

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

JUNIPER NETWORKS, INC.
Petitioner,

v.

PORTSMOUTH NETWORK CORPORATION
Patent Owner

Case No. IPR2024-00893
Patent No. 6,917,986

**PETITION FOR *INTER PARTES* REVIEW
UNDER 35 U.S.C. §§ 311-319 AND 37 C.F.R. § 42.1 *et seq.***

TABLE OF CONTENTS

I.	MANDATORY NOTICES	1
A.	Real Party-In-Interest Under 37 C.F.R. §42.8(b)(1)	1
B.	Related Matters Under 37 C.F.R. §42.8(b)(2)	1
1.	Patent Trial and Appeal Board.....	1
2.	Federal District Courts.....	1
C.	Counsel & Service Information Under 37 C.F.R. §§42.8(b)(3)-(4)	1
II.	INTRODUCTION	3
III.	CERTIFICATION OF STANDING	7
IV.	BACKGROUND	7
A.	Setting Up a Spanning Tree Network	9
1.	STP Bridge Roles.....	10
2.	STP Bridge Port Roles	11
3.	STP Bridge Port States.....	12
4.	Maintaining STP Using Bridge Databases	13
B.	STP Self-Healing Process	14
1.	Link Failure Detection	14
2.	Link Failure Recovery	15
V.	THE '986 PATENT.....	16
A.	The Challenged Claims	19
B.	Prosecution History	20
C.	Claim Interpretation	20
D.	Level of Ordinary Skill in the Art	20
VI.	TABLE OF GROUNDS	20

VII. UNPATENTABILITY GROUNDS.....	21
A. Ground 1: Claims 1-3, 6-15, and 18-24: Obvious Over Gai in View of IEEE 1999.....	21
1. U.S. Patent No. 6,032,194 (“Gai,” EX1005)	21
a. Port Reconfiguration After Link Failure	22
b. Dummy Traffic	24
c. Network Initialization Commands	24
2. IEEE 1999 (EX1009).....	25
3. Gai in View of IEEE 1999	26
4. Claim 1	30
a. [1PRE] A method for network communication, comprising:	30
b. [1A] coupling an edge switch in a network to one or more access switches via at least first and second redundant links, respectively connected to first and second ports of the edge switch;.....	31
c. [1B] placing the first port in an active state, while placing the second port in a blocking state;.....	34
d. [1C] conveying communication traffic over the first link while the first port is in the active state, thereby causing each of the switches to build a respective database for use in forwarding the traffic;.....	35
e. [1D] responsive to a failure associated with the first link, placing the second port in the active state and the first port in the blocking state; and	37
f. [1E] sending dummy traffic from the edge switch over the second link to the one or more access switches, so as to cause each of the one or more access switches to modify its respective database responsive to the second port being in the active state.....	40

5.	Claim 2: A method according to claim 1, wherein placing the second port in the blocking state comprises configuring the first and second ports so that no loop is created in the network between the edge switch and the one or more access switches.....	46
6.	Claim 3: A method according to claim 2, wherein configuring the first and second ports comprises configuring the edge switch so as to decrease a radius of a spanning tree established in the network.	47
7.	Claim 6: A method according to claim 1, wherein sending the dummy traffic comprises sending data frames originating at the edge switch while having source addresses corresponding to network addresses of other entities in the network that are accessible to the access switches via the edge switch.....	48
8.	Claim 7: A method according to claim 6, wherein sending the data frames comprises copying the source addresses for the data frames from the database of the edge switch.	49
9.	Claim 8: A method according to claim 7, wherein.....	49
a.	[8A] the edge switch has three or more ports, including the first and second ports, and.....	49
b.	[8B] wherein prior to the failure, the database comprises entries associated with a plurality of the ports, and.....	50
c.	[8C] wherein copying the source addresses comprises using the network addresses in all the entries that are not associated with the first port as the source addresses of the data frames.	52
10.	Claim 9: A method according to claim 6, wherein sending the data frames comprises incorporating a Virtual Local Area Network (VLAN) tag in a header of the data frames.....	52
11.	Claim 10: A method according to claim 6, wherein sending the data frames comprises multicasting the data frames to all of the one or more access switches.....	53

