if one drive in the array fails, data is not lost. Ex-1108, 28 (“Disk arrays are groups of disk drives that work together to achieve

higher data-transfer and I/O rates than those provided by single large drives. Arrays can also provide data redundancy so that no data is lost if a single drive (physical disk) in the array should fail. Depending on the RAID level, data is either mirrored or striped.”); Ex-1114, 144; Ex-1112, 464 (defining “RAID” as “a category of disk drives that employ two or more drives in combination for fault tolerance” and noting it “[p]rovides disk mirroring”)

102. This understanding is consistent with the ’244 Patent’s description of disk arrays (e.g., “the SureStore(R) XP512 Disk Array”) as exemplary fault-tolerant storage systems:

A fault tolerant storage systems (FTSS), such as the SureStore(R) XP512 Disk Array, which is a product of Hewlett-Packard Company, guarantees the availability and integrity of files stored by the FTSS. Fault tolerant storage systems are also available from other companies, such as Hitachi Data Systems Corporation and EMC Corporation.

Ex-1101, 4:28-34.

103. A POSITA would have understood that the exemplary SureStore(R) XP512 Disk Array was an enterprise storage solution comprising an array of disk drives, such as a RAID system. Ex-1111, 1, 6.

104. Bowman thus taught or suggested that *the first and second nodes*” (e.g., the server and client of Bowman) “*are each coupled*” (e.g., “communicatively

connected” over Communication Fabric 1010) “to a fault tolerant storage system (FTSS)” (e.g., Bowman’s replica and primary file system of Filesharing Services).

105. Moreover, the combined teachings of Bowman and Vahalia would have rendered this element obvious. For example, it would have been obvious to implement the File Sharing services of Bowman using the systems and methods of Vahalia. Vahalia teaches a data storage system that uses “an array of low-cost disk drives integrated with a large semiconductor cache memory.” Ex-1106, 1:9-15. Vahalia also teaches that its disk array is coupled over a network (e.g., “interfaced to a data network via at least one data mover computer”). Ex-1106, 1:20-21. Annotated Figure 3 of Vahalia shows “Cached Disk Array #2” (red) as part of File Server 60 (yellow):

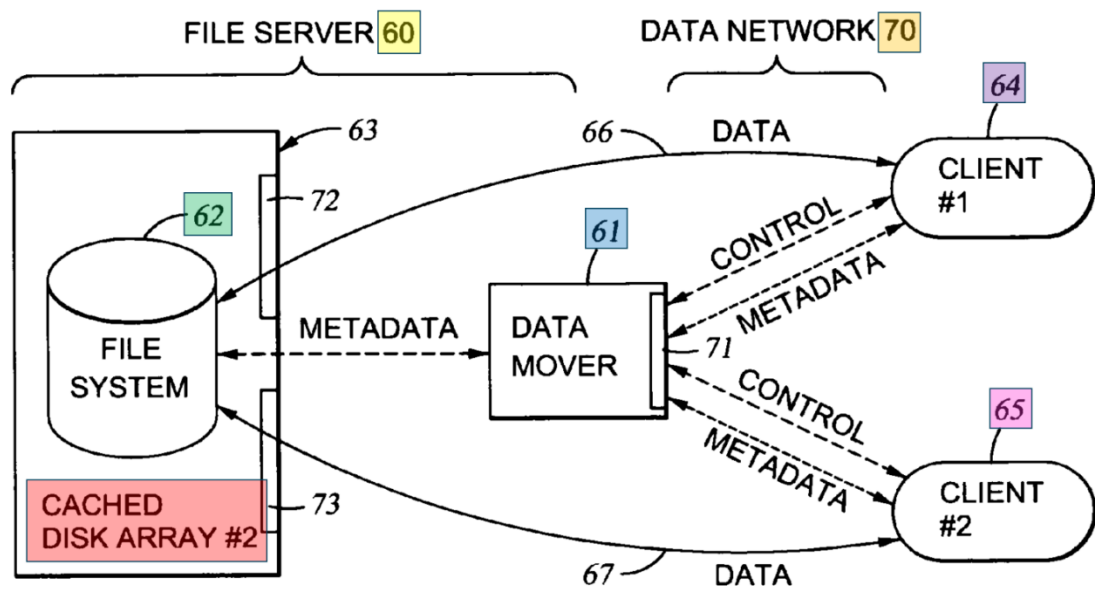


Fig. 3

106. A POSITA would have understood File Server 60 in Vahalia comprising Cached Disk Array #2 is a “*fault-tolerant storage system (FTSS)*”. Ex-1108, 28 (“Arrays can also provide data redundancy so that no data is lost if a single drive (physical disk) in the array should fail.”).

107. A POSITA would have understood that the primary and replica file system underlying Bowman’s File Sharing services could be implemented using an array of network-interfaced disk drives described in Vahalia. Moreover, a POSITA would have understood these disk arrays in Vahalia to be FTSS. This is consistent with the exemplary FTSS from the ‘244 Patent, the SureStore(R) XP512 Disk Array, which likewise provides an array of disks and SCSI and Fibre Channel links. Ex-1111, 7 (describing fibre channel ports), 17 (describing SCSI host connections). These disclosures in Vahalia are also consistent with the ’244 Patent’s explanation that “[t]ypically an FTSS is coupled to a server using a high-speed interface, such as an Ultra160 SCSI interface, a fiber channel interface, or an enterprise system connectivity (ESCON) interface.” Ex-1101, 4:35-49.

108. Vahalia further teaches that client devices (e.g., Client #1, 64 (purple) and Client # 2, 65 (pink)) move data to and from Cached Disk Array #2 over Data Network 70 via a Data Mover 61 (blue) that is part of File Server 60:

In the preferred implementation of the system of FIG. 3, the clients 64, 65 may mount file systems on the cached

disk array 63 by sending NFS commands to the data mover 61. The clients also have a configuration file associating volumes with file systems. The clients move file data directly to the cached disk array 63 using a high-speed data protocol such as is commonly used to read or write data directly to a disk drive over an SCSI or Fibre Channel link.

Ex-1106, 35:47-55.

109. A POSITA would have understood Client #1 and Client #2 to be nodes (e.g., “*a first node*” and “*a second node*”). Ex-1106, 36:11-15 (describing clients consistent with general purpose computers that may include “a data processor,” “random access memory,” and “local disk storage”). Likewise, a POSITA would have understood Client #1 and Client #2 (which are each “connected to” Data Mover 61 on File Server 60 over Data Network 70) to be *each coupled to* File Server 60. Ex-1106, 8:56 (“The data mover 61 has at least one network port 71 connected through the data network 70 to a first client 64 and a second client 65.”); Ex-1110, ¶ 50 (asserting “coupled to” means “communicatively connected”). Moreover, a POSITA would have understood that Vahalia’s Data Network 70 could be implemented over Bowman’s Communications Fabric 1010. Ex-1105, 83:56-84:3 (describing Communications Fabric 1010 as providing for “common network services”); Ex-1106, 8:51-9:17 (describing Data Network 70 as enabling common client-server network communications).

110. Accordingly, if Bowman alone did not suggest or teach this limitation, it would have been obvious based on Bowman in view of Vahalia that “*the first and second nodes*” (e.g., Bowman’s client and server) “*are each coupled to a fault tolerant storage system (FTSS)*” (e.g., are each connected to Vahalia’s File Server 60 over Bowman’s Connection Fabric 1010).

ii. [1.1] transmitting a message from the first node to a communication agent in the FTSS;

111. Bowman taught or suggested “*a communication agent.*” Bowman describes “Message-Oriented Middleware (MOM),” which “refers to the process of distributing data and control throughout the exchange of records known as messages.” Ex-1105, 69:47-56. The “Message-Oriented Middleware products typically support communication among various computing platforms.” Ex-1105, 70:5-12. And “Message-Oriented Middleware is responsible for managing the interface to the underlying communications architecture via the communications protocol APIs and ensuring the delivery of the information to the remote process,” including “[t]ransferring data and control messages during the session.” Ex-1105, 69:53-65.

112. A POSITA would have understood the Message-Oriented Middleware to be a “*communication agent*” because it is software that passes data and control messages on behalf of clients and servers. Ex-1105, 69:53-65; Ex-1112, 139

(defining “middleware” as “[s]oftware that connects two otherwise separate application” and providing “Message Passing” as an exemplary middleware category); Ex-1107, 413 (defining “agent” as “a software module which carries out the instructions of another module”); Ex-1113, 500 (“In the computer programming sense of the word, an agent acts on behalf of another person or thing, with delegated authority.”).

113. This is within the scope of *communication agent* in the ’244 Patent, which is described as software to “facilitate message transfer.” Ex-1101, 5:19-25. The Message-Oriented Middleware in Bowman, like the communication agent in the ’244 Patent, facilitates message transfers, and a POSITA would have accordingly understood the Message-Oriented Middleware to be “*a communication agent*.” Ex-1105, 69:53-65; Ex-1112, 139.

114. Vahalia likewise teaches “*a communication agent*.” Annotated Figure 3 of Vahalia describes File Server 60 (yellow) as including Data Mover 61 (blue), which arbitrates and coordinates data access requests between Client #1 64 (purple) and Client #2 65 (pink):

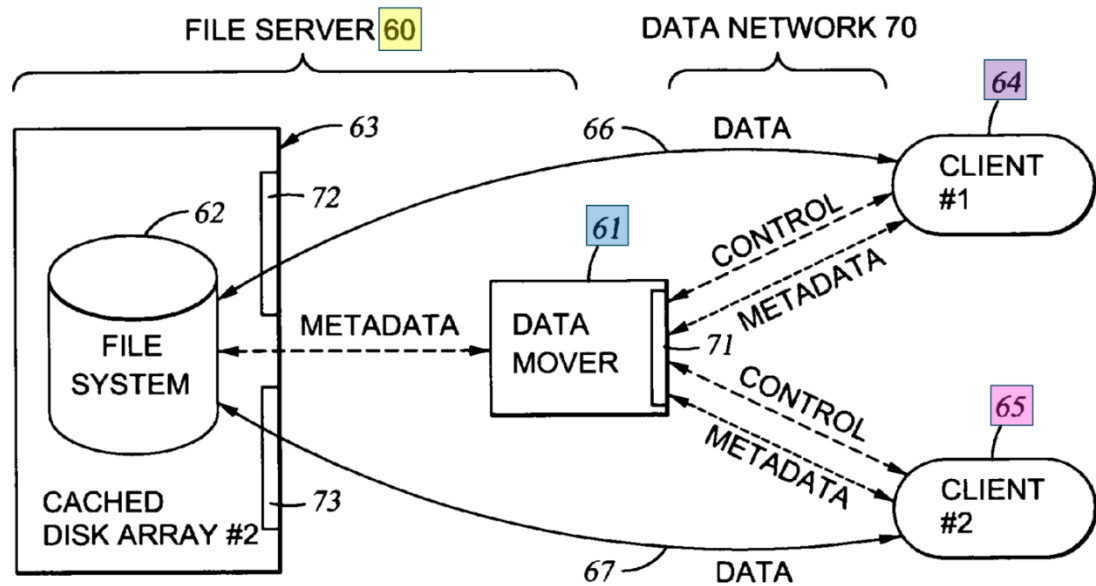


Fig. 3

Ex-1106, 35:25-68 (“As described above with reference to FIG. 3, a file server 60 including a data mover 61 and a cached disk array 63 provides direct data sharing between network clients 64, 65 by arbitrating and coordinating data access requests.”).

115. A POSITA would have understood these data access requests to be sent as network messages from one node (e.g., a client) to another (e.g., a server). Ex-1106, 2:17-24 (“Each of the data movers 21, 22 may receive file access requests from at least one network client.”), 14:32-37 (“Clients may have many outstanding requests at one time. Servers may respond to requests in any order, but a response message must always contain the same Mid value as the corresponding request message.”), 36:52-55 (“The client software modules include a conventional client

NFS module 401 or client CIFS module 402 for transmitting client messages to the data mover (61 in FIG. 3).”). A POSITA would have thus understood Vahalia’s Data Mover to be “*a communication agent*.” Ex-1113, 500; Ex-1107, 413 (defining “agent”).

116. A POSITA would thus have understood Bowman in view of Vahalia to teach “*transmitting a message*” (e.g., sending a file access request as in Vahalia) “*from the first node*” (from a client, e.g., the client in Bowman) “*to a communication agent*” (to Data Mover 61 of Vahalia) “*in the FTSS*” (e.g., the File Server 60 in Vahalia that includes Data Mover 61).

iii. ***[1.2] storing the message in a data structure in highly reliable fault-tolerant storage media of the FTSS;***

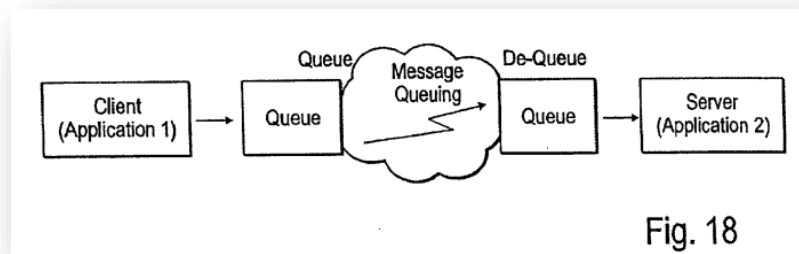
(A) ***storing the message in a data structure***

117. Bowman taught or suggested storing a message in a data structure. Bowman discloses that “Message-Oriented Middleware products typically support communication among various computing platforms (e.g., DOS, Windows, OS/2, Macintosh, UNIX, and mainframes). There are three types of Message-Oriented Middleware commonly implemented: Message Passing[,], **Message Queuing**[,], Publish and Subscribe.” Ex-1105, 70:5-12.

118. Figure 18 of Bowman describes:

Message Queuing (also known as **Store and Forward**)—as depicted in FIG. 18, is an indirect application to application communication model that allows applications to communicate via message queues, rather than by calling each other directly. Message queuing is asynchronous by nature and connectionless, meaning that the recipient need not be directly available when the message is sent. Moreover, it implies support for reliable, guaranteed and assured (non-duplicate) message delivery.

Ex-1105, 70:23-54 (emphasis added).



Id., Fig. 18 (annotated).

119. A POSITA would have understood message queuing as involving storing the message in a data structure (e.g., a queue) in memory. Ex-1107, 41 (“a queue, which is the storage mechanism to hold messages until they are processed”), 49 (“Queues are storage systems through which application components may communicate in a deferred manner.”), 425 (defining “queue” as “a memory data structure to hold messages for processing”).

120. A POSITA would accordingly have understood Bowman to teach or suggest “*storing the message*” (e.g., a file access request) “*in a data structure*” (e.g., in a message queue).

