

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

SAP America, Inc.,
Petitioner,

v.

Valtrus Innovations Ltd. and Key Patent Innovations Ltd.,
Patent Owner.

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

**PETITION FOR *INTER PARTES* REVIEW OF
U.S. PATENT NO. 6,889,244**

TABLE OF CONTENTS

	Page
I. INTRODUCTION	19
II. COMPLIANCE WITH IPR REQUIREMENTS.....	19
A. Grounds for Standing	19
B. Mandatory Notices	19
III. CHALLENGED CLAIMS	22
IV. THE '244 PATENT.....	22
A. Effective Filing Date	22
B. Person of Ordinary Skill in the Art	22
C. Overview of the '244 Patent.....	23
V. CLAIM CONSTRUCTION	25
VI. PRINCIPAL PRIOR ART	26
A. Summary of <u>Bowman</u>	26
B. Summary of <u>Vahalia</u>	26
C. Summary of <u>Tuxedo</u>	27
D. Motivation to Combine	27
1. <u>Bowman</u> / <u>Vahalia</u> combination	27
2. <u>Bowman</u> / <u>Vahalia</u> / <u>Tuxedo</u> combination.....	35
VII. CLAIMS 1–10 ARE UNPATENTABLE.....	38
A. Ground 1: Claim 1 Would Have Been Obvious Based on <u>Bowman</u> in view of <u>Vahalia</u>	38
1. Claim 1	38
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:.....	38
a. <i>A method of transmitting messages between a first node and a second node</i>	39
b. <i>wherein the first and second nodes are each coupled to a fault tolerant storage system (FTSS)</i>	40

ii.	[1.1] transmitting a message from the first node to a communication agent in the FTSS;.....	42
iii.	[1.2] storing the message in a data structure in highly reliable fault-tolerant storage media of the FTSS;	45
a.	<i>storing the message in a data structure</i>	45
b.	<i>in highly reliable fault-tolerant storage media of the FTSS</i>	46
iv.	[1.3] processing the message at the FTSS in accordance with a messaging paradigm; and	48
a.	<i>processing the message ... in accordance with a messaging paradigm</i> 48	
b.	<i>at the FTSS</i>	49
v.	[1.4] transmitting the message from the FTSS to the second node.....	50
a.	<i>transmitting the message ... to the second node</i>	50
b.	<i>from the FTSS</i>	50
B.	Ground 2: Claims 2-10 are Obvious Over <u>Bowman</u> in view of <u>Vahalia</u> and in Further View of <u>Tuxedo</u>	52
1.	Claim 2	52
i.	[2.0] The method of claim 1 wherein the messaging paradigm is a request/reply communication paradigm, and:	52
ii.	[2.1] transmitting a message from the first node to a communication agent in the FTSS comprises:	52
iii.	[2.2] transmitting a request message from the first node to a request/reply communication agent in the FTSS;	53
iv.	[2.3] storing the message in a data structure in highly reliable fault-tolerant storage media of the FTSS comprises:.....	53
v.	[2.4] storing the request message in a request queue in highly reliable fault-tolerant storage media of the FTSS;	53

vi.	[2.5] processing the message at the FTSS in accordance with a messaging paradigm comprises:	55
vii.	[2.6] retrieving the request message from the request queue; and	55
viii.	[2.7] transmitting the message from the FTSS to the second node comprises:	56
ix.	[2.8] transmitting the request message from the FTSS to a service on the second node;	56
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	59
xi.	[2.9b] transmitting the reply message from the second node to the request/reply communication agent in the FTSS;	60
xii.	[2.9c] storing the reply message in a reply queue in highly reliable fault-tolerant storage media of the FTSS;	60
xiii.	[2.9d] retrieving the reply message from the reply queue; and	60
xiv.	[2.9e] transmitting the reply message from the FTSS to the first node.	61
2.	Claim 3	62
i.	[3.0] The method of claim 1 wherein the messaging paradigm is a conversational communication paradigm, and:	62
ii.	[3.1] transmitting a message from the first node to a communication agent in the FTSS comprises:	63
iii.	[3.2] transmitting a connect message from the first node to a conversational communication agent in the FTSS;	63
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;	64

v.	[3.3b] storing the message in a data structure in highly reliable fault-tolerant storage media of the FTSS comprises:.....	65
vi.	[3.3c] storing the connect message in the conversation queue; and	65
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:.....	65
3.	Claim 4.....	66
i.	[4.0] The method of claim 3 and further comprising:	66
ii.	[4.1] transmitting a send message from the first node to the conversational communication agent in the FTSS;	66
iii.	[4.2] storing the send message in the conversation queue;.....	66
iv.	[4.3] transmitting the send message to from the FTSS to a service on the second node;.....	67
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;.....	67
vi.	[4.5] transmitting the receive message from the second node to the conversational communication agent in the FTSS;	70
vii.	[4.6] storing the receive message in the conversation queue; and	70
viii.	[4.7] transmitting the receive message to the first node.....	71
4.	Claim 5.....	71
i.	[5.0] The method of claim 3 and further comprising:	71
ii.	[5.1] transmitting a disconnect message from the first node to the conversational communication agent in the FTSS;	71

iii.	[5.2] closing the conversation queue; and	72
iv.	[5.3] transmitting the disconnect message to the second node.	73
5.	Claim 6	73
i.	[6.0] The method of claim 1 wherein the messaging paradigm is an event-based communication paradigm, and:	73
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;	73
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;	75
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	76
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.....	76
6.	Claim 7	77
i.	[7.0] The method of claim 6 and further comprising:	77
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	77
iii.	[7.2] accessing the subscription manager to record that the second node has subscribed to receive post messages associated with the event.	79

7.	Claim 8	79
i.	[8.0] The method of claim 6 and further comprising:	79
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	80
iii.	[8.2] accessing the subscription manager to record that the second node should no longer receive post messages associated with the event.	80
8.	Claim 9	81
i.	[9.0] The method of claim 1 wherein the messaging paradigm is a queue-based communication paradigm, and:	81
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;	82
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;	85
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	86
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;	87
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;	88

vii.	[9.5b] transmitting the dequeue message from the second node to the queue-based communication agent in the FTSS;	89
viii.	[9.5c] storing the dequeue message in the queue space;	89
ix.	[9.5d] retrieving the dequeue message from the queue space; and	89
x.	[9.5e] transmitting the dequeue message from the FTSS to the first node.	90
9.	Claim 10	90
i.	[10.0] The method of claim 1 wherein the messaging paradigm is an unsolicited communication paradigm, and:	90
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;	91
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;	92
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	93
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.	93
VIII.	DISCRETIONARY DENIAL IS NOT APPROPRIATE	94
A.	The Board Should Not Deny Institution Under 35 U.S.C. § 325(d)	94

B.	The Board Should Not Deny Institution Under 35 U.S.C. §	
	314(a).....	95
IX.	CONCLUSION.....	98

TABLE OF AUTHORITIES

	Page(s)
Cases	
<i>Apple, Inc. v. SEVEN Networks, LLC</i> , IPR2020-00156, Paper 10, 19 (June 15, 2020).....	79
<i>Samsung Elecs. Co. v. Dynamics Inc.</i> , IPR2020-00505, Paper 11, 15 (Aug. 12, 2020)	80
<i>Sand Revolution II, LLC v. Cont’l Intermodal Grp.-Trucking LLC</i> , IPR2019-01393, Paper 24, 12 (June 16, 2020).....	79
<i>SAP Am., Inc. v. Valtrus Innovations Ltd.</i> , 1:24-cv-00054-GBW (D. Del.).....	2
<i>SAP Am., Inc. v. Valtrus Innovations Ltd. et al.</i> , 1:24-cv-00756-GBW (D. Del.).....	2
<i>Valtrus Innovations Ltd. v. SAP Am., Inc. et al.</i> , 2:24-cv-00021-JRG (E.D. Tex.)	2
<i>Valtrus Innovations Ltd. v. SAP Am., Inc. et al.</i> , 2:24-cv-00533 (E.D. Tex.).....	2
<i>Vivid Techs., Inc. v. Am. Sci. & Eng'g Inc.</i> , 200 F.3d 795 (Fed. Cir. 1999).....	8
Statutes	
35 U.S.C. § 102(b)	9
35 U.S.C. § 102(e)	8
35 U.S.C. § 103	1, 4
35 U.S.C. § 314(a)	77
35 U.S.C. § 325(d)	76

Other Authorities

37 C.F.R. § 42.100(b).....	7
37 C.F.R. § 42.104(a).....	1
U.S. Const. art. III.....	77, 78

CLAIM LISTING

Claim 1	
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:
1.1	transmitting a message from the first node to a communication agent in the FTSS;
1.2	storing the message in a data structure in highly reliable fault-tolerant storage media of the FTSS;
1.3	processing the message at the FTSS in accordance with a messaging paradigm; and
1.4	transmitting the message from the FTSS to the second node.
Claim 2	
2.0	The method of claim 1 wherein the messaging paradigm is a request/reply communication paradigm, and:
2.1	transmitting a message from the first node to a communication agent in the FTSS comprises:
2.2	transmitting a request message from the first node to a request/reply communication agent in the FTSS;
2.3	storing the message in a data structure in highly reliable fault-tolerant storage media of the FTSS comprises:
2.4	storing the request message in a request queue in highly reliable fault-tolerant storage media of the FTSS;
2.5	processing the message at the FTSS in accordance with a messaging paradigm comprises:
2.6	retrieving the request message from the request queue; and

2.7	transmitting the message from the FTSS to the second node comprises:
2.8	transmitting the request message from the FTSS to a service on the second node;
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;
2.9b	transmitting the reply message from the second node to the request/reply communication agent in the FTSS;
2.9c	storing the reply message in a reply queue in highly reliable fault-tolerant storage media of the FTSS;
2.9d	retrieving the reply message from the reply queue; and
2.9e	transmitting the reply message from the FTSS to the first node.
Claim 3	
3.0	The method of claim 1 wherein the messaging paradigm is a conversational communication paradigm, and:
3.1	transmitting a message from the first node to a communication agent in the FTSS comprises:
3.2	transmitting a connect message from the first node to a conversational communication agent in the FTSS;
3.3a	processing the message at the FTSS in accordance with a messaging paradigm comprises:
3.3b	opening a conversation queue in highly reliable fault-tolerant storage media of the FTSS;
3.3c	storing the message in a data structure in highly reliable fault-tolerant storage media of the FTSS comprises:
3.3d	storing the connect message in the conversation queue; and

3.4	transmitting the message from the FTSS to the second node comprises: transmitting the connect message from the FTSS to the second node.
Claim 4	
4.0	The method of claim 3 and further comprising:
4.1	transmitting a send message from the first node to the conversational communication agent in the FTSS;
4.2	storing the send message in the conversation queue;
4.3	transmitting the send message to from the FTSS to a service on the second node;
4.4	executing the service in the second node using the send message as an input and providing a receive message as an output;
4.5	transmitting the receive message from the second node to the conversational communication agent in the FTSS;
4.6	storing the receive message in the conversation queue; and
4.7	transmitting the receive message to the first node.
Claim 5	
5.0	The method of claim 3 and further comprising:
5.1	transmitting a disconnect message from the first node to the conversational communication agent in the FTSS;
5.2	closing the conversation queue; and
5.3	transmitting the disconnect message to the second node.
Claim 6	
6.0	The method of claim 1 wherein the messaging paradigm is an event-based communication paradigm, and:

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;
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;
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
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.
Claim 7	
7.0	The method of claim 6 and further comprising:
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
7.2	accessing the subscription manager to record that the second node has subscribed to receive post messages associated with the event.
Claim 8	
8.0	The method of claim 6 and further comprising:
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

8.2	accessing the subscription manager to record that the second node should no longer receive post messages associated with the event.
Claim 9	
9.0	The method of claim 1 wherein the messaging paradigm is a queue-based communication paradigm, and:
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;
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;
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
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;
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;
9.5b	transmitting the dequeue message from the second node to the queue-based communication agent in the FTSS;
9.5c	storing the dequeue message in the queue space;
9.5d	retrieving the dequeue message from the queue space; and

9.5e	transmitting the dequeue message from the FTSS to the first node.
Claim 10	
10.0	The method of claim 1 wherein the messaging paradigm is an unsolicited communication paradigm, and:
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;
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;
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
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.

PETITIONER’S EXHIBIT LIST

No.	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. INTRODUCTION

Petitioner SAP America, Inc. submits this Petition for *Inter Partes* Review of claims 1-10 (the “Challenged Claims”) of U.S. Patent No. 6,889,244 (the “’244 Patent” (Ex-1101)), assigned to Patent Owners, Valtrus Innovations Ltd. and Key Patent Innovations Ltd. Petitioner respectfully submits that the Challenged Claims of the ’244 Patent are unpatentable under 35 U.S.C. § 103 in view of the prior art references discussed herein. Accordingly, it is respectfully requested that the Board institute an *inter partes* review of the ’244 Patent.

II. COMPLIANCE WITH IPR REQUIREMENTS

A. Grounds for Standing

As required by 37 C.F.R. § 42.104(a), Petitioner certifies that the ’244 Patent “is available for inter partes review and that the Petitioner is not barred or estopped from requesting an inter partes review challenging the patent claims on the grounds identified in the petition.” Neither Petitioner nor any real party in interest has filed a civil action challenging the validity of a claim of the ’244 Patent. This petition is being filed no more than one year after the date on which Petitioner or real parties in interest or privies of Petitioner were served with a complaint alleging infringement of the ’244 Patent.

B. Mandatory Notices

Real Party-in-Interest: Petitioner identifies SAP America, Inc. and SAP SE (collectively, “SAP”) as real parties-in-interest.