12.	Claim 11: A method according to claim 1, wherein sending the dummy traffic comprises sending data frames downstream from the edge switch toward the access switches.....	53
13.	Claim 12.....	54
a.	[12A] A method for network communication, comprising: coupling a first bridge in a network to a second bridge located downstream of the first bridge, by connecting at least first and second redundant links to respective first and second ports of the first bridge so as to communicate with the second bridge;....	54
b.	[12B] placing the first port in an active state, while placing the second port in a blocking state;.....	54
c.	[12C] conveying communication traffic over the first link while the first port is in the active state, thereby causing the bridges to build respective databases for use in forwarding the traffic;	54
d.	[12D] responsive to a failure associated with the first link, placing the second port in the active state and the first port in the blocking state; and	54
e.	[12E] sending dummy traffic from the first bridge over the second link downstream to the second bridge, so as to cause the second bridge to modify its database responsive to the second port being in the active state...	55
14.	Claim 13.....	55
a.	[13A] A network switching device, comprising: a network port, adapted to be coupled over a network trunk link to a core switch in a network so as to exchange communication traffic with the core switch;..	55
b.	[13B] first and second access ports, adapted to be coupled via first and second redundant network access links to one or more access switches serving network users; and	56

- c. [13C] a protocol processor, adapted to place the first access port in an active state, while placing the second access port in a blocking state, so that the communication traffic is conveyed over the first link while the first port is in the active state, thereby causing each of the access switches to build a respective access switch database for use in forwarding the traffic, the protocol processor being further adapted, responsive to a failure associated with the first link, to place the second access port in the active state and the first port in the blocking state and to send dummy traffic through the second access port over the second link to the one or more access switches, so as to cause each of the one or more access switches to modify its respective access switch database responsive to the second access port being in the active state.....56
- 15. Claim 14: A device according to claim 13, wherein the protocol processor is adapted to configure the first and second access ports so that no loop is created in the network between the edge switch and the one or more access switches. 57
- 16. Claim 15: A device according to claim 14, wherein the protocol processor is further adapted to configure the first and second access ports so as to decrease a radius of a spanning tree established in the network. 58
- 17. Claim 18: A device according to claim 13, wherein the dummy traffic comprises data frames originating at the edge switch while having source addresses corresponding to network addresses of other entities in the network that are accessible to the access switches via the edge switch. 58
- 18. Claim 19: A device according to claim 18, and.....58
 - a. [19A] comprising a memory,.....58

b.	[19B] wherein the protocol processor is adapted to build an edge switch database in the memory responsive to the communication traffic prior to the failure associated with the first link, and to copy the source addresses for the data frames from the edge switch database.	59
19.	Claim 20: A device according to claim 19, wherein prior to the failure, the edge switch database comprises entries associated with a plurality of the ports, and wherein the protocol processor is adapted to use the network addresses in all the entries that are not associated with the first access port as the source addresses of the data frames.	59
20.	Claim 21: A device according to claim 18, wherein the protocol processor is adapted to incorporate a Virtual Local Area Network (VLAN) tag in a header of the data frames.....	59
21.	Claim 22: A device according to claim 18, wherein the protocol processor is adapted to multicast the data frames to all of the one or more access switches.....	60
22.	Claim 23: A device according to claim 13, wherein the dummy traffic comprises data frames sent downstream from the edge switch toward the access switches.....	60
23.	Claim 24	60
a.	[24A] A bridge device for use in a communication network having a spanning tree root, the device comprising: a designated port, adapted to be coupled over an upstream link in the network so as to communicate with the root;	60
b.	[24B] first and second downstream ports, adapted to be coupled via first and second redundant links to a further bridge device located downstream in the network relative to the spanning tree root; and	61