(B) *in highly reliable fault-tolerant storage media of the FTSS*

121. Bowman further taught or suggested storing the message in a data structure “*in highly reliable fault-tolerant storage media of the FTSS.*” Bowman teaches the need for “reliable, guaranteed and assured (non-duplicate) message delivery.” Ex-1105, 70:53-54. It further explains the “use of primary and replica file servers to ensure high availability of file system.” Ex-1105, 59:7-8. For the same reasons described above in Section VIII.D.1, these disclosures would have taught or suggested to a POSITA using a disk array as fault-tolerant storage media for storing the data structures in which messages are stored.

122. As further described in Section VIII.D.1, a POSITA would have understood that one such fault-tolerant storage media would be the cached disk array in File Server 60 from Vahalia. Similarly, Vahalia teaches placing file access requests in a collector buffer or queue 276. *See, e.g.*, Ex-1106, 23:19-24 (“The stream handler routines 275 identify the stream context of each SMB message, place the SMB message in a collector buffer or queue 276, and invoke the high-level routine including a code thread for processing the SMB message in accordance with the

Stream context.”), 23:46-50 (“The stream handler routines 285 identify the stream context of each SMB message, place the SMB message in a collector buffer or queue 286, and invoke the high-level routines 277 including a code thread for processing the SMB message in accordance with the stream context.”), 26:45-50 (“A second approach is to put the requests of the threads into a waiting queue and return with a status Set to be in progress, and when the distributed lock becomes available, all waiting requests are given to the threads from a pre-allocated threads pool inside CFS.”). A POSITA would have understood that placing file access requests (i.e., messages) in a queue (e.g., collector buffer or queue 276) is message queuing.

123. Vahalia discloses that collector or queue 276 (brown) is part of the Data Mover (blue):

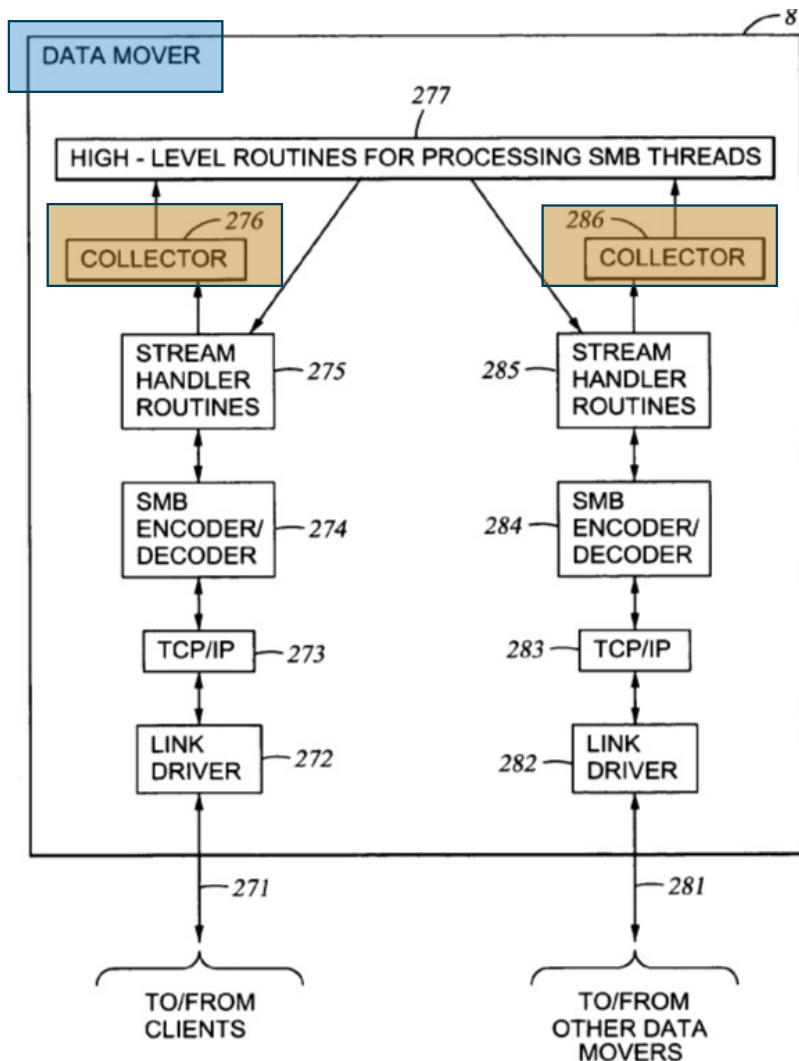


Fig. 16

124. As described in Section IX.A.1.ii, Data Mover is the communication agent of Vahalia's File Server 60, which comprises the FTSS.

125. Accordingly, Bowman in view of Vahalia would have rendered obvious "storing the message" (e.g., Vahalia's file access requests) "in a data structure" (e.g., in Vahalia's collector or queue 276) "in highly reliable fault-tolerant storage media of the FTSS" (e.g., the disk array of Vahalia's File Server 60).

iv. *[1.3] processing the message at the FTSS in accordance with a messaging paradigm; and*

(A) *processing the message ... in accordance with a messaging paradigm*

126. Bowman taught or suggested “*processing the message ... in accordance with a messaging paradigm.*” For instance, Bowman discloses that “[s]hort of interpreting the data being transmitted, the communications fabric is aware of the different message-oriented information streams in order to help the client and server communicate regardless of the different network functions implemented on each platform.” Ex-1105, 83:57-84:34.

127. Likewise, Bowman discloses that its message-oriented middleware (MOM) (i.e., its communication agent) “refers to the process of distributing data and control throughout the exchange of records known as messages. MOM provides the application developer with a set of simple verbs (e.g., connect, send, receive, and disconnect) that are used to exchange information with other distributed applications.” Ex-1105 at 69:47-52. And Bowman discloses the kinds of processing MOM can accomplish, such as:

Translating mnemonic or logical process names to operating system compatible format

Opening a communications session and negotiating parameters for the session

Translating data to the proper format

Transferring data and control messages during the session

Recovering any information if errors occur during transmission

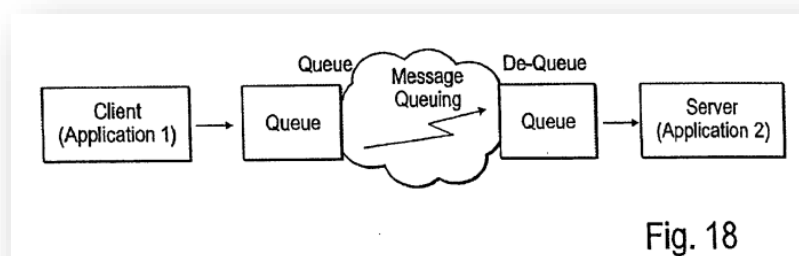
Passing results information and status to the application.

Ex-1105 at 69:59-67.

128. A POSITA would have understood these disclosures, which together describe how messages should be exchanged between applications or processes, to describe “*processing the message ... in accordance with a messaging paradigm.*”

Ex-1105, 69:59-67.

129. Moreover, Bowman discloses that “[m]essage Queuing (also known as Store and Forward)—as depicted in FIG. 18, is an indirect application to application communication model that allows applications to communicate via message queues, rather than by calling each other directly.” Ex-1105, 70:23-29.

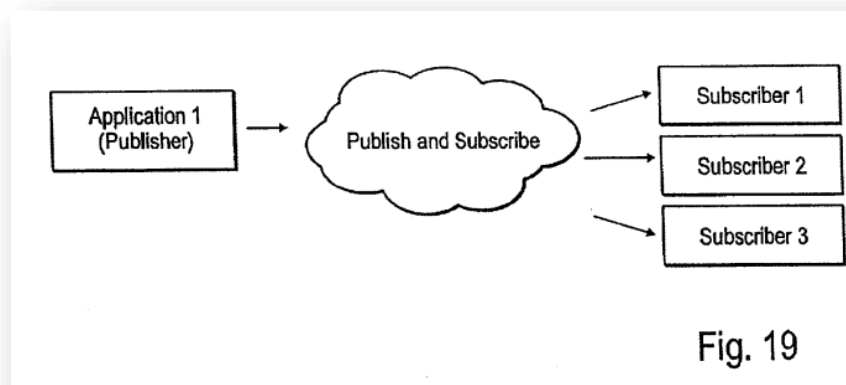


Ex-1105, Fig. 18.

130. A POSITA would have understood that message queuing is a messaging paradigm. Ex-1107, 27 (“A paradigm is a style, or form, of doing something.

Communication paradigms are styles of communication.”), 131 (“Then, a chapter is devoted to each of the four communications paradigms: request/response, conversations, application queuing, and events.”). This is also consistent with the ‘244 Patent, which describes queue-based communication as “*a messaging paradigm*” that was known in the prior art. Ex-1101, Fig. 3 (describing prior art).

131. Bowman further discloses in Figure 19 that “Publish and Subscribe (also known as Push messaging)—as shown in FIG. 19, is a special type of data delivery mechanism that allows processes to register an interest in (i.e., subscribe to) certain messages or events. An application then sends (publishes) a message.” *Id.* at 70:31-36.



Ex-1105, Fig. 19.

132. Publish and subscribe is also “*a messaging paradigm*.” Ex-1107, 27 (“A paradigm is a style, or form, of doing something. Communication paradigms are styles of communication.”), 131 (“Then, a chapter is devoted to each of the four

communications paradigms: request/response, conversations, application queuing, and events.”). This is also consistent with the ’244 Patent, which describes publish-and-subscribe communication as a paradigm known in the art. Ex-1101, 2:46-49 (“Because client and server processes that use the event broker communicate with one another based on a set of Subscriptions, this paradigm is also known in the art as publish-and-Subscribe communication.”).

133. A POSITA would thus have understood Bowman to teach or suggest “*processing a message*” (e.g., Bowman’s communications agent, MOM, delivering or storing incoming or outgoing messages) “*in accordance with a messaging paradigm*” (e.g., by applying message queueing or publish and subscribe methodologies).

(B) *at the FTSS*

134. Bowman in view of Vahalia would have rendered obvious processing the message in accordance with a messaging paradigm “*at the FTSS*.” As explained in Section IX.A.1.ii, it would have been obvious to use Vahalia’s communication agent (Data Mover) to implement the middleware communication agent described in Bowman (MOM). Further, as described in Section IX.A.1.iii, Vahalia teaches that its communication agent (Data Mover) is implemented on File Server 60, which comprises the FTSS. It would accordingly have been obvious based on Bowman in view of Vahalia to implement processing the message in accordance with a message

paradigm (which takes place at MOM in Bowman) “*at the FTSS*” (i.e., at the Data Mover on File Server 60 in Vahalia).

v. *[1.4] transmitting the message from the FTSS to the second node.*

(A) *transmitting the message ... to the second node*

135. Bowman taught “Message-Oriented Middleware (MOM),” which “refers to the process of distributing data and control **throughout the exchange of records known as messages.**” Ex-1105, 69:47-56 (emphasis added). Bowman explains that “Message-Oriented Middleware products typically support communication **among various computing platforms.**” *Id.*, 70:5-12. It further discloses that:

Clients and servers are rarely directly connected to one another, but separated by a **network of routers, servers and firewalls** providing an ever increasing number of network services such as address resolution, **message routing**, security screening and many more.

The communications fabric provides common network services to the platform-specific network services residing on the **client and server nodes.**

Id., 83:57-84:9.

136. As explained for claim limitation 1.1, MOM is a communication agent.
See Section IX.A.1.ii.

137. Bowman illustrates exemplary message routing using message queuing, as in Figure 19 below:

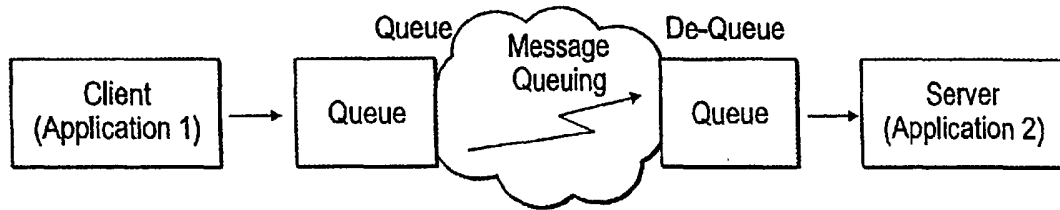


Fig. 18

Ex-1105, Fig. 19.

138. A POSITA would have understood message queuing to involve receiving a message at MOM (i.e., the communication agent) from a first node (e.g., from client (Application 1)), queueing the message, later de-queueing the message, and then “*transmitting the message to the [second node](#)*” (e.g., Server (Application 2)). Ex-1105, 70:22-29 (“Message Queuing (also known as Store and Forward)-as depicted in FIG. 18, is an indirect application to application communication model that allows applications to communicate via message queues, rather than by calling each other directly. Message queuing is asynchronous by nature and connectionless, meaning that the recipient need not be directly available when the message is sent. Moreover, it implies support for reliable, guaranteed and assured (nonduplicate) message delivery.”).

139. Bowman teaches a similar transmission by example through the push-and-subscribe messaging paradigm, as shown in Figure 19:



Fig. 19

Ex-1105, Fig. 19.

140. A POSITA would have understood push and subscribe to involve receiving a message from a first node (e.g., Application1 (Publisher)) at MOM (i.e., the communication agent) and “*then transmitting the message to the [second node](#)*” (e.g., Subscriber 1) if that node as subscribed to receive such messages. Ex-1105, 70:31-36 (“Publish and Subscribe (also known as Push messaging) as shown in FIG. 19, is a special type of data delivery mechanism that allows processes to register an interest in (i.e., subscribe to) certain messages or events. Application then sends (publishes) a message, which is then forwarded to all processes that subscribe to it.”).

141. Bowman thus teaches “*transmitting the message*” (e.g., a file access request) “*to the [second node](#)*” (e.g., the Server from Bowman Figure 10).

(B) *from the FTSS*

142. Bowman in view of Vahalia renders obvious transmitting the message “*from the FTSS*” to the “*second node*.” As explained in claim limitation 1.1 (see Section IX.A.1.ii), it would have been obvious to use Vahalia’s communication agent (Data Mover) to implement the middleware communication agent described in Bowman (MOM), which as described above Bowman teaches as transmitting the messages. Further, as described in Section IX.A.1.iii, Vahalia teaches that its communication agent (Data Mover) is implemented on File Server 60, which comprises the FTSS. It would accordingly have been obvious based on Bowman in view of Vahalia to implement “*transmitting the message*” (which takes place at MOM in Bowman) “*from the FTSS*” (i.e., at the Data Mover on File Server 60 in Vahalia).