Related Matters: No previous IPR or post-grant proceeding has been filed challenging the '244 Patent. Valtrus has asserted the '244 Patent in litigation against SAP in *Valtrus Innovations Ltd. v. SAP America, Inc. et al*, 2:24-cv-00021-JRG (E.D. Tex.); Patent Owners have asserted the '244 Patent in litigation against SAP in *Valtrus Innovations Ltd. v. SAP America, Inc. et al*, 2:24-cv-00533 (E.D. Tex.); SAP filed a declaratory judgment action of non-infringement of the '244 Patent against Patent Owners and Patent Platform Services, LLC in *SAP America, Inc. v. Valtrus Innovations Limited et al*, 1:24-cv-00756-GBW (D. Del.); and SAP filed a declaratory judgment action of non-infringement of the '244 Patent against Valtrus Innovation Ltd. in *SAP America, Inc. v. Valtrus Innovations Limited*, 1:24-cv-00054-GBW (D. Del.).

Lead and Back-up Counsel:

Lead Counsel	Back-up Counsel
Louis L. Campbell (Reg. # 59,963) WINSTON & STRAWN LLP 255 Shoreline Drive, Suite 520 Redwood City, CA 94065 Tel: (650) 858-6500 LLCampbell@winston.com	Scott M. Border (Reg. # 77,744) WINSTON & STRAWN LLP 1901 L Street NW Washington, DC 20036 Tel: (202) 282-5000 SBorder@winston.com
	Samuel W. Riebe (Reg. # 78,044) WINSTON & STRAWN LLP 2121 N. Pearl Street, Suite 900 Dallas, TX 75201 Tel: (214) 453-6500 SRiebe@winston.com

Lead Counsel	Back-up Counsel
	Michael Bittner (<i>Pro hac vice</i> to be submitted) WINSTON & STRAWN LLP 2121 N. Pearl Street, Suite 900 Dallas, TX 75201 Tel: (214) 453-6500 MBittner@winston.com
	Matt R. McCullough (<i>Pro hac vice</i> to be submitted) WINSTON & STRAWN LLP <i>Pro hac vice</i> to be submitted Winston & Strawn LLP 255 Shoreline Drive, Suite 520 Redwood City, CA 94065 Tel: (650) 858-6500 MRMcCullough@winston.com
	William Logan (<i>Pro hac vice</i> to be submitted) <i>Pro hac vice</i> to be submitted WINSTON & STRAWN LLP 800 Capitol Street, Suite 2400 Houston, TX 77002 Tel: (713) 651-2600 WLogan@winston.com
	Patrick Clark (<i>Pro hac vice</i> to be submitted) WINSTON & STRAWN LLP <i>Pro hac vice</i> to be submitted 2121 N. Pearl Street, Suite 900 Dallas, TX 75201 Tel: (214) 453-6500 PClark@winston.com

Service Information: Service information is provided in the designation of counsel above. Petitioner consents to service of all documents via electronic mail at IPR-SAP-Val@winston.com.

III. CHALLENGED CLAIMS

Ground 1: Claim 1 is unpatentable as obvious under 35 U.S.C. § 103 based on Bowman (Ex-1105) in view of Vahalia (Ex-1106).

Ground 2: Claims 2-10 are unpatentable as obvious under 35 U.S.C. § 103 based on Bowman (Ex-1105) in view of Vahalia (Ex-1106) and in further view of Tuxedo (Ex-1107).

IV. THE '244 PATENT

A. Effective Filing Date

The '244 Patent issued from an application filed October 31, 2000, and does not claim priority to any other application. Petitioner assumes for the purposes of this Petition that **October 31, 2000**, is the effective filing date.

B. Person of Ordinary Skill in the Art

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 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.

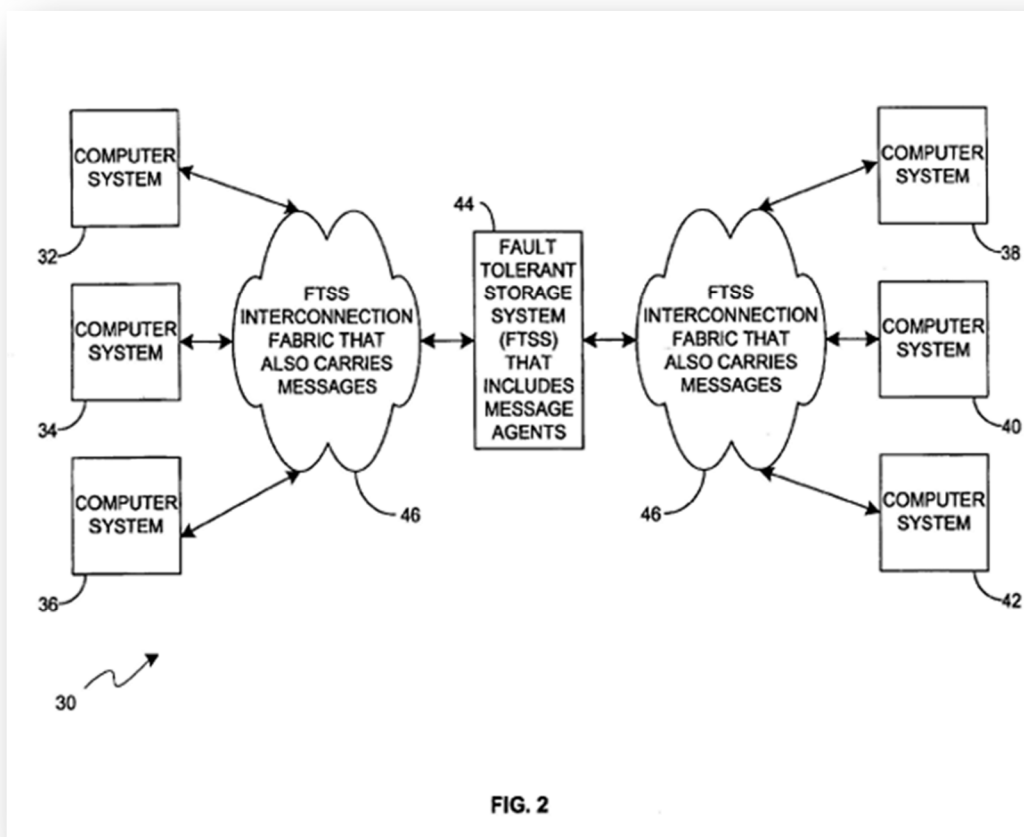
C. Overview of the '244 Patent

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.

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.

Figure 2 depicts an embodiment of the invention in the '244 Patent:



Ex-1101, Fig. 2 (annotated).

Figure 2 includes “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.

V. CLAIM CONSTRUCTION

Claims are given their “ordinary and customary meaning” as understood by a POSITA and the prosecution history pertaining to the patent. 37 C.F.R. § 42.100(b). 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, 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, 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).

Otherwise, because a POSITA would find the challenged claims unpatentable under any interpretation consistent with their plain and ordinary meaning in the context of the ’244 Patent, the Board need not expressly construe the claim terms. *See Vivid Techs., Inc. v. Am. Sci. & Eng. ’g. Inc.*, 200 F.3d 795, 803 (Fed. Cir. 1999).

VI. PRINCIPAL PRIOR ART

A. Summary of Bowman

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).

B. Summary of Vahalia

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 35 U.S.C. § 102(e).

C. Summary of Tuxedo

The TuxedoTM 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). Tuxedo was disseminated or otherwise made available to the extent that persons interested and ordinarily skilled in the art, exercising reasonable diligence, could have located it before October 31, 2000. Ex-1109, ¶¶ 43-52.

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.

D. Motivation to Combine

For the reasons below, it would have been obvious to a POSITA to combine Bowman, Vahalia, and Tuxedo. Ex-1103, ¶ 65.

1. Bowman / Vahalia combination

A POSITA would have found it obvious to use disk arrays comprising a communication agent in its fault tolerant file sharing, as taught by Vahalia. Ex-1105; Ex-1107; Ex-1103, ¶ 66.

Bowman teaches a system for efficiently retrieving data in which data from a server is requested over a network and sent to the client over the network. Ex-1105, Abstract. A high-level overview is shown in Figure 10:

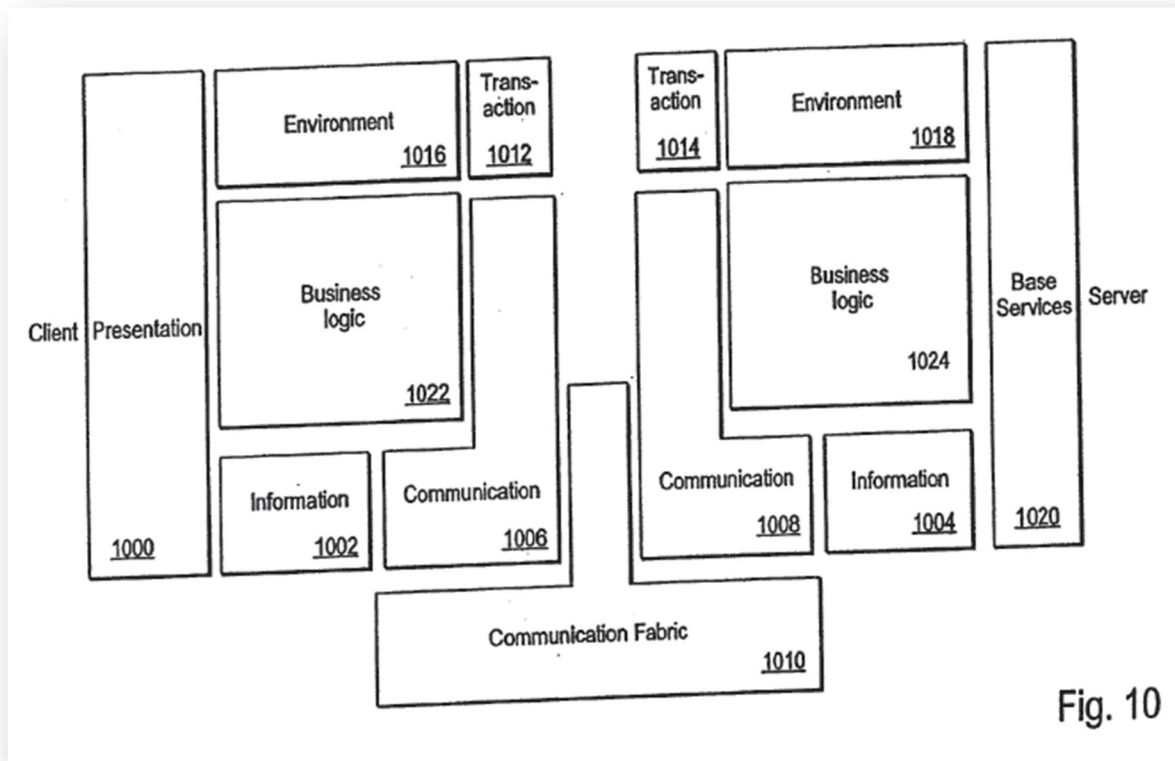


Fig. 10

Ex-1105, Fig. 10.

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.

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.

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.

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.

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 protect against a single point of failure. Ex-1103, ¶ 73;

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 understood RAID (Redundant Arrays of Independent Disks) to describe disk arrays 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-1103, ¶ 73; 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.

A POSITA would have been motivated to look to Vahalia to provide additional details about implementing the disk arrays taught or suggested in Bowman. Ex-1103, ¶ 74. Vahalia teaches a data storage system that uses “an array of low-cost disk drives.” Ex-1106, 1:9-15. Vahalia also teaches that its disk array is “interfaced to a data network.” Ex-1106, 1:20-21. Ex-110, ¶ 76. Annotated

Figure 3 of Vahalia illustrates this, showing “Cached Disk Array #2” (red) in File Server 60 (yellow):

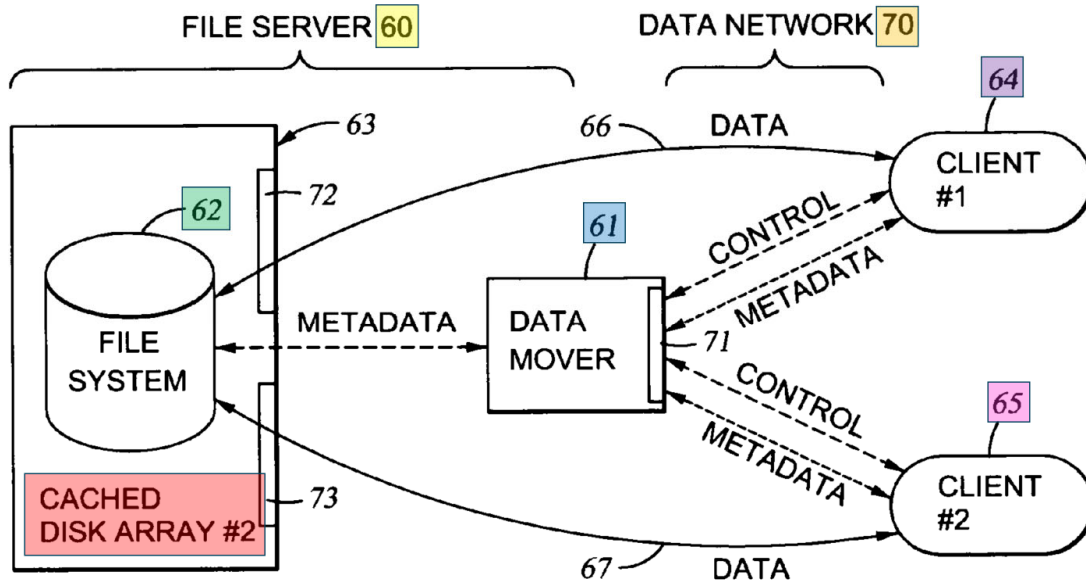


Fig. 3

Ex-1103, ¶ 76. A POSITA would have understood File Server 60 in Vahalia comprising Cached Disk Array #2 is a fault-tolerant storage system (FTSS). Ex-1103, ¶ 122 Ex-1108, 28.