c.	[24C] a protocol processor, which is adapted to place the first downstream port in an active state, while placing the second downstream port in a blocking state, so that communication traffic is conveyed over the first downstream link while the first port is in the active state, thereby causing the bridge devices to build respective databases for use in forwarding the traffic, the protocol processor being further adapted, responsive to a failure associated with the first link, to place the second downstream port in the active state and to send dummy traffic over the second link downstream to the further bridge device, so as to cause the further bridge device to modify its database responsive to the second downstream port being in the active state.....	61
B.	Ground 2: Claims 4, 5, 16, and 17: Obvious over Gai in View of IEEE 1999, Further in View of Jain	62
1.	U.S. Patent No. 6,330,229 (“Jain,” EX1006)	62
2.	Gai in View of IEEE 1999, Further in View of Jain	62
3.	Claim 4: A method according to claim 1, and comprising, responsive to placing the second port in the active state, updating the database of the edge switch so that entries in the database point to the second port instead of the first port. .	64
4.	Claim 5: A method according to claim 4, wherein updating the database comprises changing the entries in the database to point to the second port before receiving the communication traffic from the one or more access switches on the second port.	64
5.	Claim 16: A device according to claim 13, and.....	65
a.	[16A] comprising a memory,.....	65
b.	[16B] wherein the protocol processor is adapted to build an edge switch database in the memory responsive to the communication traffic prior to the failure associated with the first link.....	65

c.	[16C] and, responsive to placing the second access port in the active state, to update the edge switch database so that entries in the edge switch database point to the second access port instead of the first access port.....	66
6.	Claim 17: A device according to claim 16, wherein the protocol processor is adapted to update the entries in the database to point to the second access port before receiving the communication traffic from the one or more access switches on the second por.	66
VIII.	DISCRETIONARY DENIAL IS NOT APPROPRIATE	66
A.	Fintiv.....	66
1.	Potential Stay	67
2.	Timing of Trial.....	67
3.	Litigation Investment	68
4.	Overlap of Issues.....	68
5.	Litigation Defendants.....	68
B.	Section 325(d) and <i>Advanced Bionics</i>	69
C.	Section 314(a) and <i>General Plastic</i>	71
IX.	CONCLUSION.....	73
X.	CLAIM LISTING	76

TABLE OF AUTHORITIES

Cases

<i>Advanced Bionics, LLC v. MED-EL Elektromedizinische Geräte GmbH,</i> IPR2019-01469, Paper 6 (Feb. 13, 2020)	69, 70
<i>Align Technology, Inc. v. Dental Monitoring SAS,</i> IPR2024-00110, Paper 7 (Apr. 19, 2024)	69
<i>AliveCor, Inc. v. Apple Inc.,</i> IPR2023-00950, Paper 12 (Jan. 9, 2024)	69, 70, 72
<i>Apple Inc. v. Fintiv, Inc.,</i> IPR2020-00019, Paper 11 (Mar. 20, 2020)	66, 67, 68
<i>Apple v. Seven Networks,</i> IPR2020-00235, Paper 10 (July 28, 2020)	68
<i>Arista Networks, Inc. v. Corrigent Corp.,</i> IPR2023-00839, Paper 9 (Dec. 7, 2023)	72
<i>Bowtech, Inc. v. MCP IP, LLC,</i> IPR2019-00382, Paper 12 (Aug. 6, 2019)	71
<i>Code 2000 v. Bright Data Ltd.,</i> IPR2022-00353, Paper 8 (July 1, 2022)	70
<i>Dolby Labs. v. Intertrust Techs.,</i> IPR2020-00665, Paper 11 (Feb. 16, 2021)	67
<i>General Plastic Indus. Co. Ltd. v. Canon Kabushiki Kaisha,</i> IPR2016-01357, Paper 19 (Sept. 6, 2017)	71
<i>Google LLC v. Geoscope Techs. Pte. Ltd.,</i> IPR2023-01212, Paper 14 (Feb. 20, 2024)	68
<i>Group III Int’l, Inc. v. Targus Group Int’l,</i> IPR2021-00371, Paper 21 (July 9, 2021)	69
<i>Mercedes-Benz USA, LLC v. Spero,</i> IPR2023-01040, Paper 10 (Dec. 12, 2023)	72