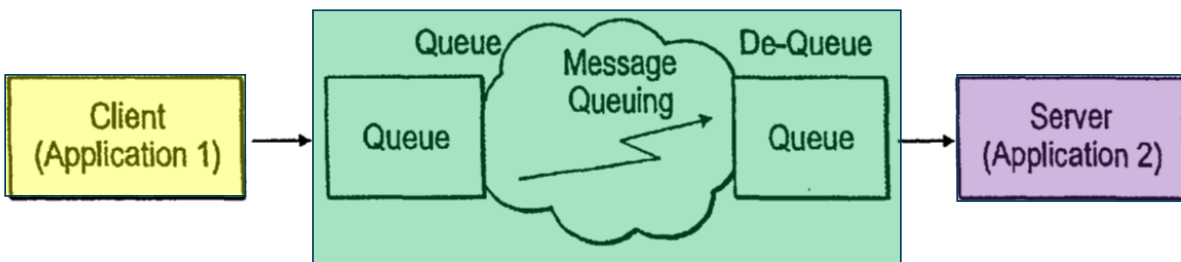


Fig. 18

143. As demonstrated in limitations 1.2 and 1.3 (see Sections IX.A.1.iiiI and X.A.1.iv), this would entail storing and processing messages from the client (yellow) at the FTSS (green). The message would then be transmitted from the FTSS to the

Server (purple), which a POSITA would have understood as “*transmitting the message from the FTSS to the second node.*” Ex-1105, 70:22-29.

144. Consequently, the combined teachings of Bowman and Vahalia would have rendered obvious every limitation in claim 1.

B. Ground 2: Claims 2–10 are Obvious Based on Bowman in view of Vahalia and in Further View of Tuxedo

1. Claim 2

- i. *[2.0] The method of claim 1 wherein the messaging paradigm is a request/reply communication paradigm, and:*

145. As described above (*see* Section VIII.A.1), Bowman in view of Vahalia renders obvious the method of claim 1.

146. Bowman in view of Vahalia and in further view of Tuxedo further renders obvious “*wherein the messaging paradigm is a request/reply communication paradigm.*” Bowman teaches the use of Message-Oriented Middleware (MOM) for “architecture via the communications protocol APIs and ensuring the delivery of the information to the remote process.” Ex-1105, 70:47-56. Bowman further discloses that such middleware may comprise various implementations, such as Message Passing, Message Queuing, and Publish and Subscribe:

There are three types of Message-Oriented Middleware commonly implemented:

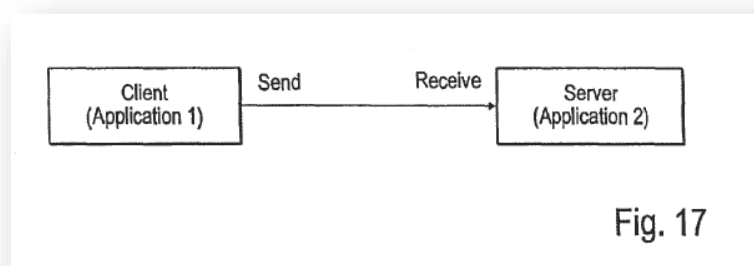
Message Passing

Message Queuing

Publish and Subscribe

Ex-1105, 70:8-12.

147. Bowman also taught or suggested a messaging paradigm consisting of “a request/reply communication paradigm.” Bowman discloses “Message Passing-as illustrated in FIG. 17, is a direct, application-to-application communication model.” Ex-1105, 70:14-15. Bowman further discloses that “[a]n application request is sent in the form of message from one application to another. The communication method can be either synchronous like RPCs or asynchronous (through callback routines). **In a message-passing model, a direct link between two applications that participate in the message exchange is always maintained.**” *Id.*, 70:15-21 (emphasis added).



Ex-1105, Fig. 17.

148. A POSITA would have understood that these forms of messaging in Bowman are messaging paradigms—also referred to as “communication

paradigms.” Ex-1107, 27 (“A paradigm is a style, or form, of doing something. Communication paradigms are styles of communication.”), 131 (“Then, a chapter is devoted to each of the four communications paradigms: request/response, conversations, application queuing, and events.”). Tuxedo teaches implementing such messaging paradigms (including message queuing and publish and subscribe) at the time of the ’244 Patent. Ex-1107, 190 (“The TUXEDO System provides a wide range of communication paradigms and control over distributed transactions in a small, well-designed interface.”), 147 (“The TUXEDO System allows programs to communicate with each other using different communications paradigms.”).

149. Bowman explicitly suggests using Tuxedo as middleware for implementing messaging paradigms (e.g., message queueing):

Communications middleware can provide additional 10 communications services that may be required by the applications. Additional services include dynamic message routing, guaranteed delivery, broadcasting, queuing, and priority delivery. These common services are usually provided in the communications middleware rather than addressing them in each application separately. Different communications middleware products provide different services. Additionally, many middleware packages, **such as Tuxedo**, provide OLTP functionality.

Ex-1105, 57:9-18, 34:18-20 (“The use of a transaction manager such as Tuxedo allows for service location independence.”), 95:15-16 (“TUXEDO has been in the TP marketplace for seven years and has the most installations of all TP monitors.”).

150. A POSITA would have understood to implement the message paradigm teachings from Tuxedo on File Server 60 of Vahalia, including based on the suggestions in Bowman that Tuxedo is well-suited for distributed transaction processing (such as the client/server model in Bowman and Vahalia) and spanning multiple hardware platforms (such as between the client and server computers in Bowman and Vahalia and File Server 60 in Vahalia that comprises the FTSS). Ex-1105, 95:65-96 (“BEA TUXEDO-provides a robust middleware engine for developing and deploying business-critical client/server applications. BEA TUXEDO handles not only distributed transaction processing, but also application and the full complement of services necessary to build and run enterprise-wide applications. It enables developers to create applications that span multiple hardware platforms, databases and operating systems.”).

151. Among the messaging paradigms that Tuxedo teaches is a request/response paradigm. Ex-1107, xxxiv (“Release 1.0 of the TUXEDO System, available in 1984 and depicted in Figure P.1, provided client-server facilities within a single random access memory computer-you can think of it as "Client Server in a Box," as shown in Figure P.1. Release 1.0 allowed an application to be partitioned

into a set of cooperating client and server processes and provided a rudimentary Application Programming Interface (API) that supported request/response interactions among them.”), 189 (“The request/response communication paradigm has been a fundamental part of the TUXEDO System since its inception.”).

152. A POSITA would have known that the Request/Response paradigm is one of the most basic communication paradigms in client-server arrangements (such as those described in Bowman and Vahalia). Ex-1107, 28 (“One of the simplest interactions between computer programs is the request/response paradigm. In this style of programming, one program, called the requestor, asks another program, the requestee to do something. The request can be to provide some information or take some action and report on its success or failure. The requestee performs the action and usually returns a result. In the communication jargon, a requestor is often called a client, and the requestee is called a server.”).

153. A POSITA would have understood the Request/Response paradigm in Tuxedo to be “*a request/reply communication paradigm*.” Ex-1107, 30 (“The important elements of request/response communication include the following: ... The communication protocol is a simple ‘one-over(request)/oneback(reply)’ protocol.”), 35 (“The protocol for request/response is very explicit: The client asks, the server replies.”), 425 (defining “request/response” as “a communication paradigm in which the client sends a request and the server

responds with a reply”). A POSITA would have thus found it obvious based on Bowman in view of Vahalia and in further view of Tuxedo that “*the messaging paradigm is a request/reply communication paradigm.*”

154. Consequently, Bowman, Vahalia, and Tuxedo teach this limitation.

ii. [2.1] transmitting a message from the first node to a communication agent in the FTSS comprises:

155. For the same reasons described above regarding claim limitation 1.1 (see Section IX.A.1.ii), the combination of Bowman and Vahalia teaches this claim limitation.

iii. [2.2] transmitting a request message from the first node to a request/reply communication agent in the FTSS;

156. For the same reasons described above regarding claim limitation 1.1 (see Section IX.A.1.ii), the combination of Bowman and Vahalia teaches “*transmitting a request message from the first node to ... a communication agent in the FTSS.*” Further, for the same reasons described above regarding claim limitation 2.0 (see Section IX.B.1.i), it would have been obvious based on Bowman in view of Vahalia and in further view of Tuxedo that the “*communication agent in the FTSS*” (e.g., Data Mover from Vahalia) would use the “*request/reply communication paradigm*” described in Bowman and Tuxedo.

157. A POSITA would have understood that when the “*communication agent*” (e.g., Data Mover from Vahalia) implementing a “*request/reply*

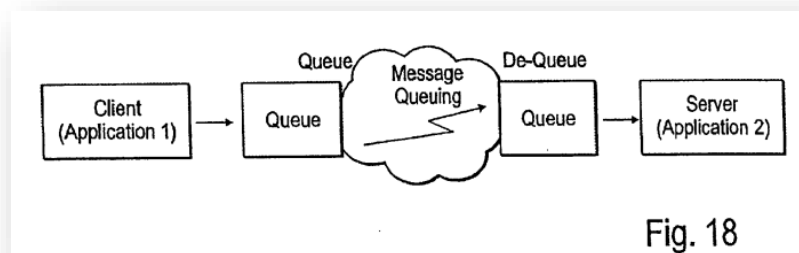
communication paradigm” (e.g., the request/response communication paradigm from Tuxedo), the communication agent (e.g., Data Mover) is “a *request/reply communication agent*.” Ex-1107, 413 (defining “agent” as “a software module which carries out the instructions of another module”).

iv. ***[2.3] storing the message in a data structure in highly reliable fault-tolerant storage media of the FTSS comprises:***

158. Bowman and Vahalia teach or render obvious “*storing the message in a data structure in highly reliable fault-tolerant storage media of the FTSS comprises*” for the same reasons described for claim limitation 1.2 (see Section IX.A.1.iii).

v. ***[2.4] storing the request message in a request queue in highly reliable fault-tolerant storage media of the FTSS;***

159. Bowman in view of Tuxedo teaches “*storing the request message in a request queue.*” Bowman teaches storing messages in a “*queue*,” as shown in Figure 18:



Ex-1105, Fig. 18.

160. Bowman discloses that Figure 18 shows “an indirect application to application communication model that allows applications to communicate via message queues, rather than by calling each other directly.” *Id.*, 70:22-26.

161. Tuxedo further discloses the use of a message queue. For instance, Tuxedo also describes the concept of indirect communication:

Sometimes it is not possible, required, or desirable, for application programs to communicate on-line. Rather, the communications between the modules are deferred over time. An analogy from everyday experience occurs when you leave a message on a telephone answering machine.

To effect deferred communication between modules, it is necessary for them to "leave messages" for each other. If the communication is to take place across a reboot of the computer, it will be necessary to leave the messages on persistent storage. The persistent storage on an answering machine is the magnetic tape or disk inside of the device.

Ex-1107, 40.

162. To effectuate these deferred (or indirect) communications, Tuxedo discloses “a *queue*, which is the storage mechanism to hold messages until they are processed.” Ex-1107, 41. Tuxedo also explains that these queues can be used with the request/response paradigm:

For example, a forwarding agent, as shown in Figure 3.10, may read a queue (flow 1), perform a request/response on-line (flows 2 and 3), and store the reply in a reply queue (flow 4). The server thinks it is talking on-line to its client. But it is really talking to a surrogate client, that is, the forwarding agent. The real client, or its successor, will get the server's response later in a response queue.

Ex-1107, 42-43.

163. As Figure 3.10 from Tuxedo shows, a request/reply paradigm may further include both a Request Queue (red) and a Reply Queue (green):

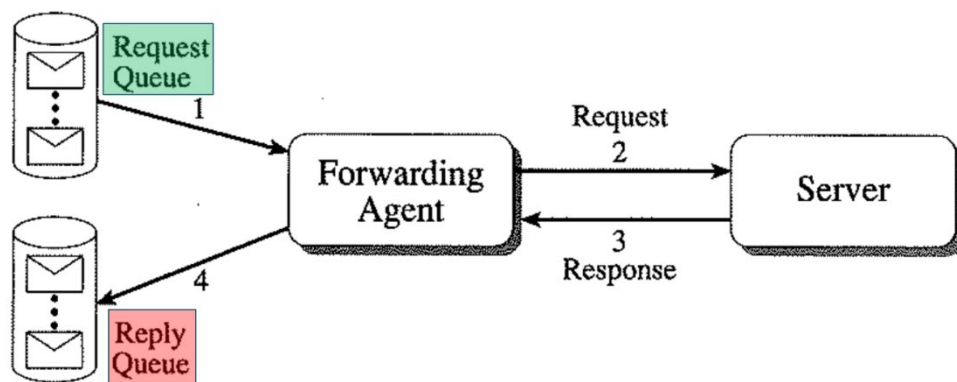


Figure 3.10 Forwarding Agent Converts Queue Operations to Request/Response

Ex-1107, 43.

164. A POSITA would have understood Figure 3.10 to show “*storing the request message*” (e.g., Request that is later retrieved by Forwarding Agent and sent to Server) “*in a request queue*” (e.g., the Request Queue from which the Request is

retrieved by Forwarding Agent). Ex-1107, 42-43, 425 (defining “request queue” as “the memory data structure that holds incoming requests for services”).

165. A POSITA would have understood Tuxedo’s Request Queue as a data structure on persistent storage (e.g., a disk drive). Ex-1107, 41 (“a queue, which is the storage mechanism to hold messages until they are processed”), 49 (“Previously, we spoke of the queuing paradigm for deferred communication. Queues are storage systems through which application components may communicate in a deferred manner.”), 75 (“In their simplest form, application queues allow programs to put messages into disk-based queues so that they can be dequeued in a first-in, first-out (FIFO) order.”), 425 (defining “request queue” as “the memory data structure that holds incoming requests for services”).

166. For at least the same reasons described above for claim 1.2 (*see* Section IX.A.1.iii), a POSITA would have understood Bowman in view of Vahalia to render obvious storing queues, such as the Request Queue, at the FTSS. Accordingly, Bowman in view of Vahalia in further view of Tuxedo would have rendered obvious “*storing the request message*” (e.g., the requests in Bowman or Tuxedo, or the file access request of Vahalia) “*in a request queue*” (e.g., in Tuxedo’s Request Queue) “*in highly reliable fault-tolerant storage media of the FTSS*” (e.g., in the disk array on Vahalia’s File Server 60).

vi. ***[2.5] processing the message at the FTSS in accordance with a messaging paradigm comprises:***

167. The combination of Bowman and Vahalia discloses “processing the message at the FTSS in accordance with a messaging paradigm comprises” for at least the same reasons described in Section IX.A.1.iv.

vii. ***[2.6] retrieving the request message from the request queue; and***

168. Tuxedo teaches “retrieving the request message from the request queue.” Tuxedo explains that “a forwarding agent, as shown in Figure 3.10, may read a queue (flow 1), perform a request/response on-line (flows 2 and 3), and store the reply in a reply queue (flow 4).” Ex-1107, 42-43. This is shown in Figure 3.10, where Forwarding Agent (purple) reads Request Queue (green) at flow 1 (yellow):

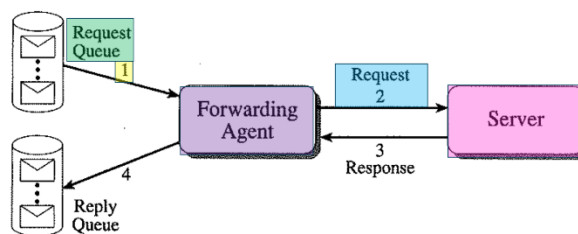


Figure 3.10 Forwarding Agent Converts Queue Operations to Request/Response

Ex-1107 at Fig. 3.10.