Vahalia further teaches that Data over 61 (blue) moves data to and from Cached Disk Array #2 to clients (e.g., Client #1, 64 (purple) and Client # 2, 65 (pink)) over Data Network 70. Ex-1106, 35:47-55. A POSITA would have understood Data Mover to be a communication agent, like Bowman’s MOM. Ex-1103, ¶ 78. Vahalia discloses that Data Mover 61 is on File Server 60 (the FTSS). Ex-1106, 35:47-55.

A POSITA would have understood that Vahalia's Data Network 70 could be implemented over Bowman's Communications Fabric 1010. Ex-1103, ¶ 78; 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).

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; Ex-1103, ¶ 79.

Both Bowman and Vahalia are analogous art to the '244 Patent. Bowman and Vahalia are directed to a similar field of endeavor as the '244 Patent—e.g., 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 efficient data storage systems in relation to network file servers. Ex-1106, 1:9-10.

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. Vahalia teaches that “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.” Ex-1106, 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.*, 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.

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. Ex-1103, ¶ 83. Immediate similarities to Bowman are apparent when Figure 10 of Bowman is compared to Figure 3 of Vahalia.

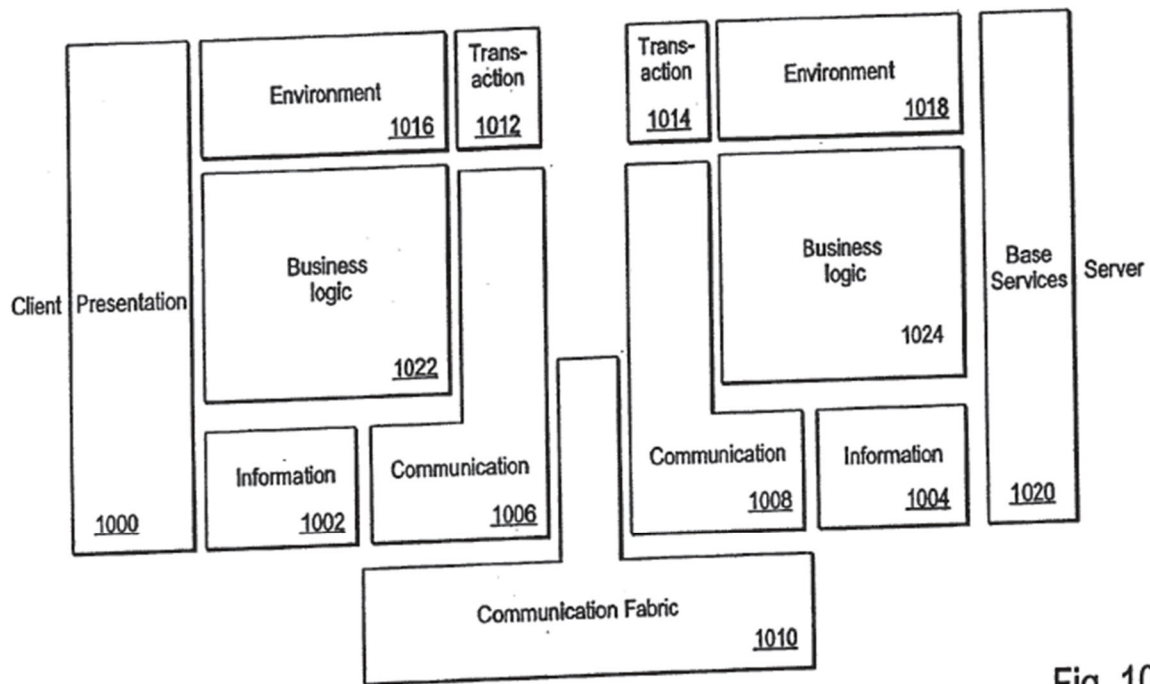


Fig. 10

Ex-1105, Fig. 10.

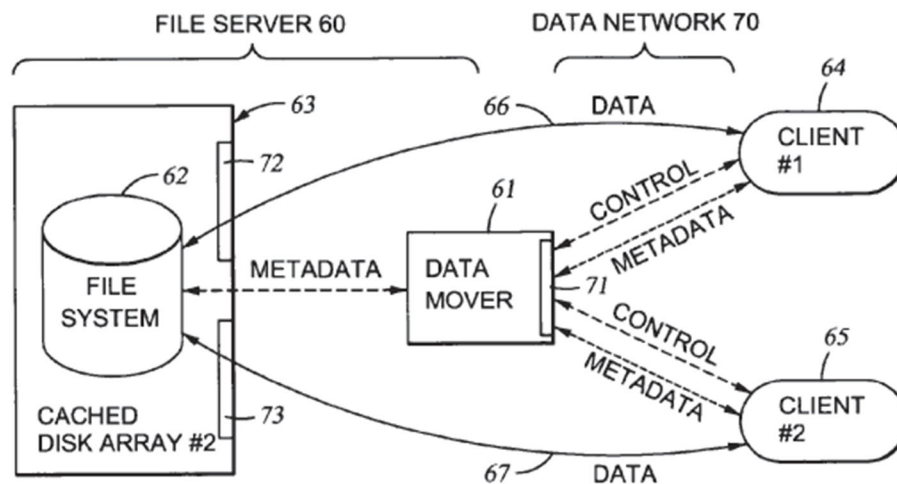


Fig. 3

Ex-1106, Fig. 3. 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). As demonstrated above, Bowman teaches implementing a disk array to create a fault tolerant file sharing system, and a POSITA would have expected to succeed in accomplishing fault tolerance by implementing Vahalia’s File Server 60 that comprises Cached Disk Array #2 and Data Mover 61. Ex-1103, ¶ 84.

2. Bowman / Vahalia / Tuxedo combination

A POSITA implementing the Bowman / Vahalia system (including incorporating Vahalia’s teaching of a data mover on file server as middleware), would have been motivated to use Tuxedo’s messaging paradigms.

Bowman discloses using 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.” Ex-1105, 70:8-12. A POSITA would have understood that these forms of messaging in Bowman are messaging paradigms—also referred to as “communications paradigms.” Ex-1103, ¶¶ 90-91; Ex-1107, 27 (“A paradigm is a style, or form, of doing something. Communication paradigms are styles of communication.”), 131

(“a chapter is devoted to each of the four communications paradigms: request/response, conversations, application queuing, and events.”).

Bowman explicitly discloses 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.”).

Tuxedo teaches implementing such messaging paradigms (including message queuing and publish and subscribe) at the time of the '244 Patent. Ex-1103, ¶¶ 88-90; 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.”). Given Bowman’s express disclosure of implementing Tuxedo, a POSITA would have been motivated to examine those teachings for additional detail regarding the

implementation of Tuxedo and combine the teachings of those publications with Bowman's teachings. Ex-1103, ¶ 91.

A POSITA would have further understood to implement the message paradigm teachings from Tuxedo on file server 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 teachings in Bowman and Vahalia and the file server that comprises the FTSS teaching in Vahalia). Ex-1103, ¶ 93; 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.”).

A POSITA would be 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 MOM) that yield predictable results (using Tuxedo's

messaging paradigms teachings with Vahalia's file server teachings in the Bowman system teachings). Ex-1105; Ex-1106; Ex-1107; Ex-1103, ¶ 93.

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. And a POSITA would have expected to succeed in using Tuxedo's messaging paradigms in the Bowman/Vahalia system due to Bowman's explicit disclosure of using Tuxedo. Ex-1105, 57:9-18, 34:18-20, 95:15-16; Ex-1103, ¶ 93.

VII. CLAIMS 1–10 ARE UNPATENTABLE

A. Ground 1: Claim 1 Would Have Been 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*

Bowman teaches “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):

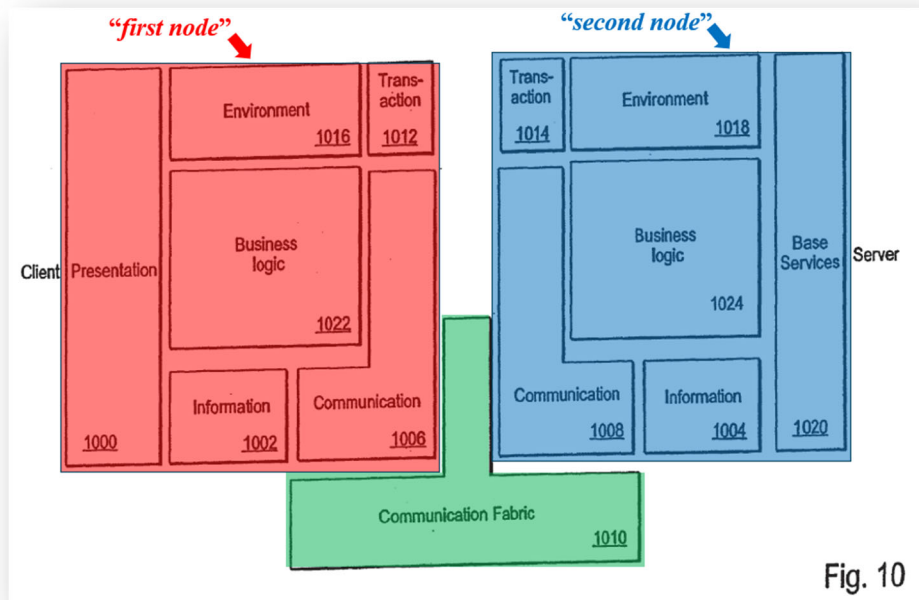


Fig. 10

Ex-1105, Fig. 10 (annotated). The client is a “*first node*” and the server is a “*second node*,” consistent with those terms in the ’244 Patent, as both are computer systems coupled to a network. Ex-1103, ¶ 95; Ex-1105, 33:5-8, 1:61-64.

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. Communications Fabric 1010 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 of transmitting messages*”). Ex-1105, 84:35-44, 84:49-16. As Dr. Almeroth explains, “transferring messages” over Communications Fabric 1010 comprises “*transmitting messages*” between the client (“*first node*”) and server (“*second node*”). Ex-1103, ¶ 96; Ex-1105, 84:35-44, 84:49-16.

Bowman thus discloses “*a method for transmitting messages*” (e.g., messaging over Communications Fabric 1010) “*between a first node*” (e.g., Server in Figure 10) and “*a second node*” (e.g., client in Figure 10). Ex-1103, ¶ 97.

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

Bowman teaches or suggests “*the first and second nodes*” (e.g., Bowman’s server and client) “*are each coupled to a fault tolerant storage system (FTSS).*”

For the reasons discussed regarding the motivation to combine Bowman and Vahalia, Bowman teaches using disk arrays (e.g., disk mirroring) as fault tolerant storage, and a POSITA would have been motivated to look to a reference like Vahalia for more implementation details. *See* §VI.D.1.

Consistent with Patent Owner’s position that “fault tolerant” will “generally describe systems with additional indications or guarantees of reliability beyond non-redundant, single point of failure systems” (Ex-1110, ¶ 50), Dr. Almeroth explains that Bowman’s disclosure of using disk arrays to store data in a fault-tolerant manner teaches or suggests a “*fault tolerant storage system (FTSS).*”

Ex-1103, ¶ 101; Ex-1108, 28; Ex-1114, 144 (listing “*see also* fault tolerance” for “disk mirroring”); Ex-1112, 464.

For example, Vahalia discloses further details about implementing networked disk array. Ex-1106, 1:20-21. Vahalia’s Figure 3 shows “Cached Disk Array #2” (red) as part of File Server 60 (yellow):

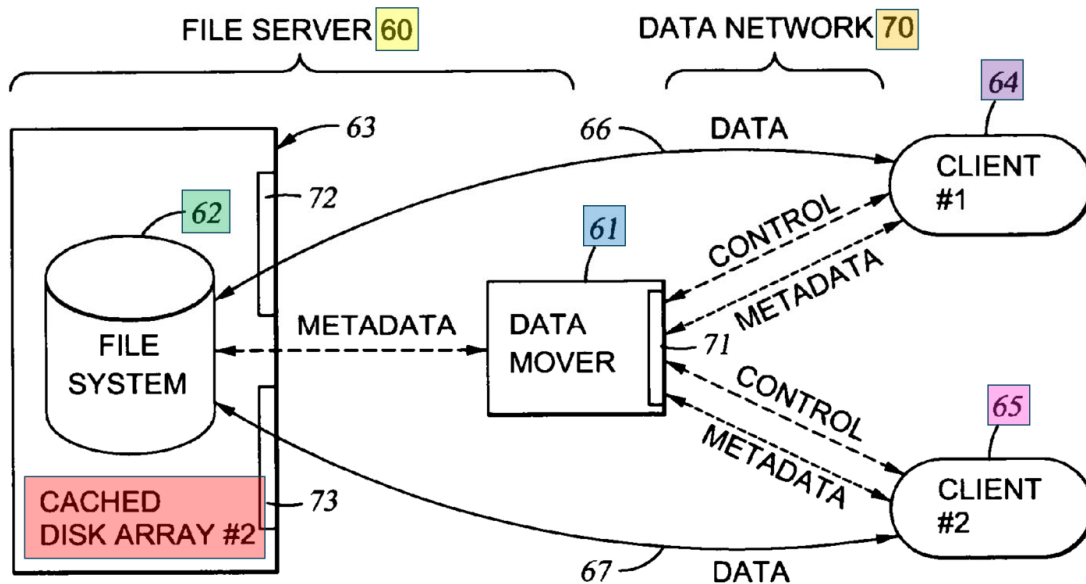


Fig. 3

Ex-1103, ¶ 105. Dr. Almeroth thus explains that a POSITA would have understood File Server 60 in Vahalia comprising Cached Disk Array #2 is a “*fault-tolerant storage system (FTSS)*.” Ex-1103, ¶ 106 (citing Ex-1108, 28).

Vahalia further teaches 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. Ex-1106, 35:47-55.