<i>Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co.</i> , 868 F.3d 1013 (Fed. Cir. 2017)	20
<i>Nokia of Am. Corp. v. TQ Delta, LLC</i> , IPR2022-00471, Paper 11 (Aug. 18, 2022)	66
<i>Pfizer Inc. v. ModernaTX, Inc.</i> , IPR2023-01358, Paper 19 (Mar. 19, 2024)	71
<i>Samsung Elec. Co., Ltd. v. Netlist, Inc.</i> , IPR2022-00063, Paper 13 (May 5, 2022)	68
<i>Sand Revolution II, LLC v. Cont’l Intermodal Grp.–Trucking LLC</i> , IPR2019-01393, Paper 24 (June 16, 2020)	67, 68
<i>Videndum Prod. Sol’ns, Inc. v. Rotolight Ltd.</i> , IPR2023-01218, Paper 12 (Apr. 19, 2024)	71
Regulations	
37 C.F.R. §42.100(b)	20
37 C.F.R. §42.104(a)	7
Statutes	
35 U.S.C. § 102(b)	21, 25
35 U.S.C. § 103	20
35 U.S.C. § 314(a)	71
35 U.S.C. § 315(b)	7, 68
35 U.S.C. § 325(d)	69
Other Authorities	
Director’s Interim Procedure for Discretionary Denials, https://tinyurl.com/2s4rmz9x	66

EXHIBIT LIST

Exhibit	Description
1001	U.S. Patent No. 6,917,986 (“the ’986 patent”)
1002	Prosecution History of U.S. Patent No. 6,917,986
1003	Declaration of Dr. Bruce McNair (“McNair”)
1004	CV of Dr. Bruce McNair
1005	U.S. Patent No. 6,032,194 (“Gai”)
1006	U.S. Patent No. 6,330,229 (“Jain”)
1007	“IEEE Standard for Local Area Network MAC (Media Access Control) Bridges,” in ANSI/IEEE Std 802.1D, 1998 Edition, pp.1-373, 11 Dec. 1998, doi: 10.1109/IEEESTD.1998.95619.
1008	Declaration of David Ringle Relating to Exhibit 1007
1009	S. Varadarajan and T. Chiueh, “Automatic Fault Detection and Recovery in Real Time Switched <i>Ethernet</i> Networks,” IEEE INFOCOM ‘99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320), New York, NY, USA, 1999, pp. 161-169 vol.1, doi: 10.1109/INFCOM.1999.749264.
1010	Declaration of Rachel J. Watters Relating to Exhibit 1009
1011	Declaration of Gordon MacPherson Relating to Exhibit 1009
1012	Federal Judicial Caseload Statistics (2024) – District of Massachusetts
1013	<i>Portsmouth Network Corp. v. Juniper Networks, Inc.</i> , 1:23-cv-12201 (D. Mass.), Dkt. No. 14 (Proof of Service) (Sept. 28, 2023)
1014	<i>Portsmouth Network Corp. v. Juniper Networks, Inc.</i> , 1:23-cv-12201 (D. Mass.), Dkt. No. 61 (Amended Complaint) (Dec. 22, 2023)
1015	<i>Portsmouth Network Corp. v. Cisco Systems, Inc.</i> , 2:23-cv-00441 (E.D. Tex.), Dkt. No. 57 (Amended Complaint) (Mar. 11, 2024)

I. MANDATORY NOTICES

A. Real Party-In-Interest Under 37 C.F.R. §42.8(b)(1)

Petitioner Juniper Networks, Inc. is the real party-in-interest.

B. Related Matters Under 37 C.F.R. §42.8(b)(2)

1. Patent Trial and Appeal Board

There is one other IPR proceeding involving U.S. Patent No. 6,917,986:

Cisco Systems, Inc. v. Portsmouth Network Corp., IPR2024-00505 (“Cisco IPR”).

2. Federal District Courts

The ’986 patent is involved in the following pending cases:

- *Portsmouth Network Corp. v. Juniper Networks, Inc.*, 1:23-cv-12201 (D. Mass.) (“Juniper Litigation”); and
- *Portsmouth Network Corp. v. Cisco Systems, Inc.*, 2:23-cv-00441 (E.D. Tex.) (“Cisco Litigation”).