169. A POSITA would have understood flow 1 (Forwarding Agent reading from the Request Queue) to involve retrieving from the Request Queue the Request that Forwarding Agent the transmits to Server (pink). Ex-1107, 42-43.

170. Consequently, Bowman in view of Vahalia in further view of Tuxedo renders obvious “*retrieving the request message*” (e.g., the requests of Bowman or Tuxedo or the file-access request of Vahalia) “*from the request queue*” (e.g., from the Request Queue of Tuxedo).

viii. [2.7] transmitting the message from the FTSS to the second node comprises:

171. For the same reasons described above regarding claim limitation 1.1 (see Section IX.A.1.ii), the combination of Bowman and Vahalia teach this claim limitation.

ix. [2.8] transmitting the request message from the FTSS to a service on the second node;

172. For the same reasons described above regarding claim limitation 1.1 (see Section IX.A.1.ii), the combination of Bowman and Vahalia teaches “*transmitting the request message from the FTSS to ... the second node.*”

173. Bowman teaches that “[i]n a client-server environment, a client makes requests of services on a Server.” Ex-1105, 213:20-21. It further teaches that “[i]n a typical two or three-tiered client-server application, the services are maintained away from the users (Client) on separate Server machines. Whenever a user needs to use a service, the user must send a request across the network to the Server machine.” *Id.*, 213:23-27. A POSITA would have understood this to mean that a first node (“Client”) is “*transmitting the request message to a service*” (“send[ing] the

request across the network” to “services on a Server”) “*on the second node*” (e.g., “the Server machine” on which the “services” reside). Ex-1105, 213:20-27.

174. Tuxedo also taught or suggested that the request message is transmitted “*to a service on the second node.*” Tuxedo teaches, for instance, that clients request services from servers:

One of the simplest interactions between computer programs is the request/response paradigm. In this style of programming, one program, called the requestor, asks another program, the requestee, to do something. The request can be to provide some information or take some action and report on its success or failure. The requestee performs the action and usually returns a result. In the communication jargon, a requestor is often called a client, and the requestee is called a server. The server carries out the wishes of its clients by performing a **service** for them.

Ex-1107, 28, 24 (“As we will explain in more detail in Chapter 6, the TUXEDO System is fundamentally a Service Request Broker (SRB) and performs many functions analogous to an ORB. In the TUXEDO System, service calls correspond to method invocations.”).

175. This is further illustrated in Tuxedo’s Figure 3.2 below, which shows client (yellow) transmits Request Data (green) to server (purple):

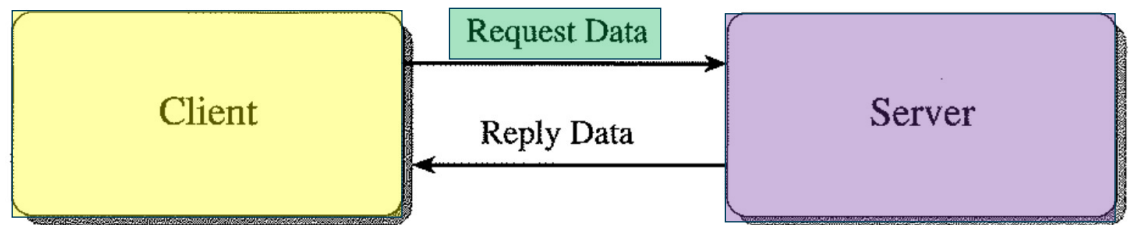


Figure 3.2 Request/Response Interaction

176. A POSITA would have understood from Tuxedo that Server in Figure 3.2 is hosting a “service” that is receiving a “request” (Request Data) from client. Ex-1107, 24 (“the TUXEDO System is fundamentally a Service Request Broker (SRB)”), 28 (“The server carries out the wishes of its clients by performing a service for them.”). A POSITA would have understood Tuxedo to teach “transmitting” (sending by client) “the request message” (Request Data) “to a service on the second node” (the service hosted on server that is performing the service responsive to Request Data). Ex-1107, 24 (“the TUXEDO System is fundamentally a Service Request Broker (SRB)”), 28 (“The server carries out the wishes of its clients by performing a service for them.”).

177. Tuxedo teaches an exemplary implementation where “a forwarding agent, as shown in Figure 3.10, may read a queue (flow 1), perform a request/response on-line (flows 2 and 3), and store the reply in a reply queue (flow 4).” Ex-1107, 42-43. More specifically, Forwarding Agent (purple) reads Request Queue (green) at flow 1 (yellow):

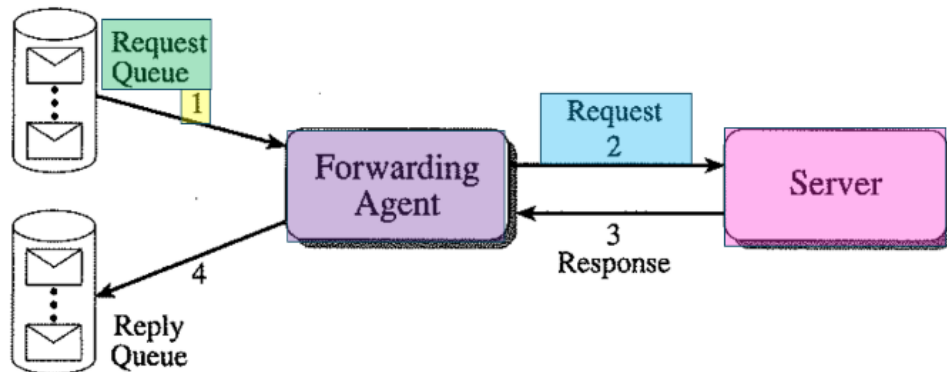


Figure 3.10 Forwarding Agent Converts Queue Operations to Request/Response

Ex-1107, 43.

178. For the same reasons explained for claim limitation 2.4 (*see* Section IX.B.1.v), a POSITA would have understood Bowman in view of Vahalia in further view of Tuxedo to suggest implementing Request Queue on the FTSS (e.g., on the cached disk array of File Server 60).

179. Given Bowman's teachings that middleware (e.g., Bowman's MOM) should mediate messaging (e.g., as described for claim limitation 1.1, *see* Section IX.A.1.ii), and Vahalia's teachings that such middleware (e.g., Vahalia's Data Mover) should be implemented on File Server 60 that comprises a cached disk array (i.e., the FTSS, as described for claim limitation 1.1, *see* Section IX.A.1.ii), a POSITA would have understood Bowman in view of Vahalia to suggest implementing Tuxedo's Forwarding Agent that manages the queues as part of the

middleware on Vahalia's File Server 60 (e.g., as Vahalia's Data Mover). Ex-1106, 8:52-58 (“In this example, a file server 60 includes a data mover 61 and data storage such as a file system 62 in a cached disk array 63. The data mover 61 owns the file System 62, and the data mover 61 exchanges metadata with the file system 62. The data mover 61 has at least one network port 71 connected through the data network 70 to a first client 64 and a second client 65.”); Ex-1107, 79 (“One common use of application queues is for storing such requests and having TUXEDO's forwarding agent dequeue and forward them to service routines.”).

180. Given the foregoing, a POSITA would have understood Bowman in view of Vahalia in further view of Tuxedo to teach “*transmitting the request message*” (e.g., Request from Tuxedo's Figure 3.10) “*from the FTSS*” (e.g., from Tuxedo's Forwarding Agent as implemented on Vahalia's File Server 60 comprising the cached disk array) “*to a service on the second node*” (e.g., to the service on Server from Tuxedo's Figure 3.10). Ex-1107, 43; Ex-1105, 213:20-27.

- x. ***[2.9a] and the method further comprises: executing the service on the second node using the request message as an input and providing a reply message as an output;***

181. The combination of Bowman and Vahalia disclose this limitation for the same reasons disclosed in Section IX.B.1.ix.

182. For the same reasons described for claim 2.8 (*see* Section IX.B.1.ix), Bowman in view of Vahalia in further view of Tuxedo teaches “*the service*” (e.g.,

the service receiving the Request from Tuxedo's Sever in Figure 3.10) “on the second node” (e.g., on Tuxedo's Server in Figure 3.10). A POSITA would have further understood Bowman in view of Vahalia in further view of Tuxedo to teach “executing the service on the second node.” Ex-1105, 248:65-249:2 (“Imagine a typical client/server system design. In almost all cases, a typical system executes data access logic on the server and presentation logic on the client, business logic is split across both the client and server. The majority of this business logic is maintained on the server.”).

183. Figure 3.10 from Tuxedo also shows Server (pink) receiving Request at flow 2 (blue) from Forwarding Agent (purple) and Server replying to Forwarding Agent with Response at flow 3 (red):

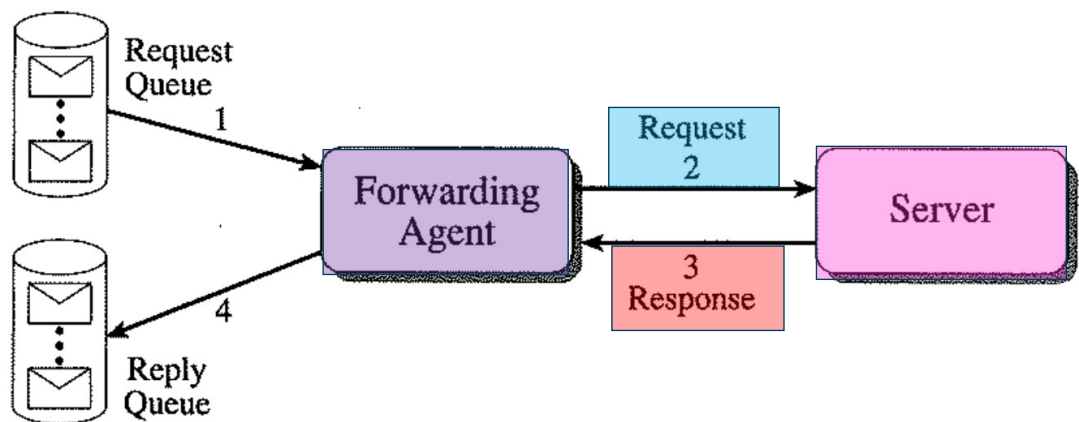


Figure 3.10 Forwarding Agent Converts Queue Operations to Request/Response

Ex-1107, 43.

184. A POSITA would have understood this to teach “*using the request message as an input*” (e.g., the Server using the incoming Request to formulate a Response) “*and providing a reply message as an output*” (e.g., sending the Response from Server in reply to the Request). Ex-1107, 42 (“In effect, a queued message is a request for some future processing. The queue serves as a place to hold the input to these requests. ... If the processing is to return some results, it can do so through another queue. The originator of the request, if it still exists, can access the response through this response (or reply) queue.”), 42-43 (“For example, a forwarding agent, as shown in Figure 3.10, may read a queue (flow 1), perform a request/response on-line (flows 2 and 3), and store the reply in a reply queue (flow 4). The server thinks it is talking on-line to its client. But it is really talking to a surrogate client, that is, the forwarding agent. The real client, or its successor, will get the server's response later in a response queue.”); Ex-1114, 404 (defining “service” as “[i]n networking, specialized, software-based functionality provided by network servers”).

185. Consequently, Bowman in view of Vahalia in further view of Tuxedo teaches this limitation.

xi. [2.9b] transmitting the reply message from the second node to the request/reply communication agent in the FTSS;

186. For the reasons discussed for claim limitations 1.1, 2.8, and 2.9a (see Sections IX.A.1.ii, IX.B.1.ix, and IX.B.1.x), Bowman in view of Vahalia in further view of Tuxedo discloses “transmitting the reply message” (e.g., the Response in Tuxedo’s Figure 3.10) “from the second node” (e.g., the Server in Tuxedo’s Figure 3.10) “to the request/reply communication agent in the FTSS” (e.g., to Vahalia’s Data Mover in File Server 60 comprising the Forwarding Agent from Tuxedo’s Figure 3.10).

xii. [2.9c] storing the reply message in a reply queue in highly reliable fault-tolerant storage media of the FTSS;

187. For the reasons discussed for claim limitations 2.4 and 2.9a (see Sections IX.B.1.v and IX.B.1.x), Bowman in view of Vahalia in further view of Tuxedo teaches “storing the reply message in a reply queue” (e.g., storing the Response in the Reply Queue as shown in Tuxedo’s Figure 3.10) “in highly reliable fault-tolerant storage media of the FTSS” (e.g., in Vahalia’s File Server 60 comprising the cached disk array). Ex-1107, 42, 425 (defining “reply queue” as “the memory data structure that holds responses to requests”).

xiii. [2.9d] retrieving the reply message from the reply queue; and

188. Tuxedo's Figure 3.10 shows that Response (red) from Server (pink) is ultimately stored by Forwarding Agent (purple) at Reply Queue:

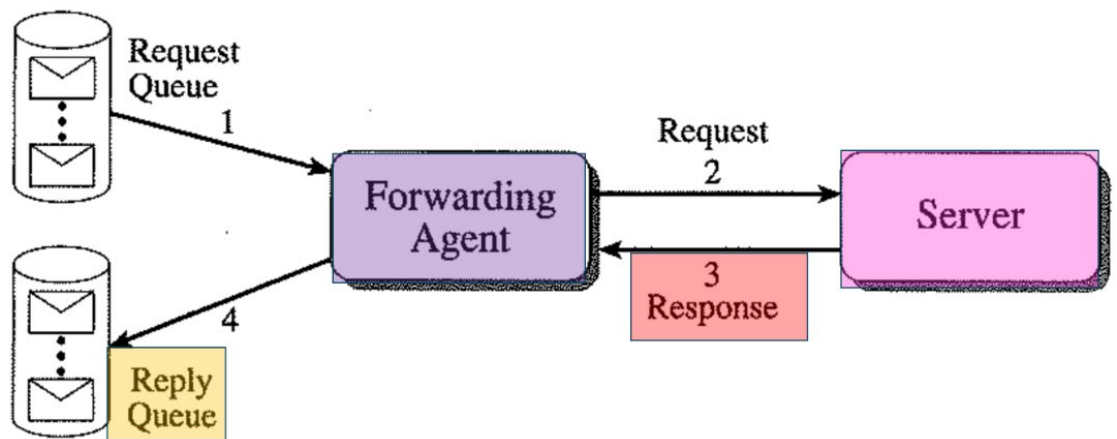


Figure 3.10 Forwarding Agent Converts Queue Operations to Request/Response

Ex-1107, 43.