Client #1 and Client #2 in Figure 3 are nodes (e.g., “*a first node* and *a second node*”). Ex-1103, ¶ 109; Ex-1106, 36:11-15 (describing clients consistent with general purpose computers).

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, at least under Patent Owner’s view of “coupled to.”¹ Ex-1103, ¶ 109; Ex-1106, 8:56; Ex-1110, ¶ 50.

Accordingly, it would have been obvious based on Bowman and Vahalia to implement an arrangement where “*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;***

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

¹ As explained above, Petitioner has adopted Patent Owner’s position that the scope of “*coupled to*” includes “communicatively connected” (e.g., over a conventional computer network). Ex-1110, ¶ 50.

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.

A POSITA would have understood MOM to be a “*communication agent*” because it is software that passes data and control messages on behalf of clients and servers. Ex-1103, ¶ 112; Ex-1105, 69:53-65; Ex-1112, 139 (defining “middleware”); Ex-1107, 413 (defining “agent”); Ex-1113, 500.

Vahalia teaches how to implement communication agents (like Bowman’s MOM) “*in the FTSS*.” 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):

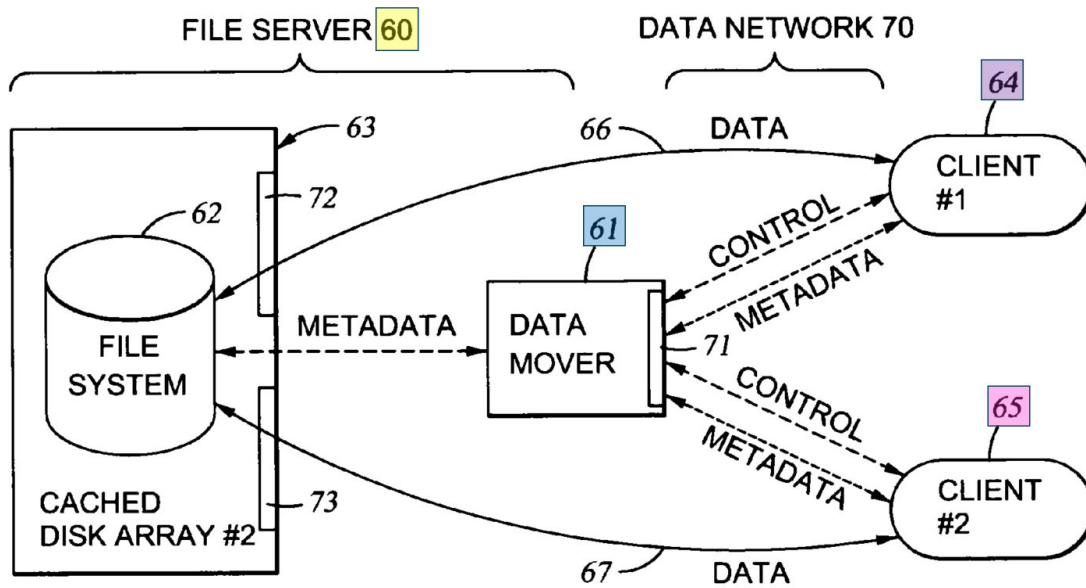


Fig. 3

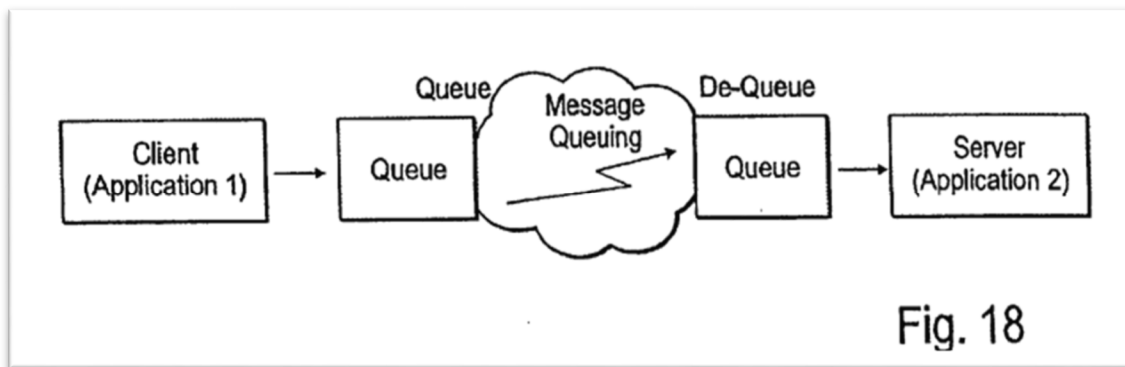
Ex-1103, ¶ 114; Ex-1106, 35:25-68. Dr. Almeroth explains that these data access requests are sent as network messages from one node (e.g., a client) to another (e.g., a server). Ex-1103, ¶ 115; Ex-1106, 2:17-24, 14:32-37, 36:52-55. A POSITA would have thus understood Vahalia's Data Mover to be a "*communication agent*." Ex-1103, ¶ 115; Ex-1113, 500; Ex-1107, 413 (defining "agent").

Bowman and Vahalia therefore teach "*transmitting a message*" (e.g., sending a file access request as in Vahalia) "*from the first node*" (e.g., the client in Bowman) "*to a communication agent*" (e.g., to Data Mover 61 of Vahalia) "*in the FTSS*" (e.g., the File Server 60 in Vahalia that includes Data Mover 61), as Dr. Almeroth explains. Ex-1103, ¶ 116.

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*

Bowman discloses that “[t]here are three types of Message-Oriented Middleware commonly implemented: Message Passing[,] **Message Queuing**[,] Publish and Subscribe.” Ex-1105, 70:5-12. Figure 18 shows message queueing:



Ex-1105, Figure 18, 70:23-54 (describing message queueing).

A POSITA would have understood message queueing as involving storing the message in a data structure (e.g., a queue) in memory. Ex-1103, ¶ 119; Ex-1107, 41, 49, 425 (defining “queue” as “a memory data structure to hold messages”).

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). Ex-1103, ¶ 120.

b. *in highly reliable fault-tolerant storage media of the FTSS*

In addition to teaching storing the message in a data structure, Bowman teaches the need for “reliable, guaranteed and assured (non-duplicate) message delivery” (Ex-1105, 70:53-54) and the “use of primary and replica file servers to ensure high availability of file system” (Ex-1105, 59:7-8). Including for the reasons discussed for limitation [1.0], these disclosures would have taught or suggested to a POSITA using the cached disk array in File Server 60 from Vahalia as fault-tolerant storage media—here, for storing the data structures in which messages are stored. Ex-1103, ¶ 121; §VI.D.1.

Vahalia also teaches placing file access requests in a collector buffer or queue 276. *See, e.g.*, Ex-1106, 23:19-24, 23:46-50, 26:45-50. 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. Ex-1103, ¶ 122. Vahalia shows Data Mover (blue) comprises collector or queue 276 (brown):

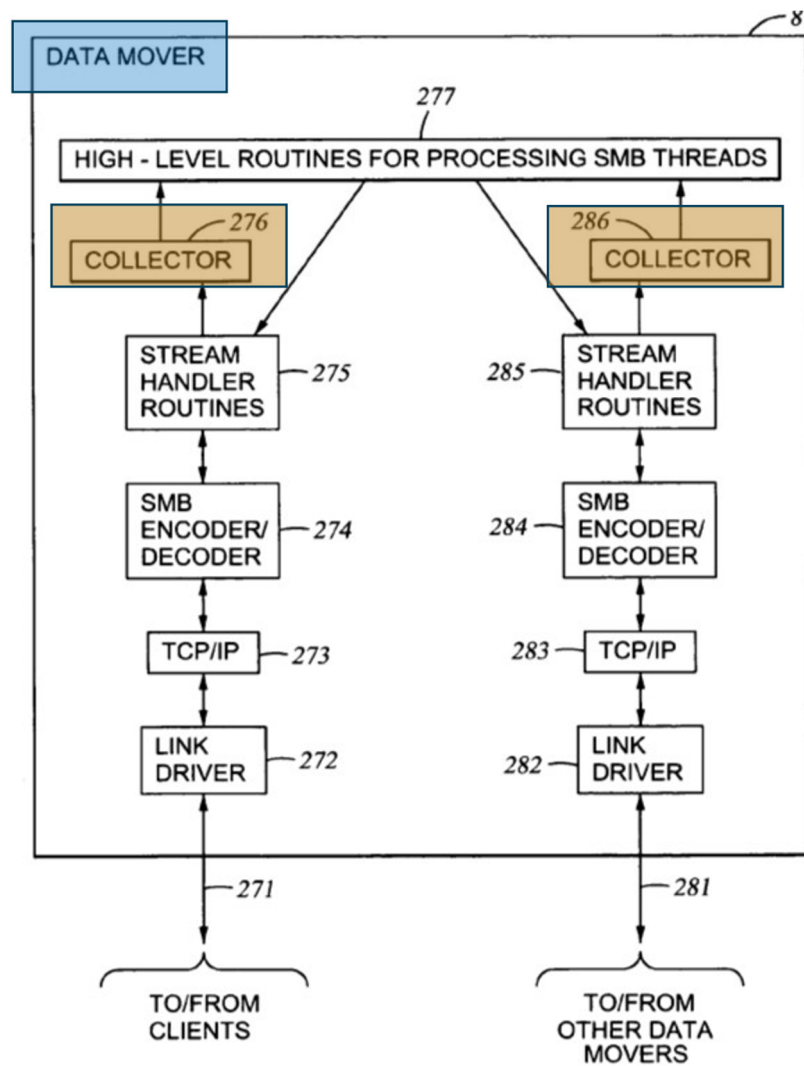


Fig. 16

Ex-1103, ¶ 123. As demonstrated for limitation [1.1], Data Mover is the communication agent of Vahalia's File Server 60, which comprises the FTSS.

Ex-1103, ¶ 124.

Bowman in view of Vahalia thus 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). Ex-1103, ¶ 125.

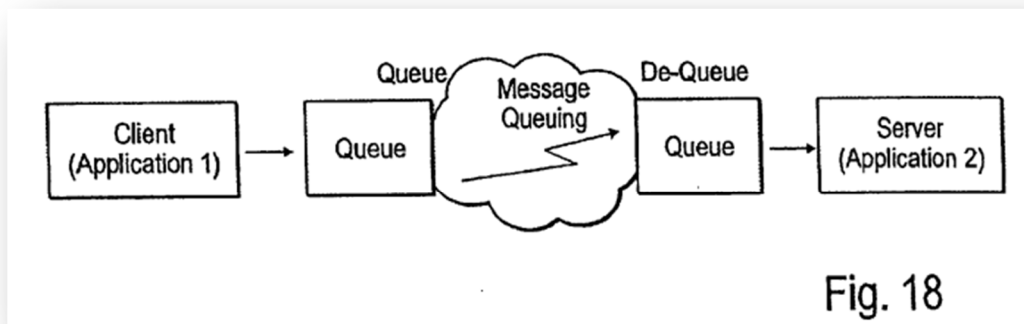
- iv. ***[1.3] processing the message at the FTSS in accordance with a messaging paradigm; and***
 - a. ***processing the message ... in accordance with a messaging paradigm***

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." Ex-1105, 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, 69:59-67. A POSITA would have understood these disclosures (e.g., describing how messages should be exchanged between applications) to describe "*processing the message*." Ex-1103, ¶ 128; Ex-1105, 69:59-67.

Bowman also discloses that "[m]essage [q]ueuing ... allows applications to communicate via message queues" (Ex-1105, 70:23-29), as shown in Figure 18:



Ex-1105, Fig. 18. A POSITA would have understood that message queuing is a messaging paradigm. Ex-1103, ¶ 130; Ex-1107, 27 (“A paradigm is a style, or form, of doing something. Communication paradigms are styles of communication.”), 131. This is consistent with the ’244 Patent, which describes queue-based communication as “a messaging paradigm” known in the prior art. Ex-1101, Fig. 3 (describing prior art); Ex-1103, ¶ 130.

b. at the FTSS

Bowman in view of Vahalia would have rendered obvious processing the message in accordance with a messaging paradigm “at the FTSS.”

As demonstrated for limitation [1.1], the communication agent is in the FTSS, and as discussed for limitation [1.2], the message is stored on the FTSS. A POSITA would have thus understood the combination of Bowman and Vahalia to teach processing the message “at the FTSS.” Ex-1103, ¶ 134.

Bowman and Vahalia thus teach this limitation.

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

a. *transmitting the message ... to the second node*

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

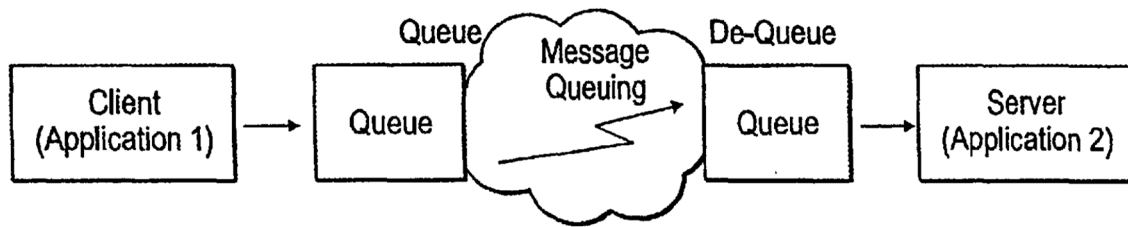


Fig. 18

Ex-1105, Fig. 19. Dr. Almeroth explains that message queuing in Bowman involves 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-1103, ¶ 138; Ex-1105, 70:22-29.

b. *from the FTSS*

As demonstrated for limitation [1.1], Bowman and Vahalia teach a communication agent in the FTSS (Ex-1103, ¶ 142), and as described above, Bowman teaches the communication agent (MOM) transmitting the message to the second node. Ex-1103, ¶ 142.