C. Counsel & Service Information Under 37 C.F.R. §§42.8(b)(3)-(4)

Lead Counsel	Michael N. Rader (Reg. No. 52,146)
Backup Counsel	Scott A. McKeown (Reg. No. 42,866)
Service Information	<u>E-mail</u>: MRader-PTAB@wolfgreenfield.com SMcKeown-PTAB@wolfgreenfield.com <u>Post and Hand Delivery</u>: Wolf, Greenfield & Sacks, P.C. 600 Atlantic Ave. Boston, MA 02210-2206 <u>Phone</u>: (617) 646-8000 <u>Facsimile</u>: (617) 646-8646

A power of attorney is submitted with this petition. Counsel for Petitioner consents to electronic service of all documents via electronic mail at the email addresses provided above.

II. INTRODUCTION

Petitioner requests *inter partes* review of claims 1-24 (“challenged claims”) of U.S. Patent No. 6,917,986 (“the ’986 patent,” EX1001).

The ’986 patent describes network switches (*e.g.*, IEEE MAC bridges) that employ redundant communication links to maintain connectivity when an active link fails. 5:31-49, 7:8-12.¹ Typically, such switches are employed in the Spanning Tree arrangement of IEEE 802.1D (EX1007) in which, upon failure of the active link, the switch’s backup port in a blocked state transitions through multiple intermediate states to the active state to become the new active port. 3:1-4. Switches also transmit configuration messages to communicate this new Spanning Tree information to the network. EX1007, 62. However, this process can be time-consuming, taking up to approximately one minute overall for network service to be restored over the newly-activated link. 2:65-3:14.

The ’986 patent purports to provide faster adaptation to Spanning Tree link failures compared to the typical STP self-healing process. Rather than employing the typical STP sequence for switching between blocked and active ports and communicating new Spanning Tree information, the ’986 patent describes, upon detection of a failure, immediately switching a back-up port from the blocked state to the active state and blocking the failed active port. 4:13-17.

¹ Citations in Section II are to the ’986 patent unless otherwise noted.

The '986 patent provides an immediate switching mechanism, and avoids standard STP configuration messaging, by using “dummy frames.” By immediately transmitting dummy frames over the new active link via the new active port, downstream switches learn of topology changes independently of STP configuration messages. 4:13-31. This is because the dummy frames have source MAC addresses corresponding to upstream entities with which the downstream switches previously communicated. Receiving the dummy frames via the new active link/port causes the downstream switches to update their filtering databases so that their future upstream traffic will be directed to the new active link. 4:20-29.

The '986 patent admits that using dummy frames to communicate topology changes immediately in a Spanning Tree network was already known from U.S. Patent No. 6,032,194 (“Gai,” EX1005), which it incorporates by reference for this very disclosure. 3:15-29. The '986 patent argues that its scheme differs from Gai's by avoiding “special commands, not a part of the 802.1D standard, [that] are used to modify the STP parameters of both the core switches and the access switches in the network” of Gai when configured by a network administrator. 3:35-39, 4:38-49; EX1005, 8:52-56. However, the challenged claims do not preclude such commands. In any event, the special commands are not necessary to the dummy message self-healing function; Gai explains that they only provide an optimized setup limiting the link choices for self-healing. EX1005, 15:26-51.

The '986 patent also purports to differentiate itself from Gai because its dummy frames are used to manage “redundant downstream ports” while Gai’s dummy messages are instead sent upstream. 4:36-43, 3:15-41, 9:42-44; 9:62-65. However, it would have been obvious to POSAs to use Gai’s dummy message technique when a switch detects a link failure downstream of itself, in which case the dummy messages would be sent downstream. Commenting on the same 802.1D standard discussed in the '986 patent, a 1999 IEEE paper (“IEEE 1999,” EX1009) teaches sending network self-healing messages both upstream and downstream after detecting a downstream link failure to reduce the total link failure recovery time by having upstream and downstream switches address the failure in parallel. *Infra* §VII.A.2. In view of that teaching, it would have been obvious to send Gai’s dummy packets both upstream and downstream after detecting a downstream link failure. *Infra* §VII.A.3.