189. Tuxedo further teaches:

In effect, a queued message is a request for some future processing. The queue serves as a place to hold the input to these requests. However, the enqueuer cannot be assured when the processing will take place. ... If the processing is to return some results, it can do so through another queue. The originator of the request, if it still exists, can access the response through this response (or reply) queue.

Ex-1107, 42.

190. A POSITA would have understood this disclosure in Tuxedo that the “originator of the request ... can access the response through this response (or reply) queue” to teach “*retrieving the reply message*” (i.e., “access[ing] the response”) “*from the reply queue*” (i.e., from the “response (or reply) queue”). Ex-1107, 111 (“The TUXEDO run-time system prepares the reply message for the client. Then, it marshals the message (if the client machine has a different byte ordering) and sends the message to the client. Finally, the TUXEDO run-time system on the client retrieves the reply message, unmarshals it (if necessary), and delivers the typed buffer to the application.”), 425 (defining “reply queue” as “the memory data structure that holds responses to requests”).

191. Consequently, Bowman in view of Vahalia in further view of Tuxedo teaches this limitation.

xiv. [2.9e] *transmitting the reply message from the FTSS to the first node.*

192. For the same reasons described above regarding claim limitations 2.9b and 2.9d (*see* Sections IX.B.1.xi and IX.B.1.xiii), Bowman in view of Vahalia in further view of Tuxedo teaches “*transmitting the reply message*” (e.g., retrieving the Response from Tuxedo’s reply queue in Figure 3.10) “*from the FTSS*” (e.g., from Vahalia’s File Server 60 comprising the cached disk array) “*to the first node*” (to the original requestor of Tuxedo, e.g., the client from Bowman and/or Vahalia).

2. Claim 3

- i. ***[3.0] The method of claim 1 wherein the messaging paradigm is a conversational communication paradigm, and:***

193. As described above (*see* Section VIII.A.1), Bowman and Vahalia disclose the method of claim 1. Further, as described above for claim 2.0 (*see* Section IX.B.1.i), Bowman in view of Vahalia in further view of Tuxedo “*teaches the messaging paradigm is a ... communication paradigm.*”

194. Tuxedo also teaches “*a conversational communication paradigm.*” Tuxedo’s Figure 3.6 shows a Conversational Programming Paradigm:

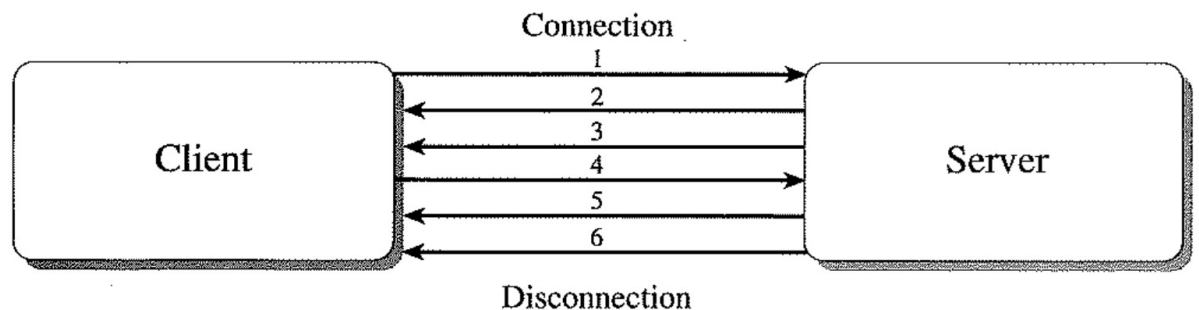


Figure 3.6 Conversational Programming Paradigm

Ex-1107, 35.

195. Tuxedo further teaches that:

Figure 3.6 depicts the flow of data between modules conducting a conversation. The client initiates the communication with a connection message sent to the server (flow 1). In response, the server sends two messages to the client (flows 2 and 3). The client sends another message to

the server (flow 4), which sends a single reply message (flow 5). The sequence is terminated by the server sending a disconnect message to the client (flow 6).

Ex-1107, 34.

196. Tuxedo also discloses that “[w]hen two software modules are conversing, it is necessary for them to keep state” (Ex-1107, 36) and that “it is sometimes the case that a number of messages need to be exchanged between software modules to complete a business operation . . . In such a situation, we say that the modules are conducting a conversation.” Ex-1107, 34.

197. A POSITA would have understood Tuxedo’s state-based approach to conducting a conversation between modules on a client and server to teach “*a conversational communication paradigm*.” Ex-1107, 27 (“A paradigm is a style, or form, of doing something. Communication paradigms are styles of communication.”), 131 (“Then, a chapter is devoted to each of the four communications paradigms: request/response, conversations, application queuing, and events.”), 217-18 (“From a programmer's perspective, the essential difference between request/response and conversational communication is how client and server programs maintain and use state information during an interaction. . . . On the other hand, conversational programs build up and use application state information during their interactions. As we will soon see, even though data is sent and received

between the client and the service, each program also maintains additional context or state information....”), xxxvi (“Release 4.2, available in 1993 and depicted in Figure P.4, provided for the inclusion of personal computers and workstations as client devices for the TUXEDO System. ... Support for conversational and queued communication paradigms was also introduced into ATMI in this release.”), xli (“Chapter 11, "Conversational Communication," presents the TUXEDO System's conversational programming interface. This interface is used when communicating modules need to conduct their business over a sequence of communications.”), 131 (“Then, a chapter is devoted to each of the four **communications paradigms**: request/response, **conversations**, application queuing, and events.”).

198. Consequently, Bowman in view of Vahalia in further view of Tuxedo discloses this limitation.

ii. ***[3.1] transmitting a message from the first node to a communication agent in the FTSS comprises:***

199. For the same reasons described above regarding claim limitation 1.1 (see Section IX.A.1.ii), the combination of Bowman and Vahalia teaches this claim limitation.

iii. ***[3.2] transmitting a connect message from the first node to a conversational communication agent in the FTSS;***

200. For the same reasons described above regarding claim limitation 1.1 (see Section IX.A.1.ii), Bowman in view of Vahalia teaches “transmitting a ...

message from the first node” (e.g., transmitting a file access request from Bowman’s Client) “*to a ... communication agent in the FTSS*” (e.g., to Vahalia’s Data Mover at File Server 60).

201. Tuxedo discloses that “it is sometimes the case that a number of messages need to be exchanged between software modules to complete a business operation ... In such a situation, we say that the modules are conducting a conversation.” Ex-1107, 34. Tuxedo also discloses that Figure 3.6 depicts the flow of data between modules conducting a conversation. *Id.*, 35-37.

202. In Figure 3.6, Client (yellow) initiates communication with a Connection message in flow 1 (blue) sent to Server (pink):

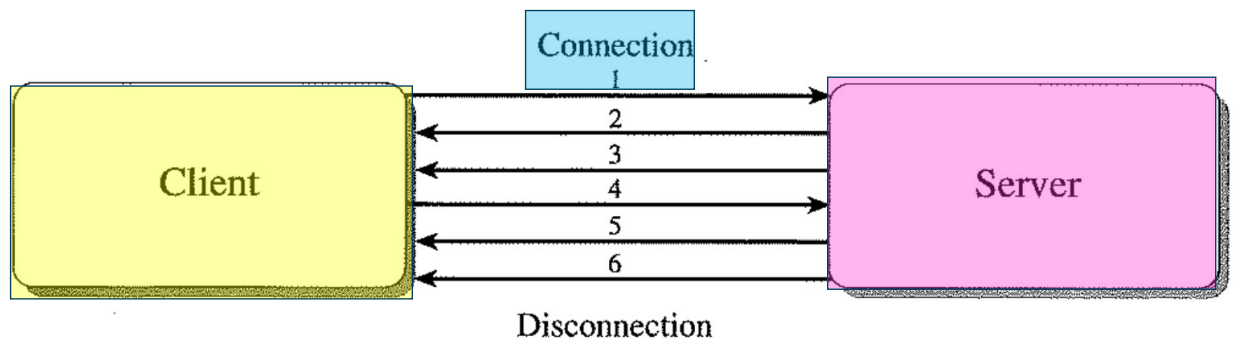


Figure 3.6 Conversational Programming Paradigm

Ex-1107, 35 (“The client initiates the communication with a connection message sent to the server (flow 1).”).

203. A POSITA would have understood the “connection message” from Client to Server that “initiated the communication” to teach “*a connect message.*”

204. Tuxedo further teaches “*a conversational communication agent.*” For the same reasons described above for claim 1.1 (*see* Section IX.A.1.ii), Bowman in view of Vahalia taught or suggested using the communication paradigms from Tuxedo to implement “*a communication agent*” (e.g., Bowman’s MOM and/or Vahalia’s Data Mover). And for the same reasons described for claim limitation 3.0 (*see* Section IX.B.2.i), Bowman in view of Vahalia in further view of Tuxedo teaches using a conversational communication paradigm. A POSITA would have understood a communication agent (e.g., Bowman’s MOM or Vahalia’s Data Mover) implemented using a conversational communication paradigm to teach “*a conversational communication agent.*”

205. Consequently, Bowman in view of Vahalia in further view of Tuxedo discloses this limitation.

iv. [3.3a] processing the message at the FTSS in accordance with a messaging paradigm comprises: opening a conversation queue in highly reliable fault-tolerant storage media of the FTSS;

206. The combination of Bowman and Vahalia discloses this limitation for the same reasons described in Section IX.B.1.v.

207. Tuxedo further discloses the use of a “*queue.*” For instance, Tuxedo describes the concept of indirect communication:

Sometimes it is not possible, required, or desirable, for application programs to communicate on-line. Rather, the communications between the modules are deferred over time. An analogy from everyday experience occurs when you leave a message on a telephone answering machine.

To effect deferred communication between modules, it is necessary for them to "leave messages" for each other. If the communication is to take place across a reboot of the computer, it will be necessary to leave the messages on persistent storage. The persistent storage on an answering machine is the magnetic tape or disk inside of the device.

Ex-1107, 40.

208. To effectuate these deferred (or indirect) communications, Tuxedo discloses “a queue, which is the storage mechanism to hold messages until they are processed.” Ex-1107 at 41. Tuxedo also teaches that such queues (e.g., application queues) are used with “request/response or conversational models”:

There are times when your programs cannot or need not communicate in an online manner using either of the request/response or conversational models. This is precisely why the TUXEDO System provides a means for your programs to **store messages into, and retrieve them from, disk-based storage areas, called application queues**. You should think of an application queue as a database of messages.

Ex-1107, 227.

209. For the reasons explained for claim limitations 2.0 and 3.0 (*see* Sections IX.B.1.i and IX.B.2.i), the “request/response or conversational models” described in Tuxedo are communication paradigms. Ex-1107, 27 (“A paradigm is a style, or form, of doing something. Communication paradigms are styles of communication.”), 74 (describing “TUXEDO System's request/response and conversational paradigms”).

210. Tuxedo teaches that “[y]our application can have one or more queue spaces each of which can contain one or more application queues. Because application queues reside on disk, the messages stored in them are guaranteed to be available across failures that cause the machines in your application to be rebooted.” Ex-1107, 74.

211. For the reasons described above for claim 2.4 (*see* Section IX.B.1.v), Bowman in view of Vahalia in further view of Tuxedo teaches implementing a “queue” (e.g., Tuxedo’s application queue) “*in highly reliable fault-tolerant storage media of the FTSS*” (e.g., in Vahalia’s File Server 60 comprising a cached disk array). Ex-1107, 227 (“There are times when your programs cannot or need not communicate in an online manner using ... conversational models. This is precisely why the TUXEDO System provides a means for your programs to store messages into, and retrieve them from, disk-based storage areas, called application queues.”).

Moreover, a POSITA would have understood Tuxedo's disclosures of retrieving information from such application queues as teaching "*opening*" the application queue. Ex-1107, 227 ("You should think of an application queue as a database of messages."), 283 ("Database administrators provide the information necessary to 'open' the database resources used by the application.").

212. Also, a POSITA would have understood Tuxedo's disclosures about using queues with a conversational communication paradigm to teach or suggest "*a conversation queue*."

213. Consequently, Bowman in view of Vahalia in further view of Tuxedo teaches "*opening a conversation queue*" (e.g., accessing a Tuxedo application queue used for implementing a conversational communication paradigm) "*in highly reliable fault-tolerant storage media of the FTSS*" (e.g., in Vahalia's File Server 60 comprising a cached disk array).

v. ***[3.3b] storing the message in a data structure in highly reliable fault-tolerant storage media of the FTSS comprises:***

214. For the same reasons described above regarding claim limitation 1.2 (see Section IX.A.1.iii), the combination of Bowman and Vahalia teach this claim limitation.

vi. *[3.3c] storing the connect message in the conversation queue; and*

215. Bowman discloses that “Message-Oriented Middleware (MOM) refers to the process of disturbing data and control throughout the exchange of records known as messages. MOM provides the application developer with a set of simple verbs (e.g., **connect**, send, receive, and disconnect) that are used to exchange information with other distributed applications.” Ex-1105, 69:47-70:4.

216. Bowman further discloses that “MOMs are designed for heavily used production applications and are generally capable of high throughput rates and fast transfer times. Data is usually forwarded immediately, although it is possible to store it for later processing.” Ex-1105, 70:50-54.

217. For the reasons described for claim 3.3a (*see* Section IX.B.2.iv), Tuxedo teaches storing messages for a conversational communication paradigm “*in the conversation queue*.” Consequently, Bowman in view of Vahalia in further view of Tuxedo teaches “*storing the connect message*” (e.g., Bowman’s “connect”) “*in the conversation queue*” (e.g., Tuxedo’s application queue).

vii. [3.4] transmitting the message from the FTSS to the second node comprises: transmitting the connect message from the FTSS to the second node.

218. For the same reasons described above regarding claim limitation 1.4 (see Section IX.A.1.v), the combination of Bowman and Vahalia teach this claim limitation.

219. For the same reasons described above regarding claim limitation 3.2 (see Section IX.B.2.iii), Bowman, Vahalia, and Tuxedo each teach a “connect message,” and for the same reasons described above regarding claim limitation 1.4 (see Section IX.A.1.v), the combination of Bowman, Vahalia, and Tuxedo teach “transmitting the message from the FTSS to the second node.”

3. Claim 4

i. [4.0] The method of claim 3 and further comprising:

220. As discussed above (see Section IX.B.2), Bowman in view of Vahalia and Tuxedo discloses “[t]he method of claim 3.”

ii. [4.1] transmitting a send message from the first node to the conversational communication agent in the FTSS;

221. For the same reasons described above regarding claim limitations 1.1 and 3.0 (see Sections IX.A.1.ii and IX.B.2.i), the combination of Bowman, Vahalia, and Tuxedo discloses “transmitting a ... message from the first node to the conversational communication agent in the FTSS.”