A POSITA would thus have understood the combination of Bowman and Vahalia to teach or suggest “*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):

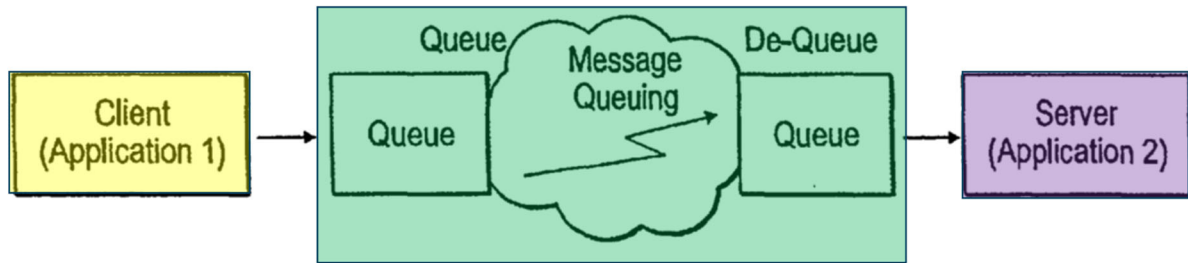


Fig. 18

Ex-1103, ¶ 142. For example, as demonstrated in limitations [1.2] and [1.3], 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) (“*transmitting the message from the FTSS to the second node.*”) Ex-1103, ¶ 143; Ex-1105, 70:22-29.

The combination of Bowman and Vahalia thus teaches this limitation and every limitation in claim 1.

B. Ground 2: Claims 2-10 are Obvious Over 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:*

As explained above, Bowman and Vahalia disclose “messaging paradigm[s]” and Tuxedo teaches implementing them. See §VI.D.2.

Among the messaging paradigms that Tuxedo teaches is a “request/response paradigm” (Ex-1107, xxxiv, 189), which is “a request/reply communication paradigm.” Ex-1103, ¶ 146 (citing Ex-1107, 30 (“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”).

Bowman, Vahalia, and Tuxedo thus disclose this limitation.

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

As demonstrated for limitation [1.1], Bowman and Vahalia teach this claim limitation. Ex-1103, ¶ 155.

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

As demonstrated for limitation [1.1], Bowman and Vahalia teach “transmitting a request message from the first node to ... a communication agent in the FTSS.” As demonstrated for limitation [2.0], Bowman, Vahalia, and Tuxedo teaches the “communication agent in the FTSS” (e.g., Vahalia’s Data Mover) using Tuxedo’s “request/reply communication paradigm.”

Dr. Almeroth states that when Vahalia’s “communication agent in the FTSS” implements Tuxedo’s “request/reply” communication paradigm, Vahalia’s communication agent is “a request/reply communication agent.” Ex-1103, ¶ 157 (citing Ex-1107, 413 (defining “agent”)).

Bowman, Vahalia, and Tuxedo thus teach this limitation.

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

As demonstrated for limitation [1.2], Bowman and Vahalia teach this limitation. Ex-1103, ¶ 158.

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

As demonstrated for limitation [1.2], Bowman teaches “storing the message in a data structure in highly reliable fault-tolerant storage media of the FTSS.”

Bowman teaches storing messages in a “*queue*,” as shown in Figure 18, which “allows applications to communicate via message queues, rather than by calling each other directly.” Ex-1105, 70:22-26; Ex-1103, ¶ 160.

Tuxedo also discloses the use of a message queue. Ex-1107, 40-41. Tuxedo further teaches that these queues can be used with the request/response paradigm. Ex-1107, 42-43. Tuxedo’s Figure 3.10 shows a request/reply paradigm including both a Request Queue (red) and a Reply Queue (green):

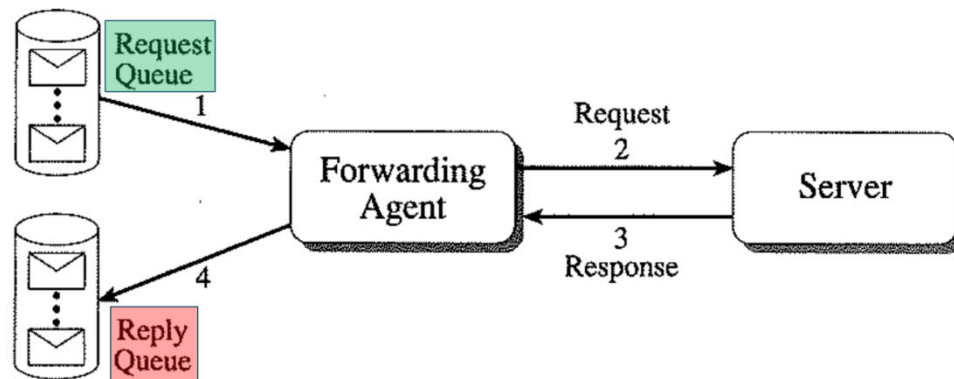


Figure 3.10 Forwarding Agent Converts Queue Operations to Request/Response

Ex-1107, 43; Ex-1103, ¶ 163. Figure 3.10 thus shows “*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-1103, ¶ 164; Ex-1107, 42-43, 425 (defining “request queue” as “the memory data structure that holds incoming requests for services”).

As demonstrated for limitation [1.2], Bowman and Vahalia teach storing messages in data structures (such as queues) in highly reliable fault-tolerant storage media of the FTSS, and thus a POSITA would have understood Bowman and Vahalia to teach storing the request queue there. Ex-1103, ¶¶ 165-66.

Bowman, Vahalia, and Tuxedo thus teach this limitation.

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

As demonstrated for limitation [1.3], Bowman and Vahalia teach this limitation. Ex-1103, ¶ 167.

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

Tuxedo explains that “a forwarding agent, 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. In Figure 3.10, Forwarding Agent (purple) reads Request Queue (green) at flow 1 (yellow):

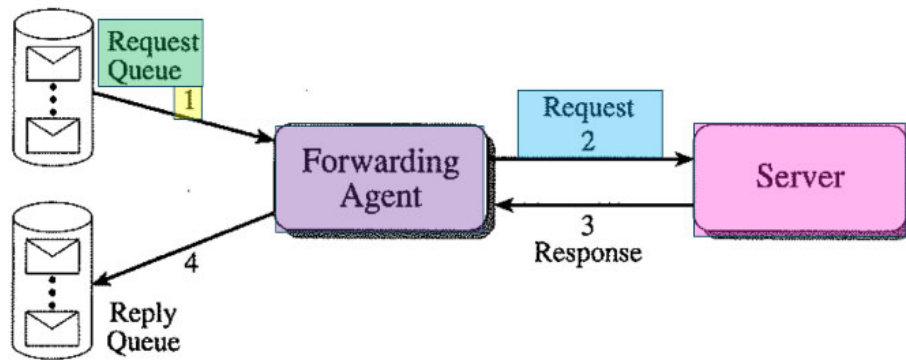


Figure 3.10 Forwarding Agent Converts Queue Operations to Request/Response

Ex-1107, Fig. 3.10. Flow 1 would involve retrieving from the Request Queue the Request that Forwarding Agent the transmits to Server (pink). Ex-1103, ¶ 169 (citing Ex-1107, 42-43).

Bowman, Vahalia, and Tuxedo thus disclose this limitation.

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

As demonstrated for limitation [1.1], Bowman and Vahalia teach this claim limitation. Ex-1103, ¶ 171.

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

As demonstrated for limitation [1.1], Bowman and Vahalia teach “transmitting the request message from the FTSS to ... the second node.”

Bowman teaches that “[i]n a client-server environment, a client makes requests of services on a server.” Ex-1105, 213:20-21. “Whenever a user needs to

use a service, the user must send a request across the network to the Server machine.” Ex-1105, 213:23-27. Dr. Almeroth explains this means that the first node (“client”) of Bowman 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-1103, ¶ 173.

Tuxedo also shows that the “*request message*” is transmitted “*to a service on the second node.*” Tuxedo teaches that clients request services from servers. Ex-1107, 28. This is illustrated in Tuxedo’s Figure 3.2, which shows client (yellow) transmitting Request Data (green) to Server (purple):

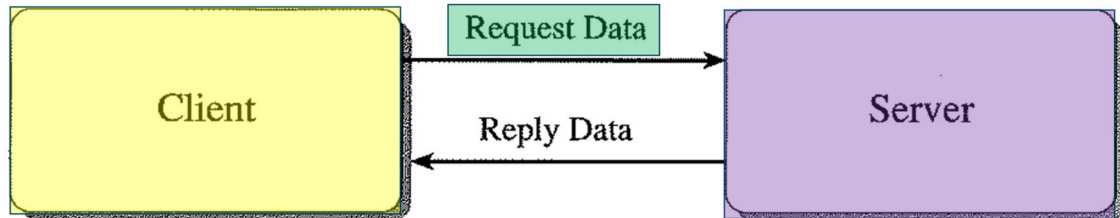


Figure 3.2 Request/Response Interaction

Ex-1107, 29.

The Server in Figure 3.2 is hosting a “*service*” that is receiving a “*request*” (Request Data) from client. Ex-1103, ¶ 176 (citing Ex-1107, 28). Tuxedo thus discloses “*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-1103, ¶ 176 (citing Ex-1107, 28).

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. Forwarding Agent (purple) reads Request Queue (green) at flow 1 (yellow):

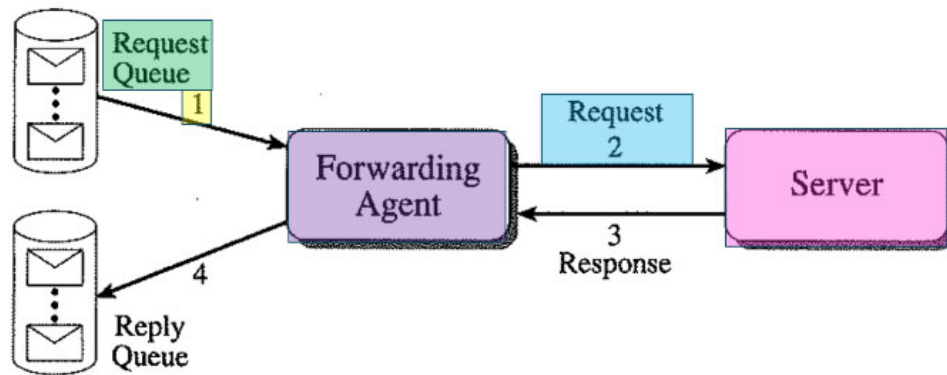


Figure 3.10 Forwarding Agent Converts Queue Operations to Request/Response

Ex-1107, 43. As demonstrated for limitation [2.4], the Request Queue is in the FTSS.

As demonstrated for limitation [1.1], the communication agent that facilitates the messaging is in the FTSS, and as discussed for limitations [1.2] and [2.4], the request message is stored in the FTSS. Thus, a POSITA would have understood that the request message is being transmitted “*from the FTSS*.”

Bowman, Vahalia, and Tuxedo thus disclose this limitation.

- 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*

As demonstrated for limitation [2.8], Bowman, Vahalia, and Tuxedo teach “the service on the second node.”

Bowman, Vahalia, and Tuxedo further teach “executing the service on the second node.” For example, Figure 3.10 of Tuxedo 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):

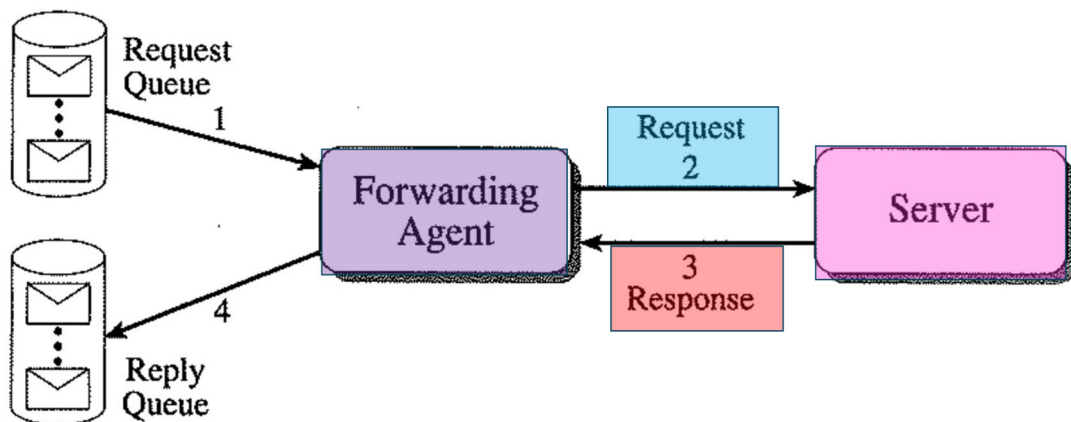


Figure 3.10 Forwarding Agent Converts Queue Operations to Request/Response

Ex-1107, 43. Tuxedo, therefore, teaches “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 the Server in reply to the Request). Ex-1103, ¶ 184 (citing Ex-1107, 42, 42-43); Ex-1114, 404 (defining “service”).

Bowman, Vahalia, and Tuxedo thus teach this limitation.

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

As demonstrated for limitations [1.1], [2.8], and [2.9a], Bowman, Vahalia, and Tuxedo disclose “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 Forwarding Agent from Tuxedo Figure 3.10). Ex-1103, ¶ 186.

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

As demonstrated for limitations [2.4] and [2.9a], Bowman, Vahalia, and Tuxedo disclose “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-1103, ¶ 187; Ex-1107, 42, 425.