Further, Gai discloses that the same mechanism that its access switches use to send dummy packets upstream (*i.e.*, rapid reconfiguration entity 234), is implemented by upstream switches as well as access switches, indicating that such upstream switches can also send dummy messages. Thus, in view of IEEE 1999, the technique claimed in the '986 patent would have been obvious to use with any switch in the network, including an “edge switch” as recited in claim 1 of the '986 patent. *Infra* §VII.A.3.

Ground 1 demonstrates that claims 1-3, 6-15, and 18-24 are unpatentable as obvious over Gai in view of IEEE 1999.

Ground 2 demonstrates that claims 4-5, and 16-17 are unpatentable as obvious over Gai in view of IEEE 1999, further in view of U.S. Patent No. 6,330,229 (“Jain,” EX1006).

Accordingly, Petitioner respectfully requests that the challenged claims be canceled.

III. CERTIFICATION OF STANDING

The '986 patent is available for IPR and Petitioner is not barred or estopped from requesting IPR of the challenged claims. 37 C.F.R. §42.104(a). This Petition is being filed within the one-year period of 35 U.S.C. §315(b).

IV. BACKGROUND

The IEEE 802.1D standard defines a self-healing protocol for computer networks called Spanning Tree Protocol (STP). STP creates, communicates, and maintains information about a “spanning tree,” *i.e.*, a series of interconnected nodes (*e.g.*, bridges or local area networks (LANs)), that spans all the nodes by creating a tree structure. “Spanning” refers to the STP algorithm processing each node in the network to ensure that all the nodes are interconnected so that none are left out of the tree structure and therefore unable to communicate with the other nodes. Once the tree structure is created, the STP algorithm manipulates port states of its nodes to eliminate any communication loops. McNair, ¶¶46-65.

A computer network prior to executing the STP algorithm is shown in Figure 1 below. The network is formed by connecting bridge A to bridges B and C via their respective ports. Bridge A is connected to bridges B and C via redundant links, which creates a loop. The loop’s existence is not in accordance with STP, which removes such loops from a network. McNair, ¶¶66.

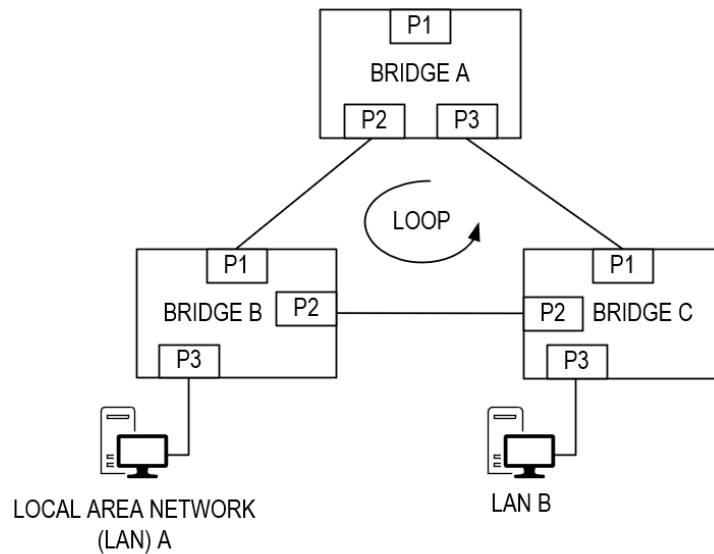


Figure 1: Computer network with loop; not in accordance with STP

Loops are to be avoided, particularly when broadcast or multicast traffic will be employed. If a loop exists, broadcast messages might circulate around the loop, creating unnecessary network congestion. A tree structure ensures there are no loops in the network. A fully connected tree structure, *i.e.*, one that spans all the nodes, ensures that all nodes are linked without loops and allows complete interconnection of the nodes with the minimum number of links. McNair, ¶¶28-48.

While a tree structure is advantageous for the reasons just described, it is inherently unreliable because failure of any single link in the tree severs portions of the network that are connected through this link into two distinct, separate networks with no ability to communicate. STP includes a self-healing function to solve this problem. McNair, ¶49.