222. Bowman further teaches “*transmitting a send message.*” Bowman teaches a communication agent (MOM) that “provides the application developer with a set of simple verbs (e.g., connect, send, receive, and disconnect) that are used to exchange information with other distributed applications.” Ex-1105, 69:49-52. A POSITA would have understood this disclosure of using the verb “*send*” in information exchanges between distributed applications to teach “*transmitting a send message.*” Ex-1105, 69:47-49 (“Message-Oriented Middleware (MOM) refers to the process of distributing data and control throughout the exchange of records known as messages.”).

iii. [4.2] storing the send message in the conversation queue;

223. For the same reasons described above regarding claim limitation 3.3c (see Section IX.B.2.vi), the combination of Bowman, Vahalia, and Tuxedo discloses “*storing the ... message in the conversation queue.*” Further, for the same reasons described above regarding claim limitation 4.1 (see Section IX.B.3.ii), the combination of Bowman, Vahalia, and Tuxedo discloses that the message is a “*send message.*” The combination of Bowman, Vahalia, and Tuxedo thus discloses this limitation.

iv. [4.3] transmitting the send message to from the FTSS to a service on the second node;

224. For the same reasons described above regarding claim limitation 2.8 (see Section IX.B.1.ix), the combination of Bowman, Vahalia, and Tuxedo discloses “transmitting the ... message [from] the FTSS to a service on the second node.” Further, for the same reasons described above regarding claim limitation 4.1 (see Section IX.B.3.ii), the combination of Bowman, Vahalia, and Tuxedo discloses that the message is a “send message.” The combination of Bowman, Vahalia, and Tuxedo thus discloses this limitation.

v. [4.4] executing the service in the second node using the send message as an input and providing a receive message as an output;

225. As demonstrated for claim limitation 2.9a (see Section IX.B.1.x), the combination of Bowman, Vahalia, and Tuxedo discloses “executing the service on the second node using the ... message as an input and providing a ... message as an output.”

226. As demonstrated for claim limitation 2.9a (see Section IX.B.1.x), the combination of Bowman, Vahalia, and Tuxedo teaches transmitting a “send message” to the service. A POSITA would have understood the service on the second node to use the incoming send message as “an input.” Ex-1107, 67, 177.

227. Bowman teaches using “simple verbs” for messaging, including “receive.” Ex-1105, 69:47-52. A POSITA would have understood a message of “receive” as taught in Bowman to be a “*receive message*.” Ex-1105, 69:47-52.

228. Tuxedo discloses “*executing the service in the second node*.” For example, Tuxedo discloses that:

One common use for conversations is to buffer portions of a lengthy reply from a server to a client. In this situation, a client requests information from a server. The server has a lot of reply data, so much, in fact, that the client cannot accept it all at once. In this case, the server might only send part of the data and indicate to the client that it has “more” reply data.

Ex-1107, 35.

229. Tuxedo further teaches “*using the send message as an input and providing a receive message as an output*.” For example, Tuxedo discloses that:

T]he server keeps several items in its state including the fact that there is more data to be sent and the starting point of the reply data that has not yet been sent. When the client asks for “more,” the server starts sending from this point.

Ex-1107, 35.

230. Therefore, the combination of Bowman, Vahalia, and Tuxedo teaches this claim limitation.

vi. [4.5] transmitting the receive message from the second node to the conversational communication agent in the FTSS;

231. As demonstrated for claim limitation 1.1 (see Section IX.A.1.ii), the combination of Bowman and Vahalia discloses “*transmitting the ... message from the second node to the ... communication agent in the FTSS.*”

232. As demonstrated for claim limitation 3.2 (see Section IX.B.2.iii), the combination of Bowman, Vahalia, and Tuxedo discloses the “*message*” being a “*receive message.*”

233. As demonstrated for claim limitation 4.4 (see Section IX.B.3.v), the combination of Bowman, Vahalia, and Tuxedo discloses the “*communication agent*” being a “*conversational communication agent.*” The combination of Bowman, Vahalia, and Tuxedo thus teaches this limitation.

vii. [4.6] storing the receive message in the conversation queue; and

234. As demonstrated for claim limitation 2.4 (see Section IX.B.1.v), the combination of Bowman and Vahalia discloses “*storing the ... message in the ... queue.*”

235. As demonstrated for claim limitation 3.2 (see Section IX.B.2.iii), the combination of Bowman, Vahalia, and Tuxedo discloses the “*message*” being a “*receive message.*”

236. As demonstrated for claim limitation 4.4 (*see* Section IX.B.3.v), the combination of Bowman, Vahalia, and Tuxedo discloses the “*queue*” being a “*conversation queue*.” The combination of Bowman, Vahalia, and Tuxedo thus teaches this limitation.

viii. [4.7] transmitting the receive message to the first node.

237. As demonstrated for claim limitation 2.9e (*see* Section IX.B.1.xiv), the combination of Bowman and Vahalia discloses “*transmitting the ... message to the first node*.” As demonstrated for claim limitation 3.2 (*see* Section IX.B.2.iii), the combination of Bowman, Vahalia, and Tuxedo discloses the “*message*” being a “*receive message*.” The combination of Bowman, Vahalia, and Tuxedo thus teaches this limitation.

4. Claim 5

i. [5.0] The method of claim 3 and further comprising:

238. As discussed above (*see* Section IX.B.2), Bowman in view of Vahalia and Tuxedo discloses “[*t*]he method of claim 3.”

ii. [5.1] transmitting a disconnect message from the first node to the conversational communication agent in the FTSS;

239. For the reasons described above for claim limitation 1.1 (*see* Section IX.A.1.ii), the combination of Bowman and Vahalia discloses “*transmitting a ... message from the first node to the ... communication agent in the FTSS*.”

240. For the reasons described above for claim limitation 3.2 (*see* Section IX.B.2.iii), the combination of Bowman, Vahalia, and Tuxedo discloses the “*communication agent*” being a “*conversational communication element*.”

241. Tuxedo further teaches “*transmitting a disconnect message*.” Specifically, Tuxedo discloses “*transmitting a disconnect message from the first node*.” For example, Tuxedo teaches that “[f]or a conversation, the termination will normally happen by agreement between the modules. One side or the other will terminate the conversation, which is presumably accepted by the other.” Ex-1107, 35-36. This is illustrated in Tuxedo’s Figure 3.6:

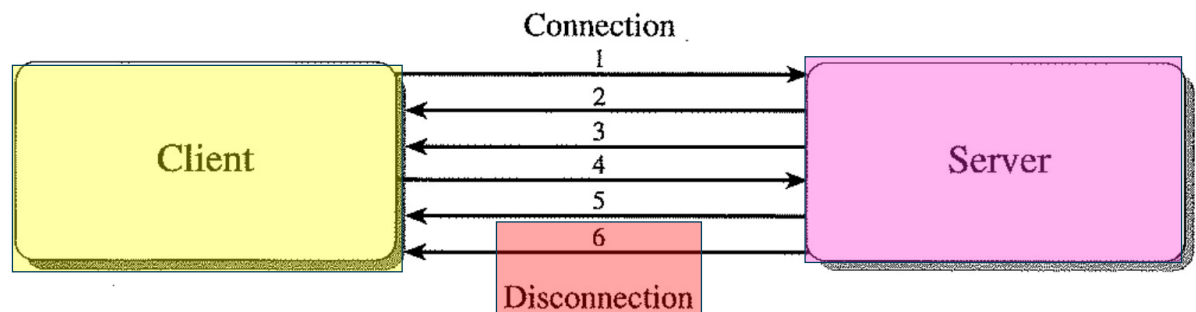


Figure 3.6 Conversational Programming Paradigm

Ex-1107, 35-36.

242. The conversation is terminated by Server (pink) sending a “*disconnect message*” at flow 6 (red) to Client (yellow). *Id.*, 34 (“The sequence is terminated by the server sending a **disconnect message** to the client (flow 6).”). Accordingly, a POSITA would have understood Server in this arrangement to be “*transmitting a disconnect message*.” Ex-1107, 34.

243. Thus, the combination of Bowman, Vahalia, and Tuxedo teaches this claim limitation.

iii. *[5.2] closing the conversation queue; and*

244. Tuxedo explains that a conversation “is terminated by the server sending a disconnect message to the client.” Ex-1107, 34 (“The sequence is terminated by the server sending a **disconnect message** to the client (flow 6).”).

245. A POSITA would have understood that “terminating” the conversation to include “*closing the conversation queue*” by releasing access to the storage space on which the queue is stored. Ex-1107, 227 (“You should think of an application queue as a database of messages.”), 283 (“Database administrators provide the information necessary to ‘open’ the database resources used by the application.”), 175 (“After a client program is finished processing requests for your TUXEDO application, the client should leave the application by calling the termination procedure, *tpterm*. This will unregister the client from the application, and free up resources that have been allocated while the client was part of the application. These resources include communications resources (**conversational connections are closed**), transaction resources (transactions are rolled back), IPC resources, and memory resources”). A POSITA would have understood Tuxedo’s description of the unregistering and freeing of resources to be the process of closing as understood within the ’244 Patent. Ex-1114, 90 (defining “close” as “[t]o end an application’s

relationship with an open file so that the application will no longer be able to access the file without opening it again.”), 516 (defining “terminate” as “to end a process or program.”).

246. Thus, the combination of Bowman, Vahalia, and Tuxedo teaches this claim limitation.

iv. [5.3] transmitting the disconnect message to the second node.

247. For the same reasons described above for claim limitation 1.4 (see Section IX.A.1.v), the combination of Bowman and Vahalia discloses “*transmitting the ... message to the second node.*”

248. For the same reasons described above for claim limitation 5.1 (see Section IX.B.4.ii), the combination of Bowan, Vahalia, and Tuxedo teaches the message as being a “*disconnect message.*”

5. Claim 6

i. [6.0] The method of claim 1 wherein the messaging paradigm is an event-based communication paradigm, and:

249. As described above (see Section VIII.A.1), Bowman and Vahalia disclose the method of claim 1.

250. Tuxedo teaches an “*event-based communication paradigm.*” For instance, Tuxedo discloses:

The fourth major communication paradigm offered in ATMI is event-based communication. Unlike the client/server models where a server is waiting at-the-ready to process a client's request, the essence of event-based communication is that a program being contacted is not waiting for that communication to take place. Rather, it registers that it is willing to be contacted, but it does not actively wait or poll for communication to occur.

Ex-1107, 241.

251. A POSITA would have understood this to teach that “*the messaging paradigm is an event-based communication paradigm.*” Ex-1107, 241-42. Thus, the combination of Bowman, Vahalia, and Tuxedo teaches this claim limitation.

- ii. ***[6.1] transmitting a message from the first node to a communication agent in the FTSS comprises: transmitting a post message from the first node to an event-based communication agent in the FTSS;***

252. For the same reasons described above regarding claim limitation 1.1 (see Section IX.A.1.ii), the combination of Bowman and Vahalia discloses this limitation.

253. For the same reasons described above regarding claim limitation 1.1 (see Section IX.A.1.ii), the combination of Bowman and Vahalia discloses “*transmitting a ... message from the first node to [a] ... communication agent in the FTSS.*”

254. Tuxedo teaches “*a post message*.”

255. The ’244 Patent describes the use of post messages in an event-based messaging paradigm as being a message that is sent after an event generator notifies the event broker or “posts” an event. Ex-1101, 2:50-60.

256. Similarly, as part of its event-based communication paradigm, Tuxedo describes “[b]rokered events” that “allow programs to post events without having to worry which other program are supposed to receive notification of an event’s occurrence.” Ex-1107, 250. It further teaches that “[a]ny client or service can post an event.” *Id.* It also defines “event posting” as “sending an event to the EventBroker that will be forwarded to interested targets.” *Id.*, 418 (defining “EventBroker” as “module that forwards posted events to interested (registered) targets”). A POSITA would have understood an event posting sent to EventBroker, or an event forwarded by EventBroker to interested targets, to be “*a post message*” as described in the ’244 Patent. Ex-1107, 421 (defining “message” as “data exchanged between communicating executables”), 250, 418.

257. A POSITA would have understood that when the “*communication agent*” (e.g., Data Mover from Vahalia) is implementing an event-based communication paradigm, the communication agent (e.g., Data Mover) is “*an event-based communication agent*.” Ex-1107, 413 (defining “agent” as “a software module which carries out the instructions of another module”).

258. Thus, the combination of Bowman, Vahalia, and Tuxedo teaches this claim limitation.

- iii. *[6.2] processing the message at the FTSS in accordance with a messaging paradigm comprises: accessing a subscription manager to determine whether any nodes have subscribed to receive the event represented by the post message;*

259. For the same reasons described above regarding claim limitation 1.3 (see Section IX.A.1.iv), the combination of Bowman and Vahalia discloses this limitation.

260. Tuxedo discloses using an event broker (red) to maintain a subscription list (yellow):

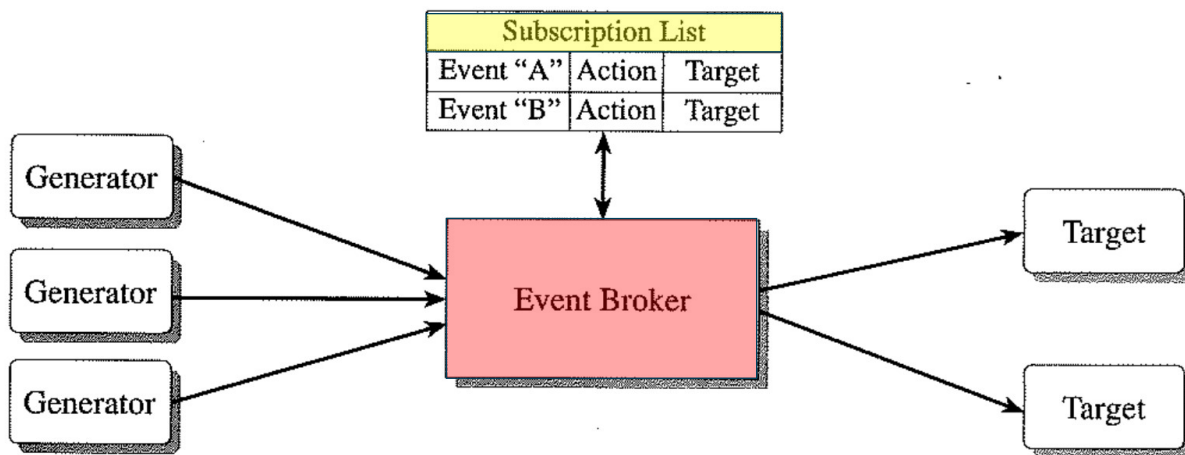


Figure 3.8 Delivery of Events via a Broker

Ex-1107, 39.

261. Tuxedo explains that the event broker's function "is to accept target registrations and to deliver events." *Id.* In this arrangement, "event generators and targets need only to inform the broker, which takes care of the rest." *Id.*

262. A POSITA would have understood this arrangement to teach accessing a subscription manager (event broker) to determine whether any nodes have subscribed to receive the event represented by the post message (i.e., to determine whether subscription list indicates that any targets should be delivered an event based on the action, such as receiving the post message). Ex-1107, 39-40. The combination of Bowman, Vahalia, and Tuxedo thus teaches this limitation.