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

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

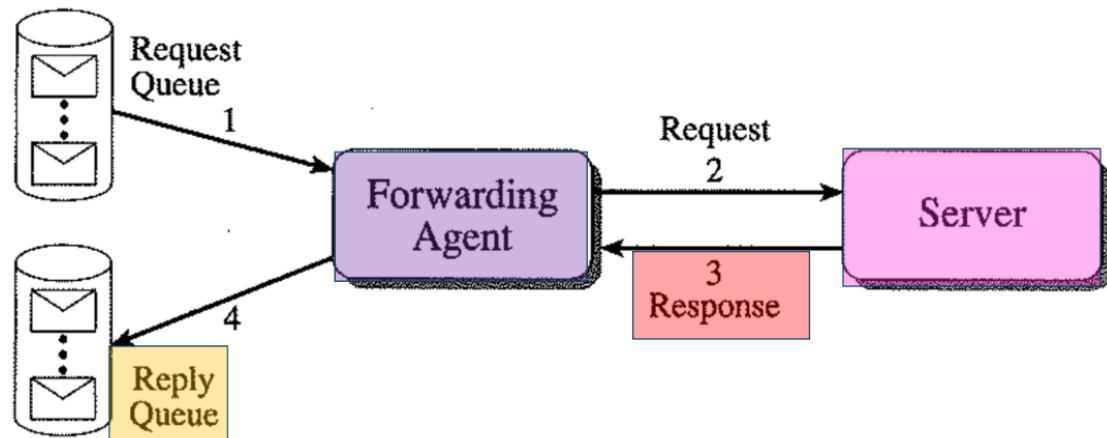


Figure 3.10 Forwarding Agent Converts Queue Operations to Request/Response

Ex-1103, ¶ 188; Ex-1107, 43. Tuxedo further discloses that “[t]he originator of the request, if it still exists, can access the response [*“retrieving the reply message”*] through this response (or reply) queue [*“from the reply queue”*].” Ex-1107, 42, 111m 425; Ex-1103, ¶ 190.

Bowman, Vahalia, and Tuxedo thus teach this limitation.

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

As demonstrated for limitation [2.9b] and [2.9d], Bowman, Vahalia, and Tuxedo disclose “*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 disk array) “*to the first node*” (to the original requestor of Tuxedo, e.g., the client from Bowman and/or Vahalia). Ex-1103, ¶ 192.

2. Claim 3

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

As demonstrated for claim [2.0], Bowman, Vahalia, and Tuxedo disclose “the messaging paradigm is a ... communication paradigm.”

Tuxedo further teaches “a conversational communication paradigm.”

Tuxedo’s Figure 3.6 shows a Conversational Programming Paradigm:

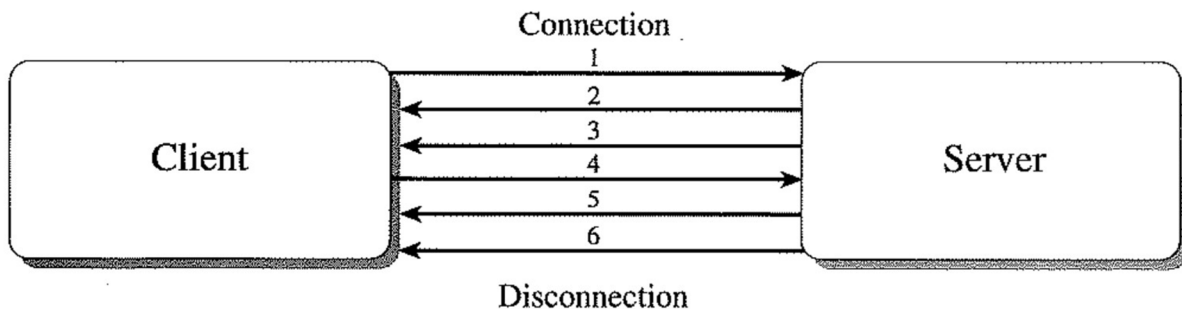


Figure 3.6 Conversational Programming Paradigm

Ex-1107, 35. Tuxedo explains:

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. 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. Ex-1103, ¶ 196.

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-1103, ¶ 197; Ex-1107, 27, 131, 217-18, xxxvi, xli, 131.

Bowman, Vahalia, and Tuxedo thus teach this limitation.

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

As demonstrated for limitation [1.1], Bowman and Vahalia teach this claim limitation. Ex-1103, ¶ 199.

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

As demonstrated for limitation [1.1], Bowman and Vahalia teach “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). Further, as demonstrated for limitation [3.0], the “communication agent in the FTSS” (e.g., Vahalia’s Data Mover) would use the “conversational communication paradigm” described in Tuxedo.

In addition, when Vahalia’s “communication agent in the FTSS” implements Tuxedo’s “conversational communication,” Dr. Almeroth explains that Vahalia’s

communication agent is “*a conversational communication agent.*” Ex-1103, ¶ 204; Ex-1107, 413 (defining “agent”).

In Figure 3.6, Tuxedo discloses that client (yellow) initiates communication with a Connection message [i.e., “*a connect message*”] in flow 1 (blue) sent to Server (pink):

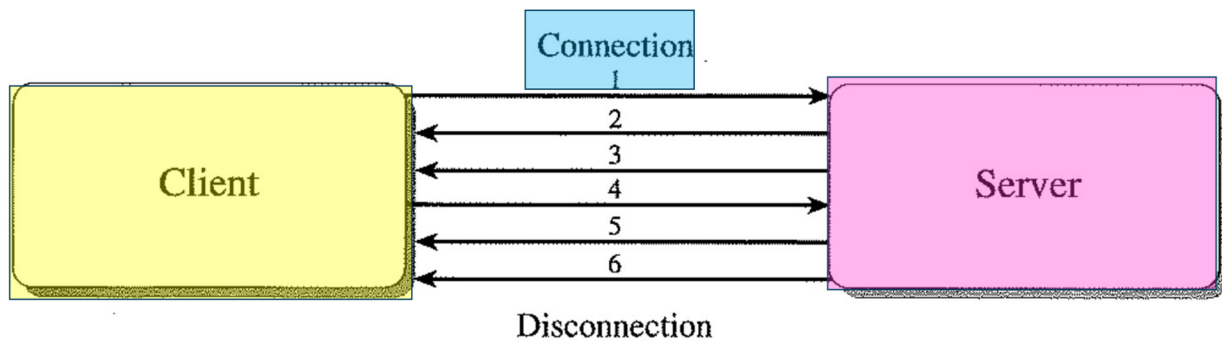


Figure 3.6 Conversational Programming Paradigm

Ex-1107, 35 (“The client initiates the communication with a connection message sent to the server (flow 1).”). Ex-1103, ¶¶ 202-03.

Bowman, Vahalia, and Tuxedo thus teach 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;***

As demonstrated in claim [2.4], Bowman, Vahalia, and Tuxedo teach “*a ... queue in highly reliable fault-tolerant storage media of the FTSS.*”

Tuxedo further teaches that “queues” are used with “conversational models,” (i.e., “*a conversation queue*”), Ex-1107, 227, and that these “conversational

models” are communication paradigms. Ex-1103, ¶¶ 208-09; Ex-1107, 27, 74 (describing “TUXEDO System’s request/response and conversational paradigms”).

Bowman, Vahalia, and Tuxedo thus teach this limitation.

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

As demonstrated for limitation [1.2], Bowman and Vahalia teach this claim limitation. Ex-1103, ¶ 214.

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

As demonstrated for limitation [2.4], Bowman, Vahalia, and Tuxedo teach “storing the ... message in the ... queue.”

As demonstrated for limitation [3.3a], Bowman, Vahalia, and Tuxedo teach that the queue is a “conversation queue.”

As demonstrated for limitation [3.2], Bowman, Vahalia, and Tuxedo teach that the message is a “connect message.”

Bowman, Vahalia, and Tuxedo thus teach this limitation.

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:

As demonstrated for limitation [1.4], Bowman and Vahalia teach “transmitting a message from the FTSS to the second node.”

As demonstrated for limitation [3.2], Bowman, Vahalia, and Tuxedo teach that the message is a “*connect message*.”

Bowman, Vahalia, and Tuxedo thus teach this limitation.

3. Claim 4

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

See claim 3.

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

As demonstrated for limitations [1.1] and [3.0], Bowman, Vahalia, and Tuxedo teach “*transmitting a ... message from the first node to the conversational communication agent in the FTSS*.”

Bowman further teaches “*transmitting a send message*.” For example, Bowman explains that a communication agent (MOM) “provides the application developer with a set of simple verbs (e.g., connect, send, receive, and disconnect) that are used to exchange information [“*transmitting a send message*”] with other distributed applications.” Ex-1105, 69:49-52. Ex-1103, ¶ 222.

Bowman, Vahalia, and Tuxedo thus teach this limitation.

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

As demonstrated for limitation [3.3c], Bowman, Vahalia, and Tuxedo teach “*storing the ... message in the conversation queue*.”

As demonstrated for limitation [4.1], Bowman, Vahalia, and Tuxedo disclose that the message is a “*send message*.”

Bowman, Vahalia, and Tuxedo thus disclose this limitation.

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

As demonstrated for limitation [2.8], Bowman and Vahalia disclose “*transmitting the ... message [from] the FTSS to a service on the second node.*”

As demonstrated for limitation [4.1], Bowman, Vahalia, and Tuxedo disclose that the message is a “*send message*.”

Bowman, Vahalia, and Tuxedo thus disclose 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;

As demonstrated for limitation [2.9a], Bowman, Vahalia, and Tuxedo disclose “*executing the service on the second node using the ... message as an input and providing a ... message as an output.*”

As demonstrated by limitation [2.9a], Bowman, Vahalia, and Tuxedo teach transmitting a “*send message*” to the service, and a POSITA would have understood the “*service in the second node*” to use the incoming send message as “*an input.*” Ex-1103, ¶ 226; Ex-1107, 67, 177.

Bowman teaches using “simple verbs” for messaging, including “receive” (i.e., a “*receive message*”). Ex-1103, ¶ 227; Ex-1105, 69:47-52.

Tuxedo teaches buffering communication for longer messages as part of implementing a conversational messaging paradigm. Ex-1107, 35 (“One common use for conversations is to buffer portions of a lengthy reply from a server to a client.”). In such situations, where Tuxedo’s server has only sent part of some data Tuxedo’s client has requested, client transmits a message to server requesting that server send information held in the buffer. Ex-1103, ¶ 228; Ex-1107, 35 (“In this case, the server might only send part of the data and indicate to the client that it has ‘more’ reply data. The client accepts the initial amount of reply data, processes it, and when it is ready, requests that the server send some ‘more.’”).

One way to communicate this request is by transmitting one of Bowman’s simple verbs (e.g., Bowman’s “send”) from client to server, such that server can determine from the “*send message*” that it should respond with more data. Ex-1103, ¶ 227; Ex-1107, 34-36. In this arrangement, “*the second node*” (server) would be “*using the send message*” (Bowman’s “send”) “*as an input*” (as an indication to server that it should respond with more data). Ex-1103, ¶ 229; Ex-1107, 34-36.

Tuxedo further teaches that in the buffered conversational communication described above it is necessary for the server and client to signal when each is ready to receive further information. Ex-1107, 35-36 (“In the buffering example cited previously, it serves no purpose for the server just to keep sending

information to the client. The whole purpose for the buffering is for the server to hold the information until the client is ready to accept it. ... For example, it is generally a protocol violation for both sides of a conversation to be awaiting a message from the other party.”). Dr. Almeroth explains that one way to implement such a signal would be using a “simple verb” from Bowman, such as “receive.” Ex-1103, ¶ 227 (citing Ex-1105, 69:47-52 (“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.”)). Moreover, in the buffered scenario described in Tuxedo, the client should be alerted that the server has finished transmitting information and is ready to receive further messages, as Dr. Almeroth demonstrates. Ex-1103, ¶ 228 (citing Ex-1107, 35-36).

Finally, the message of “receive” from server to client is “*a receive message*” and the server is “*providing the receive message as an output.*” Ex-1103, ¶ 229 (citing Ex-1107, 34-36).

Bowman, Vahalia, and Tuxedo thus disclose this limitation.

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

As demonstrated for limitation [1.1], Bowman and Vahalia disclose “transmitting the ... message from the second node to the ... communication agent in the FTSS.”

As demonstrated for limitation [3.2], Bowman, Vahalia, and Tuxedo disclose the “message” is a “receive message.”

As demonstrated for limitation [4.4], Bowman, Vahalia, and Tuxedo disclose the “communication agent” is a “conversational communication agent.”

Bowman, Vahalia, and Tuxedo thus teach this limitation.

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

As demonstrated for limitation [2.4], Bowman and Vahalia disclose “storing the ... message in the ... queue.”

As demonstrated for limitation [3.2], Bowman, Vahalia, and Tuxedo disclose the “message” is “a receive message.”

As demonstrated for limitation [4.4], Bowman, Vahalia, and Tuxedo disclose the “queue” being a “conversation queue.”

Bowman, Vahalia, and Tuxedo thus teach this limitation.

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

As demonstrated for limitation [2.9e], Bowman and Vahalia disclose
“*transmitting the ... message to the first node.*”

As demonstrated for limitation [3.2], Bowman, Vahalia, and Tuxedo disclose
“*message*” is a “*receive message.*”

Bowman, Vahalia, and Tuxedo thus teach this limitation.

4. Claim 5

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

See claim 3.

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

As demonstrated for limitation [1.1], Bowman and Vahalia disclose
“*transmitting a ... message from the first node to the ... communication agent in the FTSS.*”

As demonstrated for limitation [3.2], Bowman, Vahalia, and Tuxedo disclose
that the “*communication agent*” is a “*conversational communication element.*”

Tuxedo further teaches “*transmitting a disconnect message from the first node.*” For example, Tuxedo explains 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:

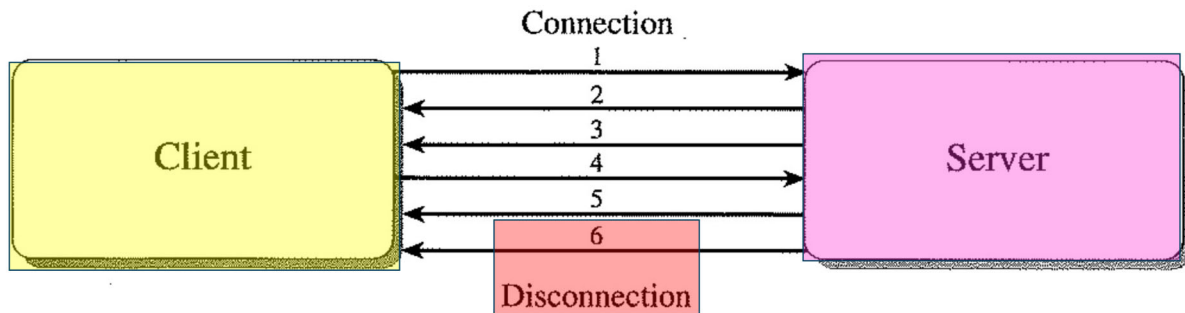


Figure 3.6 Conversational Programming Paradigm

Ex-1107, 35-36. 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).” [i.e., “transmitting a disconnect message”]). Ex-1103, ¶¶ 241-42. Ex-1107, 34.

Bowman, Vahalia, and Tuxedo thus disclose this limitation.

iii. [5.2] closing the conversation queue; and

Tuxedo explains that a conversation “is terminated by the server sending a disconnect message to the client,” Ex-1107, 34, which Dr. Almeroth explains would comprise “closing the conversation queue” because it is releasing access to the storage space on which the queue is stored. Ex-1103, ¶ 245 (citing Ex-1107, 227, 283, 175 (“conversational connections are closed”)); Ex-1114, 90 (defining “close”), 516 (defining “terminate”)).

Bowman, Vahalia, and Tuxedo thus teach this claim limitation.

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

As demonstrated for limitation [1.4], Bowman and Vahalia disclose “transmitting the ... message to the second node.”

As demonstrated for limitation [5.1], Bowan, Vahalia, and Tuxedo teach the message is a “disconnect message.”

Bowman, Vahalia, and Tuxedo thus teach this limitation.

5. Claim 6

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

Tuxedo teaches an “event-based communication paradigm.” For instance, Tuxedo discloses that “[t]he fourth major communication paradigm offered in ATMI is event-based communication [i.e., “the messaging paradigm is an event-based communication paradigm.”]” Ex-1103, ¶ 250 (citing Ex-1107, 241-242 (describing implementation)).

Bowman, Vahalia, and Tuxedo thus teach this 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;***

As demonstrated for limitation [1.1], Bowman and Vahalia discloses “transmitting a ... message from the first node to [a] ... communication agent in the FTSS.”

Initially, the '244 Patent describes the use of “*post message[s]*” in an event-based messaging paradigm as being a message which is sent after an event generator notifies the event broker or “posts” an event. Ex-1101, 2:50-60.

Tuxedo further teaches such “*a post message*.” For example, as part of its event-based communication paradigm, Tuxedo describes “[b]rokered events” that “allow programs to post events [i.e., “*a post message*”] without having to worry which other programs 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.*, and defines “event posting” as “sending an event to the EventBroker that will be forwarded to interested targets [i.e., “*a post message*”].” *Id.*, 418 (defining “EventBroker” as “module that forwards posted events to interested (registered) targets”). Ex-1103, ¶ 256 (citing Ex-1107, 421, 250, 418).

Moreover, because the “*communication agent*” is implementing an event-based communication paradigm, the communication agent is “*an event-based communication agent*.” Ex-1103, ¶ 257; Ex-1107, 413.

Bowman, Vahalia, and Tuxedo thus teach this 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;*

Tuxedo discloses using an event broker (red) “accessing a subscription manager” to maintain a subscription list (yellow):

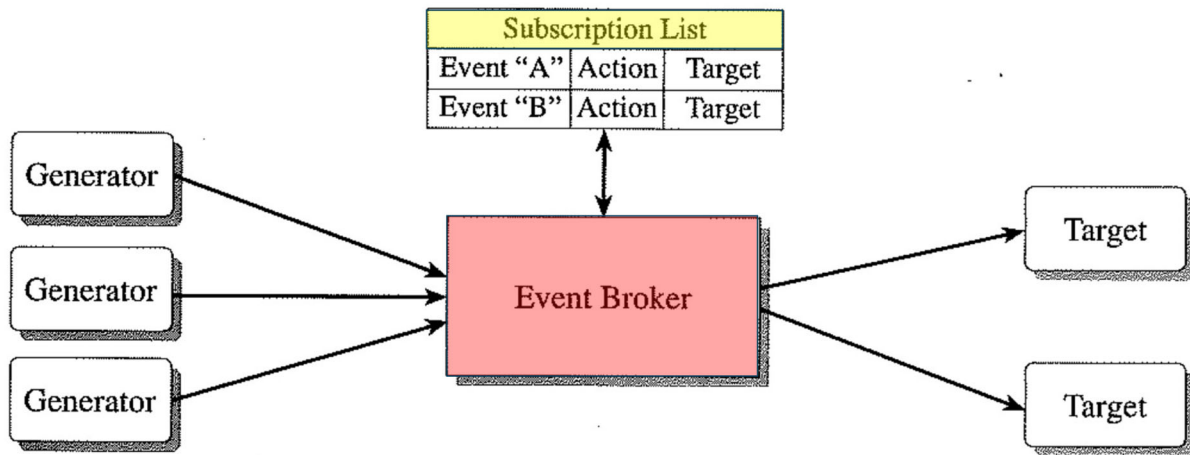


Figure 3.8 Delivery of Events via a Broker

Ex-1107, 39. 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.*; Ex-1103 ¶ 261.

Accordingly, Tuxedo’s “subscription manager” (event broker) is “access[ed] . . . to determine whether any nodes have subscribed to receive the event represented by the post message” (i.e., determine whether subscription list indicates any targets should be delivered an event based on the action, such as receiving the post message). Ex-1103 ¶ 262; Ex-1107, 39-40.

Bowman, Vahalia, and Tuxedo thus teach 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***

As demonstrated for limitation [2.4], Bowman, Vahalia, and Tuxedo disclose “storing the ... message in ... [a] queue in highly reliable fault-tolerant storage media of the FTSS.”

As demonstrated for limitation [6.2], Bowman, Vahalia, and Tuxedo disclose the message is “a post message.”

Bowman, Vahalia, and Tuxedo thus teach this limitation.

- 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.***

As demonstrated for limitation [1.4], Bowman and Vahalia teach “transmitting the ... message from the FTSS to the second.”

As demonstrated for limitation [6.2], Bowman, Vahalia, and Tuxedo discloses the message is “a post message.”

Tuxedo teaches transmitting the post message “if the second node has subscribed to receive the event represented by the post message.” As demonstrated for limitation [6.4], Tuxedo teaches using the 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. Dr. Almeroth explains that Tuxedo is disclosing 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-1103, ¶ 268; Ex-1107, 39-40.

Bowman, Vahalia, and Tuxedo thus disclose this limitation.

6. Claim 7

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

See 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*

As demonstrated for limitation [2.9b], Bowman, Vahalia, and Tuxedo disclose “*transmitting a ... message from the second node.*”

As demonstrated for limitation [6.2], Bowman, Vahalia, and Tuxedo “*discloses the event-based communication manager in the FTSS.*”

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. Dr. Almeroth explains that 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-1103, ¶ 272; Ex-1107, 39-40.

Moreover, Tuxedo explains that the targets are indicating to the event broker “their interest in events.” Ex-1107, 39. Tuxedo further teaches that “brokered event communication” includes “[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-1103, ¶ 273; Ex-1107, 39-40.

Bowman, Vahalia, and Tuxedo thus teach 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.***

As demonstrated for limitation [6.1], Bowman, Vahalia, and Tuxedo teach “*post messages.*”

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. Tuxedo thus teaches “*accessing the subscription manager*” (i.e., requesting that the 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-1103, ¶ 276; Ex-1107, 39-40.

Tuxedo further teaches that a target’s entry on the subscription list subscribes the target to receive notifications when the relevant event takes place i.e., “*receive post messages associated with the event.*” 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.”). Ex-1103, ¶ 277 (citing Ex-1107, 39-40).

Bowman, Vahalia, and Tuxedo thus disclose this limitation.

7. Claim 8

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

See 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***

As demonstrated for limitation [7.1], Bowman, Vahalia, and Tuxedo teaches “transmitting [a] ... message from the second node to the event-based communication manager in the FTSS, wherein the ... message includes an event.”

Tuxedo teaches that targets may unsubscribe from events (i.e., “an unsubscribe message”). 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. Given the disclosure of subscribe messages (as discussed for limitation [7.1]), Dr. Almeroth explains that these unsubscribe requests and related parameters are delivered to the event broker as a message—i.e., as “an unsubscribe message.” Ex-1103, ¶ 281; Ex-1107, 39-40, 255-56.

Bowman, Vahalia, and Tuxedo thus disclose 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.***

As demonstrated for limitation [6.4], Tuxedo teaches that the event broker maintains a subscription list indicating which targets have registered to receive post messages for which events.

As demonstrated for limitation [8.1], Tuxedo teaches that the event broker may receive unsubscribe messages to remove targets from the subscription list.

Tuxedo, therefore, discloses “*accessing the subscription manager*” (e.g., requesting that the 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-1103, ¶ 284; Ex-1107, 39-40, 255-56.

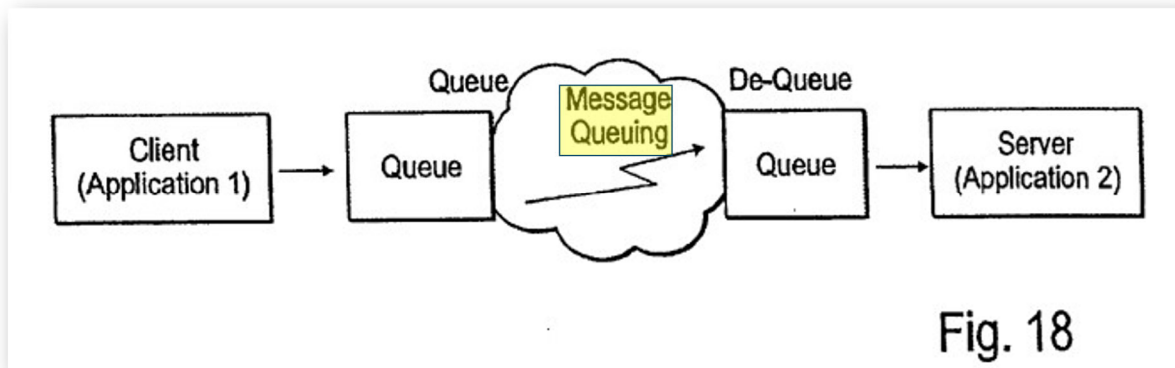
Tuxedo thus teaches this limitation.

8. Claim 9

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

As demonstrated for claim limitations [2.4], [2.6], [2.9c], [2.9d], [3.3c], [4.2], [4.6], [5.2], and [6.3], Bowman, Vahalia, and Tuxedo disclose using queues to implement messaging, including Tuxedo’s request queues, reply queues, and application queues.

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



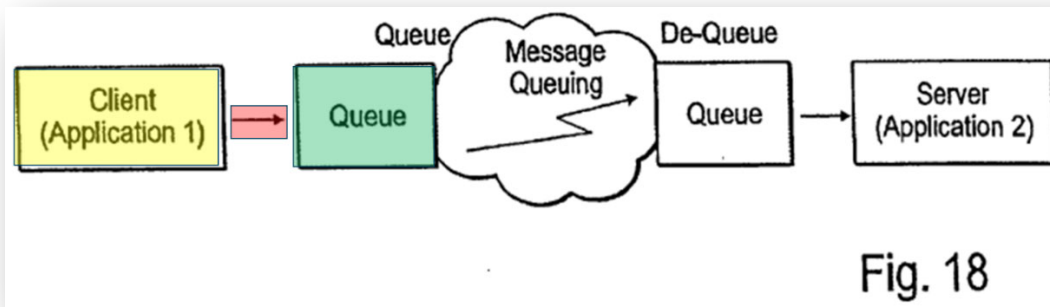
Ex-1105, Fig. 18. The use of queues to facilitate messaging (e.g., Tuxedo's application queues and Bowman's Message Queueing) thus discloses “a messaging paradigm” that “is a queue-based communication paradigm,” as Dr. Almeroth explains. Ex-1103, ¶ 289; Ex-1107, 27, 131.

Bowman, Vahalia, and Tuxedo thus teach 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;***

As demonstrated for limitation [1.1], Bowman and Vahalia teach “transmitting [a] ... message from the first node to a ... communication agent in the FTSS.”

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



Ex-1105, Fig. 18. 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.*, 70:22-26.

The '244 Patent describes an “*enqueue message*” as a message which has been pushed into a message queue. Ex-1101, 3:12-17, 9:45-53, 10:1-5. A POSITA would have understood the message in Figure 18 from the client to the queue to be “an *enqueue message*” as described in the '244 Patent. Ex-1103, ¶ 294; Ex-1107, 74 (“Both TUXEDO client and server programs can enqueue messages into or dequeue messages from application queues.”).

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.”

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-1103, ¶ 296; 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”).

Bowman and Tuxedo further teach 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. This queue manager “is responsible for enqueueing messages from message producers and providing them to message consumers [i.e., “*a queue-based communication agent*”] that request them via dequeue operations.” Ex-1107, 42; Ex-1103, ¶ 297 (citing Ex-1107, 413 (defining “agent”; Ex-1113, 500).