- iv. ***[6.3] storing the message in a data structure in highly reliable fault-tolerant storage media of the FTSS comprises: storing the post message in an event queue in highly reliable fault-tolerant storage media of the FTSS; and***

263. For the same reasons described above regarding claim limitation 1.2 (see Section IX.A.1.iii), Bowman and Vahalia disclose this limitation.

264. For the same reasons described above regarding claim limitations 2.4 (see Section IX.B.1.v), Bowman, Vahalia, and Tuxedo disclose “*storing the ... message in ... [a] queue in highly reliable fault-tolerant storage media of the FTSS.*”

265. For the same reasons described above regarding claim limitation 6.2 (see Section IX.B.5.iii), the combination of Bowman, Vahalia, and Tuxedo discloses the message as “*a post message.*”

- v. *[6.4] transmitting the message from the FTSS to the second node comprises: transmitting the post message from the FTSS to the second node if the second node has subscribed to receive the event represented by the post message.*

266. For the same reasons described above regarding claim limitation 1.4 (see Section IX.A.1.v), Bowman discloses this limitation.

267. For the same reasons described above regarding claim limitation 1.4 (see Section IX.A.1.v), the combination of Bowman and Vahalia teaches “transmitting the ... message from the FTSS to the second.” For the same reasons described above regarding claim limitation 6.2 (see Section IX.B.5.iii), the combination of Bowman, Vahalia, and Tuxedo discloses the message as “a post message.”

268. Tuxedo teaches transmitting the post message “if the second node has subscribed to receive the event represented by the post message.” As described above regarding claim limitation 6.4 (see Section IX.B.5.v), Tuxedo teaches using event broker to maintain a subscription list. It further explains that “event targets register with the broker, which keeps a subscription list of such registrations. Event generators inform the broker when they wish to indicate an event has occurred. The broker fulfills the subscriptions by taking actions to notify the targets.” Ex-1107, 40. A POSITA would have understood this to mean transmitting the post message from the FTSS to the second node “if the second node” (the target) “has subscribed” (has

been registered on the subscription list) “*to receive the event represented by the post message*” (e.g., to be notified when the action occurs). Ex-1107, 39-40.

269. The combination of Bowman, Vahalia, and Tuxedo thus discloses this limitation.

6. Claim 7

i. [7.0] *The method of claim 6 and further comprising:*

270. As discussed above (*see* Section IX.B.5), Bowman in view of Vahalia and Tuxedo discloses “[*t*]he method of claim 6.”

ii. [7.1] *transmitting a subscribe message from the second node to the event-based communication manager in the FTSS, wherein the subscribe message includes an event; and*

271. For the same reasons described above regarding claim limitation 2.9b (*see* Section IX.B.1.xi), the combination of Bowman, Vahalia, and Tuxedo discloses “*transmitting a ... message from the second node.*” For the same reasons described above regarding claim limitation 6.2, Bowman, Vahalia, and Tuxedo “*discloses the event-based communication manager in the FTSS.*”

272. Tuxedo teaches “*a subscribe message.*” Tuxedo explains that event broker will “accept target registrations” and that “targets ‘register’ their interest in events with the broker.” Ex-1107, 39. In such instances, “targets need only to inform the broker, which takes care of the rest.” *Id.* “Event targets register with the broker, which keeps a subscription list of such registrations.” *Id.*, 40. A POSITA would have

understood that the targets “inform” and “register” with the event broker via messages—and these messages directed to being added to the subscription list are accordingly “*subscribe message[s]*.” Ex-1107, 39-40.

273. Moreover, Tuxedo explains that the targets are indicating to the event broker “their interest in events.” Ex-1107, 39 (“Targets ‘register’ their interest in events with the broker.”). Tuxedo further teaches that “brokered event communication” include “[e]vent identifiers” that are used to name events and “[e]vent data” that is “associated with an event.” A POSITA would have understood that these communications, which are intended to register a target for notifications of a particular event, would include an indication of the event in interest—i.e., that “*the subscribe message includes an event.*” Ex-1107, 39-40.

274. The combination of Bowman, Vahalia, and Tuxedo thus discloses this limitation.

iii. [7.2] *accessing the subscription manager to record that the second node has subscribed to receive post messages associated with the event.*

275. For the same reasons described above regarding claim limitation 6.1 (see Section IX.B.5.ii), the combination of Bowman, Vahalia, and Tuxedo teaches “*post messages.*”

276. Tuxedo teaches that for a target to register for an event a record must be placed into the event broker’s subscription list. Ex-1107, 40 (“Here, event targets

register with the broker, which keeps a subscription list of such registrations.”). A POSITA would have understood this to teach “*accessing the subscription manager*” (i.e., requesting that event broker register the target’s interest) “*to record that the second node has subscribed*” (to enter the target’s interest into the subscription list). Ex-1107, 39-40.

277. Tuxedo further teaches that a target’s entry on the subscription list subscribes the target to receive notifications when the relevant event takes place. Ex-1107, 40 (“Event generators inform the broker when they wish to indicate an event has occurred. The broker fulfills the subscriptions by taking actions to notify the targets.”). A POSITA would have understood these notifications informing the target about the event to be post messages “*associate with the event.*” Ex-1107, 39-40.

278. The combination of Bowman, Vahalia, and Tuxedo thus discloses this limitation.

7. Claim 8

i. [8.0] *The method of claim 6 and further comprising:*

279. As discussed above (*see* Section IX.B.5), Bowman in view of Vahalia and Tuxedo discloses “[t]he method of claim 6.”

- ii. ***[8.1] transmitting an unsubscribe message from the second node to the event-based communication manager in the FTSS, wherein the unsubscribe message includes an event; and***

280. For the same reasons described above regarding claim limitation 7.1 (see Section IX.B.6.ii), the combination of Bowman, Vahalia, and Tuxedo discloses “transmitting [a] ... message from the second node to the event-based communication manager in the FTSS, wherein the ... message includes an event.”

281. Tuxedo teaches that targets may unsubscribe from events. Ex-1107, 138, 255 (“When tpsubscribe successfully registers a subscription with the EventBroker, it returns a handle for the subscription that any program can use later for removing it via tpunsubscribe.”), 256 (“The first parameter to tpunsubscribe is a subscription handle returned previously from tpsubscribe. The second parameter is the flags settings (TPNOBLOCK, TPNOTIME, and TPSIGRSTRT). A client should remove its ‘personal’ subscriptions before exiting.”). A POSITA would have understood, particularly in light of the subscribe messages described in claim limitation 7.1 (see Section IX.B.6.ii), that these unsubscribe requests and related parameters are delivered to event broker as a message—i.e., as “an unsubscribe message.” Ex-1107, 39-40, 255-56.

282. The combination of Bowman, Vahalia, and Tuxedo thus discloses this limitation.

- iii. ***[8.2] accessing the subscription manager to record that the second node should no longer receive post messages associated with the event.***

283. As described above regarding claim limitation 6.4 (*see* Section IX.B.5.v), Tuxedo teaches that event broker maintains a subscription list indicating which targets have registered to receive post messages for which events, and as describe above regarding claim limitation 8.1 (*see* Section IX.B.7.ii), Tuxedo teaches that event broker may receive unsubscribe messages to remove targets from the subscription list.

284. A POSITA would have understood these disclosures to teach or suggest “*accessing the subscription manager*” (e.g., requesting that event broker modify the subscription list) “*to record that the second node should no longer receive post messages associated with the event*” (to remove the entry from the subscription list that indicates the target should receive notifications when the action occurs). Ex-1107, 39-40, 255-56.

285. Tuxedo accordingly discloses this limitation.

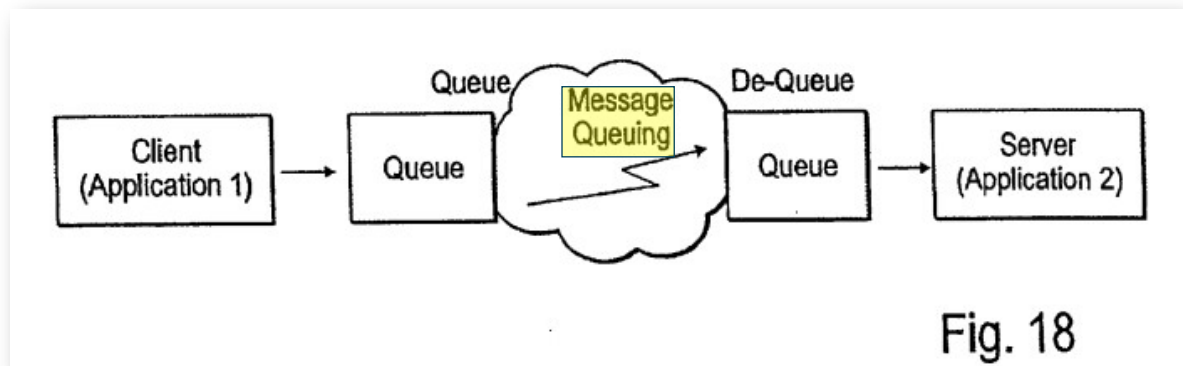
8. Claim 9

- i. ***[9.0] The method of claim 1 wherein the messaging paradigm is a queue-based communication paradigm, and:***

286. As described above (*see* Section VIII.A.1), Bowman and Vahalia disclose the method of claim 1, including the use of a “*messaging paradigm*.”

287. Further, as described above regarding claim limitations 2.4, 2.6, 2.9c, 2.9d, 3.3c, 4.2, 4.6, 5.2, and 6.3 (*see* Sections IX.B.1.v, IX.B.1.vii, IX.B.1.xii, IX.B.1.xiii, IX.B.2.vi, IX.B.3.iii, IX.B.3.vii, IX.B.4.iii, and IX.B.5.iv), the combination of Bowman, Vahalia, and Tuxedo discloses using queues to implement messaging, including Tuxedo's request queues, reply queues, and application queues.

288. Bowman also illustrates Message Queuing (yellow) in Figure 18:



Ex-115, Fig. 18.

289. A POSITA would have understood using queues to facilitate messaging (e.g., Tuxedo's application queues and Bowman's Message Queueing) to teach the “messaging paradigm” being “a queue-based communication paradigm.” Ex-1107, 27 (“A paradigm is a style, or form, of doing something. Communication paradigms are styles of communication.”), 131 (“Then, a chapter is devoted to each of the four communications paradigms: request/response, conversations, application queuing, and events.”).

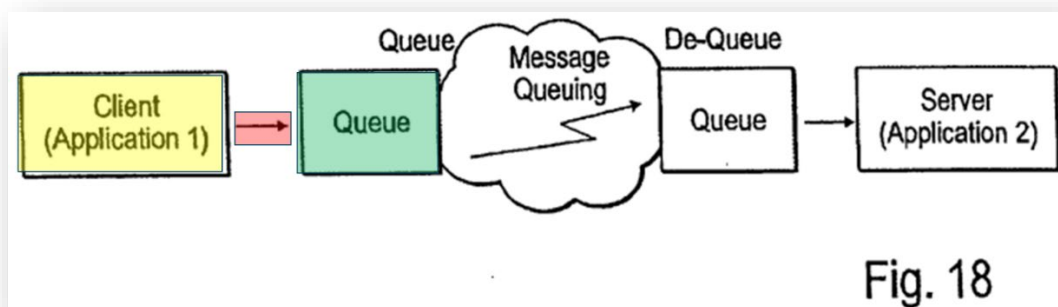
290. The combination of Bowman, Vahalia, and Tuxedo thus discloses this limitation.

- ii. ***[9.1] transmitting a message from the first node to a communication agent in the FTSS comprises: transmitting an enqueue message from the first node to a queue-based communication agent in the FTSS;***

291. For the same reasons described above regarding claim limitation 1.1 (see Section IX.A.1.ii), Bowman discloses this limitation.

292. For the same reasons described above regarding claim limitation 1.1 (see Section IX.A.1.ii), the combination of Bowman and Vahalia teaches “transmitting [a] ... message from the first node to a ... communication agent in the FTSS.”

293. In Figure 18, Bowman discloses transmitting a message (red) from client (yellow) to queue (green):



Ex-1105, Fig. 18.

294. Bowman explains that Figure 18 shows “an indirect application to application communication model that allows applications to communicate via message queues, rather than by calling each other directly.” *Id.* at 70:22-26. The ’244 Patent describes an “*enqueue message*” as a message which has been pushed into a message queue. *See* Ex-1101, 3:12-17, 9:45-53, 10:1-5. A POSITA would have understood the message in Figure 18 from client to the queue to be “*an enqueue message*” as described in the ’244 Patent. Ex-1107, 74 (“Both TUXEDO client and server programs can enqueue messages into or dequeue messages from application queues.”).

295. Tuxedo provides further details about enqueueing messages. It explains that “a queued message is a request for some future processing. The queue serves as a place to hold the input to these requests.” Ex-1107, 42. In such arrangements, “an enqueueing module will write one or more items to a queue.” *Id.* Thus, “[t]he queue manager is responsible for enqueueing messages from message producers and providing them to message consumers that request them via dequeue operations.”

296. Given these disclosures in Tuxedo of “enqueueing messages” (e.g., receiving the message in Bowman’s Figure 18 at queue) from “message producers” (e.g., Bowman’s client), a POSITA would have understood Bowman and Tuxedo to each disclose “*an enqueue message*.” Ex-1107, 421 (defining “message” as “data exchanged between communicating executables”), 424 (defining “queued

communications” as “deferred communications between two modules, ... so that the message is stored in a queue until the communications can be completed”).

297. Bowman in view of Tuxedo further teaches that the “Message Queuing” in Figure 18 is performed by “*a queue-based communication agent.*” Tuxedo teaches using a “queue manager” as a “broker” to monitor and control the queue. Ex-1107, 42 (“Additionally, there is an anonymity present between the enqueueer and the dequeuer of messages on a queue. Similar to the event broker, the queue is monitored and controlled by a broker called the queue manager, as shown in Figure 3.9.”). This queue manager “is responsible for enqueueing messages from message producers and providing them to message consumers that request them via dequeue operations.” *Id.* A POSITA would have understood Tuxedo’s queue manager to be “*a queue-based communication agent.*” Ex-1107, 413 (defining “agent” as “a software module which carries out the instructions of another module”); Ex-1113, 500 (“In the computer programming sense of the word, an agent acts on behalf of another person or thing, with delegated authority.”). Further, as described for claim limitations 1.1 and 1.2, a POSITA would have understood based on the combination of Bowman and Vahalia that this queue-based communication agent is “*in the FTSS.*”

298. The combination of Bowman, Vahalia, and Tuxedo thus discloses this limitation.

- iii. ***[9.2] storing the message in a data structure in highly reliable fault-tolerant storage media of the FTSS comprises: storing the enqueue message in a queue space in highly reliable fault tolerant storage media of the FTSS;***

299. For the same reasons described above regarding claim limitation 1.2 (see Section IX.A.1.iii), Bowman and Vahalia disclose this limitation.