As demonstrated for limitations [1.1] and [1.2], this “*queue-based communication agent is in the FTSS.*”

Bowman, Vahalia, and Tuxedo thus disclose 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;***

As demonstrated for limitation [2.4], Bowman, Vahalia, and Tuxedo discloses “*storing the message in a queue ... in highly reliable fault tolerant storage media of the FTSS.*”

As demonstrated for limitation [9.2], Bowman, Vahalia, and Tuxedo disclose that “*the message*” is an “*enqueue message.*”

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, 228; Ex-1103, ¶ 301. 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.”).

Bowman, Vahalia, and Tuxedo thus teaches 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***

As demonstrated for limitation [1.3], Bowman and Vahalia teach “*processing the message at the FTSS in accordance with a messaging paradigm.*”

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. ... 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.”).

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-1103, ¶ 305; Ex-1107, 78, 240.

Bowman, Vahalia, and Tuxedo thus teach 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;***

As demonstrated for limitation [1.2], Bowman and Vahalia disclose “*transmitting the ... message from the FTSS to a service on the second node.*”

As demonstrated for limitation [9.4], Tuxedo teaches enqueueing messages when a service is unavailable for later processing (e.g., dequeuing) when the service is available.

Tuxedo further teaches that processing the message when the service becomes available also includes forwarding the message to the service (“*transmitting the enqueue message*”). 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-1103, ¶ 309; Ex-1107, 78, 240.

Bowman, Vahalia, and Tuxedo thus teaches 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;***

As demonstrated for limitation [2.1], 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 demonstrated for limitations [9.2] and [9.1], Bowman and Tuxedo teach enqueue messages and dequeue messages.

Bowman, Vahalia, and Tuxedo thus teach this limitation.

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

As demonstrated for limitation [2.9b], Bowman, Vahalia, and Tuxedo teach “transmitting the ... message from the second node to the ... communication agent in the FTSS.”

As demonstrated for limitations [9.2] and [9.1], Bowman, Vahalia, and Tuxedo teach “a queue-based communication agent” and “a deque message.”

Bowman, Vahalia, and Tuxedo thus teach this limitation.

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

As demonstrated for limitation [2.1], Bowman, Vahalia, and Tuxedo teach “storing the ... message in a data structure,” including a queue.

As demonstrated for limitations [9.4] and [9.1], Bowman, Vahalia, and Tuxedo discloses a “dequeue message” and teaches that the queue (and therefore the messages in the queue) is stored in a “queue space.”

Bowman, Vahalia, and Tuxedo thus teach this limitation.

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

As demonstrated for limitation [2.6], Bowman, Vahalia, and Tuxedo disclose “retrieving the ... message from the queue.” As demonstrated for limitation [9.4], Bowman, Vahalia, and Tuxedo teach that the queue is part of a “queue space,” such that the message is retrieved “from the queue space.”

As demonstrated for limitation [9.1], Bowman, Vahalia, and Tuxedo teach that “*the message*” is a “*dequeue message*.”

Bowman, Vahalia, and Tuxedo thus teaches this limitation.

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

As demonstrated for limitation [2.1], Bowman, Vahalia, and Tuxedo disclose “*transmitting the ... message from the FTSS to the first node*.”

As demonstrated for limitation [9.1], Bowman, Vahalia, and Tuxedo teach that the message is a “*dequeue message*.”

Bowman, Vahalia, and Tuxedo thus teach this limitation.

9. Claim 10

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

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. ... 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, 37-38, 79-80 (“These mechanisms allow your programs to communicate unsolicited notifications [*“an unsolicited communication paradigm”*] to client programs.”). Ex-1103, ¶ 318; Ex-1107, 27.

Bowman, Vahalia, and Tuxedo thus teach 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;***

As demonstrated for limitation [1.1], Bowman discloses “*transmitting a ... message from the first node to [a] ... communication agent in the FTSS.*”

Tuxedo discloses transmitting “*a broadcast message.*” Tuxedo teaches two mechanisms of event messaging: “single-program notifications and multiple-program broadcasting.” Ex-1107, 79. The multiple-program broadcasting mechanism [i.e., “*a broadcast message*”] “allows one or more clients to receive a single unsolicited notification.” Ex-1107, 80; Ex-1103, ¶ 321; 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”).

A POSITA would have understood that when the communication agent (e.g., Vahalia’s Data Mover) implements an unsolicited communication paradigm (e.g., Tuxedo’s multiple-program broadcasting), the communication agent is “*an unsolicited communication agent.*” Ex-1103, ¶ 322; Ex-1107, 413; Ex-1107, 171.

Bowman, Vahalia, and Tuxedo thus teach this limitation.

- 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;*²**

As demonstrated for limitations [1.3] and [6.4], 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. Dr. Almeroth explains that Tuxedo thus 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-1103, ¶ 324; Ex-1107, 39-40.

Bowman, Vahalia, and Tuxedo thus teach this limitation.

² Claim [10.2] includes what appears to be a typographical error (“accessing determining which nodes”). Petitioner has treated this term here to mean “determining which nodes.”

- 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***

As demonstrated for limitations [1.3] and [2.4], Bowman, Vahalia, and Tuxedo teach “*storing the ... message in an ... message queue in highly reliable fault-tolerant storage media of the FTSS.*” As demonstrated for limitation [10.0], Tuxedo further teaches an unsolicited communication paradigm based on broadcasting events.

As discussed for limitation [6.3], 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 the multi-program broadcasting described for limitation [10.2] to teach using the event queue as “*an unsolicited message queue.*” Ex-1103, ¶ 326.

Bowman, Vahalia, and Tuxedo thus teach 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.***

As demonstrated for limitations [1.4] and [10.4], 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”).

A POSITA would have understood these disclosures of notifying the multiple clients listed in subscription list 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-1103, ¶ 329; Ex-1107, 39-40, 79-80, 171-72, 241-42, 248-250, 421.

Bowman, Vahalia, and Tuxedo thus teach this limitation.

VIII. DISCRETIONARY DENIAL IS NOT APPROPRIATE

A. The Board Should Not Deny Institution Under 35 U.S.C. § 325(d)

Denial under § 325(d) is not warranted because the challenges presented in this Petition are neither cumulative nor redundant to the prosecution of the ’244 Patent (or related patents), as neither Bowman nor Vahalia were cited or considered by the Examiner during prosecution of the ’244 Patent. The references in this Petition were neither relied upon during prosecution nor are cumulative of art relied upon during prosecution. Further, the ’244 Patent has not been previously

challenged in an IPR. In addition, the Examiner also did not have the benefit of Kevin Almeroth, Ph.D.'s detailed expert testimony and the further evidence of record.

B. The Board Should Not Deny Institution Under 35 U.S.C. § 314(a)

The merits of Petitioner's arguments are compelling and the evidence in support is substantial. That "alone demonstrates that the PTAB should not discretionarily deny institution under *Fintiv*." Ex-1116, 4-5. But in any event, the six *Fintiv* factors do not justify denying institution.

Factor 1 (stay) is **neutral**. SAP has not yet moved to stay pending this IPR.

Factor 2 (proximity to trial) 2 is **neutral or weighs against denial**. In the originally filed litigation by Patent Owner Valtrus (*Valtrus I*), the trial date is not currently set (though jury selection is set for September 22, 2025. Ex-1117.

However, the Board should not rely on the current schedule in *Valtrus I* because Patent Owner Valtrus filed suit alone (without the other named Patent Owner, Key Patent Innovations) and thus did not have Article III Standing. Ex-1118; Ex-1119.

Petitioner filed a Motion to Dismiss for Lack of Standing in *Valtrus I*, reconsideration of which remains pending and casts serious doubt on whether *Valtrus I* will actually proceed along the current schedule. Indeed, in response to Petitioner's Motion to Dismiss for Lack of Standing, Patent Owners essentially concede that Valtrus alone lacked standing because Patent Owners together filed a

second, duplicative lawsuit (*Valtrus II*) in an effort to rectify the Article III standing deficiencies in *Valtrus I*. Ex-1120; Ex-1121. This conduct confirms that Patent Owners themselves doubt that the trial schedule in *Valtrus I* will hold, further supporting that the Board should not rely on that schedule at all. Additionally, the other pending litigations involving the challenged patent (*Valtrus II* and a declaratory judgment action filed by Petitioners) do not have any trial date currently set, meaning there is no reliable trial date for the Board to consider in relation to this Petition.

Even if the Board assumes the *Valtrus I* trial date applies, this factor is neutral because there will be other claims not litigated at trial that are covered by this Petition. Therefore, a final written decision will issue first at least on the claims not litigated at trial but covered by this Petition.

Factor 3 (investment in parallel proceedings) **weighs against denial**. The related litigations are in its early stages of discovery, motions to transfer and dismiss have been filed, and the investment by the parties has been relatively minimal. All litigations involving the challenged patent are at an early stage, with major deadlines several months away. Ex-1117 Patent Owners may argue that the parties have invested substantial resources on procedural motions, but the resources expended have been directed toward resolving procedural missteps perpetuated by Patent Owner Valtrus's choice to file alone (without Article III

standing) and Patent Owners’ later attempts to fix that with procedurally improper duplicative litigation. These procedural disputes (caused by Patent Owners’ actions) should not be held against Petitioner under this factor.

Factor 4 (overlap) **weighs against denial** because Petitioner hereby stipulates that, if the IPR is instituted, it will not pursue the IPR grounds in the district court. Indeed, Vahalia and Tuxedo are not at issue in the District Court Litigation. Ex-1121. Thus, “[i]nstituting trial here serves overall system efficiency and integrity goals by not duplicating efforts and by resolving materially different patentability issues.” *Apple, Inc. v. SEVEN Networks, LLC*, IPR2020-00156, Paper 10, 19 (June 15, 2020); *see also Sand Revolution II, LLC v. Continental Intermodal Group-Trucking LLC*, IPR2019-01393, Paper 24, 12 (June 16, 2020). In addition, Petitioner challenges more claims than Patent Owners will ultimately assert at trial. Patent Owner Valtrus agreed to narrow to 30 total asserted claims by April 30, 2025, and a maximum of six asserted claims per asserted patent. Ex-1117. Thus, Patent Owner will be limited to asserting up to six of the ten asserted claims of the ’244 Patent, merely a subset of the claims challenged in this Petition. *See, e.g., Apple*, No. IPR2020-00156, Paper 10,, 17, 21–22.

Factor 6 (other circumstances) **weigh against denial**. Here, the merits of the Petition are particularly compelling—for example, REFERENCE is a strikingly similar disclosure to that of the ’244 Patent, is directed at the same problem and

proposes the same solutions as the '244 Patent and discloses nearly identical architecture, including several nearly identical figures. The evidence presented, if unrebutted in trial, would plainly lead to a conclusion that one or more claims are unpatentable by a preponderance of the evidence.

While Factor 5 (parties) may weigh slightly in favor of denial because the parties are the same as in district court, under the “holistic view” of the remaining factors and whether integrity of the system and efficiency is best served, institution is favored. *Samsung Elecs. Co. Ltd. v. Dynamics Inc.*, IPR2020-00505, Paper 11, 15 (Aug. 12, 2020).

IX. CONCLUSION

Claims 1–10 of the '244 Patent are unpatentable for the reasons demonstrated above.

Dated: January 15, 2025

Respectfully Submitted,

/s/ Louis L. Campbell

Louis L. Campbell (Lead Counsel)

llcampbell@winston.com

USPTO Reg. No. 59,963

Winston & Strawn LLP

255 Shoreline Dr., Ste. 520

Redwood City, CA 94065

T: (650) 858-6500

Lead Counsel for Petitioner

CERTIFICATE OF COMPLIANCE

This Petition complies with the word count limits set forth in 37 C.F.R.

§ 42.24(a)(1)(i), because this Petition contains 13,496 words, excluding the parts of the Petition exempted by 37 C.F.R. § 42.24(a)(1), determined using the word count provided by Microsoft Word, which was used to prepare this Petition.

Dated: January 15, 2025

Respectfully submitted,

/s/ Louis L. Campbell

Louis L. Campbell (Lead Counsel)

llcampbell@winston.com

USPTO Reg. No. 59,963

Winston & Strawn LLP

255 Shoreline Dr., Ste. 520

Redwood City, CA 94065

T: (650) 858-6500

Lead Counsel for Petitioner

CERTIFICATE OF SERVICE

Under 37 C.F.R. §§ 42.6(e) and 42.105(a), this is to certify that on January 15, 2025, I caused to be served a true and correct copy of the foregoing “**PETITION *INTER PARTES* REVIEW OF U.S. PATENT NO. 6,889,244**” and accompanying exhibits by Federal Express priority mail on the following:

Botos Churchill IP Law LLP
430 Mountain Avenue
Suite 401
New Providence, NJ 07974

With a courtesy copy of this Petition and supporting material was also served on litigation counsel for Patent Owners via email:

Matt Berkowitz
mberkowitz@reichmanjorgensen.com

Patrick R. Colsher
pcolsher@reichmanjorgensen.com

Aaron Morris
amorris@reichmanjorgensen.com

Navid Cyrus Bayar
nbayar@reichmanjorgensen.com

Reichman Jorgensen Lehman & Feldberg LLP
100 Marine Parkway
Suite 300
Redwood Shores, CA 94065

Khue Van Hoang
khoang@reichmanjorgensen.com
Reichman Jorgensen Lehman & Feldberg LLP
400 Madison Avenue
Suite 14d
New York, NY 1001

Philip Ekle
peklem@reichmanjorgensen.com
Savannah Harman Carnes

scarnes@reichmanjorgensen.com

Reichman Jorgensen Lehman & Feldberg LLP
1909 K Street, NW
Suite 800
Washington, DC 20006

/s/ Louis L. Campbell

Louis L. Campbell (Lead Counsel)

llcampbell@winston.com

USPTO Reg. No. 59,963

Winston & Strawn LLP

255 Shoreline Dr., Ste. 520

Redwood City, CA 94065

T: (650) 858-6500

Lead Counsel for Petitioner