300. For the same reasons described above regarding claim limitation 2.4 (see Section IX.B.1.v), the combination of Bowman, Vahalia, and Tuxedo discloses “storing the message in a queue ... in highly reliable fault tolerant storage media of the FTSS.” For the same reasons described above regarding claim limitation 9.2 (see Section IX.B.8.iii), the combination of Bowman, Vahalia, and Tuxedo discloses that “the message” is an “enqueue message.”

301. Tuxedo teaches using a “queue space” as “a collection of one or more application queues, used for doing queues communications.” Ex-1107, 424 (defining “queue space”), 74 (“A queue space is a collection of diskbased application queues.”), 228 (“Your application administrator uses TUXEDO administrative tools to create queue spaces and the queues that reside within them.”). For instance, Tuxedo explains that its request and reply queues are part a queue space. Ex-1107, 230 (“Regardless, the reply queue must be the name of a queue that exists within the same queue space as the queue where the enqueuer placed the request message.”), 231 (“Like the reply queue, the failure queue must exist in the same queue space as

the queue in which the initial message was enqueued.”). Tuxedo teaches that the queue space is the storage location for the queues and the messages the queues contain. Ex-1107, 355 (“A queue space defines a storage location for queues and messages. Queue spaces are stored as system files on a TUXEDO file-system.”).

302. A POSITA would thus have understood the combination of Bowman, Vahalia, and Tuxedo to teach this limitation.

iv. *[9.3] processing the message at the FTSS in accordance with a messaging paradigm comprises: retrieving the enqueue message from the queue space only if the second node is available; and*

303. For the same reasons described above regarding claim limitation 1.3 (see Section IX.A.1.iv), Bowman and Vahalia disclose this limitation.

304. Tuxedo teaches enqueueing messages when the recipient (e.g., a service running on a server) is unavailable so that the messages can later be processed when the recipient is available:

Let's say your client program tries to send an online request to a service routine only to find that the service is not available. This could happen because the service has not yet been started or perhaps the machine on which it was running crashed, thus making the service unavailable. Whatever the case, your client has just finished collecting and formatting important data that needs to be processed by a service routine that is no longer available.

One common use of application queues is for storing such requests and having TUXEDO's forwarding agent dequeue and forward them to service routines. From the TUXEDO programmer's view, the on-line service request that failed could be enqueued in an application queue for later processing. Thus, the program can recover from the on-line error by enqueueing the request to an application queue and still be given the guarantee that its request will be processed at a later time.

Ex-1107, 78 (“Here, the program shifts from an on-line request/response, where the client waits synchronously for the response, to a time-independent request/response, where the client is guaranteed that its request will eventually arrive at a service but it most likely will not wait synchronously for a reply to be returned via an application queue.”).

305. Because Tuxedo teaches enqueueing the message in the queue (which resides in the queue space as discussed for claim limitation 9.4) when a service is offline for later processing, and then dequeuing and forwarding them later (i.e., when the service is online), a POSITA would have understood Tuxedo to disclose “*retrieving the enqueue message from the queue space*” (e.g., dequeuing the message from the application queue) “*only if the second node is available*” (i.e., processing the message once the service is no longer offline). Ex-1107, 78, 240 (“In addition,

we covered the forwarding agent, which is a system-supplied server built upon these primitives, that allows you to combine both the request/response and the queuing models. This allows you to add much flexibility to your application because your application is no longer constrained to do all of its work on-line. Thus, if some part of your application is unavailable, work for that part of the system can be enqueued and processed later when it is available again.”).

306. Bowman, Vahalia, and Tuxedo accordingly disclose this limitation.

- v. ***[9.4] transmitting the message from the FTSS to the second node comprises: transmitting the enqueue message from the FTSS to a service on the second node only if the second node is available;***

307. As demonstrated for claim limitation 1.2 (*see* Section IX.A.1.iii), Bowman and Vahalia disclose “*transmitting the ... message from the FTSS to a service on the second node.*”

308. As demonstrated for claim limitation 9.4 (*see* Section IX.B.8.v), Tuxedo teaches enqueueing messages when a service is unavailable for later processing (e.g., dequeuing) when the service is available.

309. Tuxedo further teaches that processing the message when the service becomes available also includes forwarding the message to the service. Ex-1107, 78 (“One common use of application queues is for storing such requests and having TUXEDO's forwarding agent dequeue and forward them to service routines.”). A

POSITA would have understood this to disclose “*transmitting the enqueue message*” (i.e., forwarding the message from the queue) ... to a service on the second node (i.e., to the service that was previously unavailable) only if the second node is available (i.e., forwarding the message once the service is available). Ex-1107, 78, 240.

310. The combination of Bowman, Vahalia, and Tuxedo that discloses this limitation.

- vi. ***[9.5a] and the method further comprises: executing the service on the second node using the enqueue message as an input and providing a dequeue message as an output;***

311. For the same reasons described above regarding claim limitation 2.1 (see Section IX.B.1.ii), the combination of Bowman, Vahalia, and Tuxedo discloses “*executing the service on the second node using the ... message as an input and providing a ... message as an output.*” As described for claim limitations 9.2 and 9.1 (see Sections IX.B.8.ii and IX.B.8.iii), Bowman and Tuxedo further teach enqueue messages and dequeue messages. The combination of Bowman, Vahalia, and Tuxedo accordingly teaches this limitation.

vii. [9.5b] transmitting the dequeue message from the second node to the queue-based communication agent in the FTSS;

312. For the same reasons described above regarding claim limitation 2.9b (see Section IX.B.1.xi), the combination of Bowman, Vahalia, and Tuxedo teaches “transmitting the ... message from the second node to the ... communication agent in the FTSS.” For the same reasons described above regarding claim limitations 9.2 and 9.1 (see Sections IX.B.8.ii and IX.B.8.iii), the combination of Bowman, Vahalia, and Tuxedo teaches “a queue-based communication agent and a deque message.” The combination of Bowman, Vahalia, and Tuxedo accordingly teaches this limitation.

viii. [9.5c] storing the dequeue message in the queue space;

313. For the same reasons described above regarding claim limitation 2.1 (see Section IX.B.1.ii), the combination of Bowman and Vahalia teaches “storing the ... message in a data structure,” including a queue. See also claim limitations 2.3 and 2.4 (see Sections IX.B.1.iv and IX.B.1.v). As discussed for claim limitations 9.4 and 9.1 (see Sections IX.B.8.ii and IX.B.8.v), the combination of Bowman, Vahalia, and Tuxedo further discloses a “dequeue message” and teaches that the queue (and therefore the messages in the queue) is stored in a “queue space.” The combination of Bowman, Vahalia, and Tuxedo accordingly teaches this limitation.

ix. [9.5d] retrieving the dequeue message from the queue space; and

314. For the same reasons described above regarding claim limitation 2.6 (see Section IX.B.1.vii), the combination of Bowman, Vahalia, and Tuxedo discloses “retrieving the ... message from the queue.” As discussed for claim limitation 9.4, the combination of Bowman, Vahalia, and Tuxedo teaches that the queue is part of a queue space, such that the message is retrieved “from the queue space.” Further, as explained for claim limitation 9.1 (see Section IX.B.8.ii), the combination of Bowman, Vahalia, and Tuxedo teaches that the message is a “dequeue message.” The combination of Bowman, Vahalia, and Tuxedo accordingly teaches this limitation.

x. [9.5e] transmitting the dequeue message from the FTSS to the first node.

315. For the same reasons described above regarding claim limitation 2.1 (see Section IX.B.1.ii), the combination of Bowman, Vahalia, and Tuxedo discloses “transmitting the ... message from the FTSS to the first node.” Further, as explained for claim limitation 9.1 (see Section IX.B.8.ii), the combination of Bowman, Vahalia, and Tuxedo teaches that the message is a “dequeue message.” The combination of Bowman, Vahalia, and Tuxedo accordingly teaches this limitation.

9. Claim 10

- i. ***[10.0] The method of claim 1 wherein the messaging paradigm is an unsolicited communication paradigm, and:***

316. For the same reasons described above regarding claim 1 (*see* Section IX.A.1), the combination of Bowman and Vahalia disclose claim 1.

317. Tuxedo discloses unsolicited communication in the form of an unsolicited notification when information is available or something has taken place, and terms such an unsolicited communication an “event”:

Request/response interactions are a form of polling. The requestor is asking for some information from, or for an action to be taken by, a server. The server replies when asked, but otherwise is idle. Sometimes software would rather not ask-polling can be tedious and resource-consumption intensive. Rather, it would like to be notified when the information it wants is available or, more generally, that “something has happened.”

...

We call an unsolicited communication of this form an *event*.

Ex-1107 at 37-38 (emphasis in original), 79-80 (“ATMI has two mechanisms to handle simple events: single-program notifications and multiple-program broadcasting. These mechanisms allow your programs to communicate unsolicited notifications to client programs.”).

318. Because Tuxedo's "event" mechanism describes a form of providing unsolicited communications, a POSITA would have understood it to be "*an unsolicited communication paradigm*." Ex-1107, 27 ("A paradigm is a style, or form, of doing something. Communication paradigms are styles of communication."). The combination of Bowman, Vahalia, and Tuxedo thus discloses this limitation.

- ii. ***[10.1] transmitting a message from the first node to a communication agent in the FTSS comprises: transmitting a broadcast message from the first node to an unsolicited communication agent in the FTSS;***

319. For the same reasons described above regarding claim limitation 1.1 (see Section IX.A.1.ii), Bowman discloses this limitation.

320. For the same reasons described above regarding claim limitation 1.1 (see Section IX.A.1.ii), Bowman discloses "*transmitting a ... message from the first node to [a] ... communication agent in the FTSS*."

321. Tuxedo discloses transmitting "*a broadcast message*." Specifically, Tuxedo teaches two mechanisms of event messaging: "single-program notifications and multiple-program broadcasting." Ex-1107, 79. The multiple-program broadcasting mechanism "allows one or more clients to receive a single unsolicited notification." Ex-1107, 80. A POSITA would have understood a message destined for delivery to multiple clients as in Tuxedo's multiple-program broadcasting to be "*a broadcast message*." Ex-1107, 79-80, 171-72, 241-42, 248-250, 421 (defining

“multiple-program broadcasting” as “communication mechanism used by a client or server to send an unsolicited notification, along with any data, to one or more client programs”).

322. A POSITA would have understood that when the communication agent (e.g., Data Mover from Vahalia) implements an unsolicited communication paradigm (*see* analysis for claim limitation 10.0 in Section IX.B.9.i) (e.g., the multiple-program broadcasting from Tuxedo), the communication agent is “*an unsolicited communication agent*.” Ex-1107, 413 (defining “agent” as “a software module which carries out the instructions of another module”), 171 (“Message broadcasting is also one of the mechanisms used for notification from the Event Broker.”). The combination of Bowman, Vahalia, and Tuxedo thus discloses this limitation.

iii. [10.2] processing the message at the FTSS in accordance with a messaging paradigm comprises: accessing determining¹ which nodes should receive the broadcast message based on the broadcast message;

323. For the same reasons described above regarding claim limitation 1.3 (*see* Section IX.A.1.iv), the combination of Bowman and Vahalia discloses this limitation.

¹ Claim [10.2] includes what appears to be a typographical error (“accessing determining which nodes”). I have been informed to treat this term here to mean “determining which nodes.”

324. As discussed for claim limitation 6.4 (*see* Section IX.B.5.v), Tuxedo teaches accessing a subscription manager (event broker) to determine whether any nodes (targets) have subscribed to receive the event (the action). Tuxedo further teaches that “[m]essage broadcasting is also one of the mechanisms used for notification from the Event Broker.” Ex-1107, 171. A POSITA would accordingly have understood that Tuxedo teaches “*determining which nodes should receive the broadcast message*” (i.e., determining whether subscription list indicates that nodes should receive the broadcast message) “*based on the broadcast message*” (i.e., by determining whether there is a relationship between the node and the broadcast message in the subscription list). Ex-1107, 39-40. Thus, the combination of Bowman, Vahalia, and Tuxedo disclose this limitation.

- iv. ***[10.3] storing the message in a data structure in highly reliable fault-tolerant storage media of the FTSS comprises: storing the broadcast message in an unsolicited message queue in highly reliable fault-tolerant storage media of the FTSS; and***

325. For the same reasons described above regarding claim limitation 1.3 (*see* Section IX.A.1.iv), the combination of Bowman and Vahalia discloses this limitation.

326. For the same reasons described above regarding claim limitation 2.4 (*see* Section IX.B.1.v), the combination of Bowman, Vahalia, and Tuxedo teaches “*storing the ... message in an ... message queue in highly reliable fault-tolerant*

storage media of the FTSS.” As explained for claim limitation 10.0 (*see* Section IX.B.9.i), Tuxedo further teaches an unsolicited communication paradigm based on broadcasting events. And as discussed for claim limitation 6.3 (*see* Section IX.B.5.iv), Tuxedo teaches that events are stored in a queue. A POSITA would have understood the event queue discussed in claim limitation 6.3 when used for multi-program broadcasting described for claim 10.2 (*see* Section IX.B.9.iii) to teach using the event queue as “*an unsolicited message queue.*” The combination of Bowman, Vahalia, and Tuxedo thus discloses this limitation.

- v. ***[10.4] transmitting the message from the FTSS to the second node comprises: transmitting the post message from the FTSS to all nodes that should receive the broadcast message.***

327. For the same reasons described above regarding claim limitation 1.4 (*see* Section IX.A.1.v), Bowman discloses this limitation.

328. As described for claim limitation 10.4 (*see* Section IX.B.9.v), Tuxedo teaches that event broker uses subscription list to determine which targets should receive the event notification. Tuxedo further teaches transmitting the event notification (i.e., the broadcast message or post message) to the nodes that are registered to receive it. Ex-1107, 39 (explaining event broker’s function “to deliver events”), 40 (“The broker fulfills the subscriptions by taking actions to notify the targets.”), 79-80, 171-72, 241-42, 248-250, 421 (defining “multiple-program

broadcasting” as “communication mechanism used by a client or server to send an unsolicited notification, along with any data, to one or more client programs”).

329. A POSITA would have understood these disclosures of notifying the multiple clients listed in subscription list as desiring notifications about an event occurrence as “*transmitting the post message*” (transmitting the event notification) ... “*to all nodes that should receive the broadcast message*” (to all of the targets in subscription list that have registered to receive such event notifications). Ex-1107, 39-40, 79-80, 171-72, 241-42, 248-250, 421.

330. The combination of Bowman, Vahalia, and Tuxedo accordingly discloses this limitation.

X. CONCLUSION

I declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and that these statements were made with knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under section 1101 of Title 18 of the United States Code.

January 15, 2025
Date


Kevin C. Almeroth, Ph.D.