A. Setting Up a Spanning Tree Network

Applying the STP algorithm to the computer network of Figure 1 above yields the result shown in Figure 2 below, with the interconnected nodes in a Spanning Tree arrangement. The STP algorithm forms the tree structure shown in Figure 2 below by first “spanning” the nodes to ensure they are all interconnected. For example, each node (*i.e.*, bridge or LAN) is connected to at least one other node in the network. Second, the STP algorithm creates the tree structure by identifying a **root bridge** at the “top” of the tree and “branches” and “leaves” downstream of the **root bridge**. In Figure 2 below, the STP algorithm identifies **designated bridge A** and **designated bridge B** as the “branches” of the tree and **local area network (LAN) A** and **LAN B** as the “leaves” of the tree. A bridge (*e.g.*, root bridge or designated bridge) is a switch that connects two LANs. For example, **designated bridge A** is a bridge because it connects LAN A and LAN B (via **designated bridge B**). McNair, ¶67.

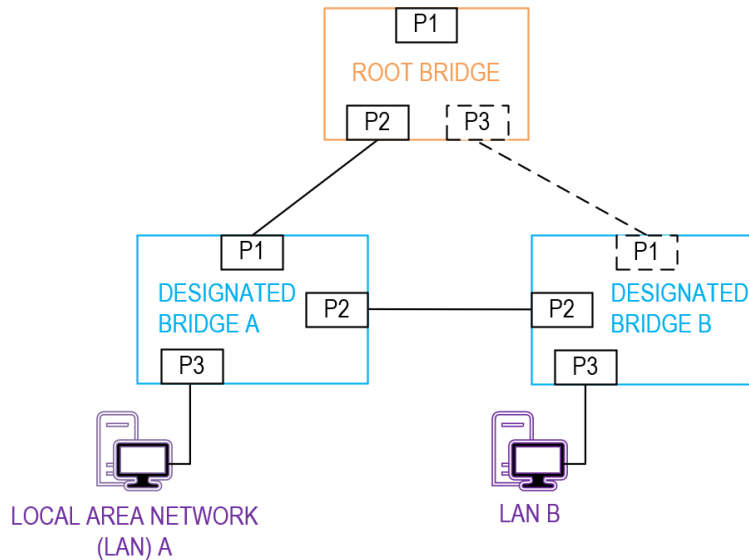


Figure 2: Computer network in accordance with STP

Once the tree structure is created, the STP algorithm manipulates the states of nodes' ports to eliminate loops. As represented by the dashed line in Figure 2 above, the STP algorithm in this example blocks port P3 of the **root bridge** and port P1 of **designated bridge B** to eliminate the connection between them, thus eliminating the loop that previously existed (shown in Figure 1 above). However, other implementations also comply with the IEEE 802.1D standard; port P3 of the **root bridge** can be left as a designated port in the forwarding state. McNair, ¶68.

1. STP Bridge Roles

The STP algorithm operates by assigning bridges a role: **root** or **designated**. The STP algorithm identifies one and only one bridge in the network as the **root bridge** and identifies the remaining bridges as **designated bridges**. A **root bridge** is a reference point for all other bridges in a network with a spanning tree topology.

A **designated bridge** routes its traffic upstream toward the **root bridge**. A **LAN** connects to a **designated bridge**. In Figure 2 above, the STP algorithm identifies bridge A of Figure 1 as the **root bridge** and bridges B and C of Figure 1 as **designated bridges A and B**, respectively. **LANs A and B** are connected to **designated bridges A and B**, respectively. McNair, ¶69.

2. STP Bridge Port Roles

The STP algorithm also assigns to each port of a bridge a role: root, designated, or alternate/backup (referred to herein as “blocked”). The result of assigning roles to the ports of the bridges in Figure 2 above is shown in Figure 3 below. A bridge’s root port is used to send data messages to the **root bridge** along the lowest cost path, *i.e.*, the path with the fewest number of hops or stops in a network from a designated bridge to the **root bridge**. McNair, ¶¶62-70.

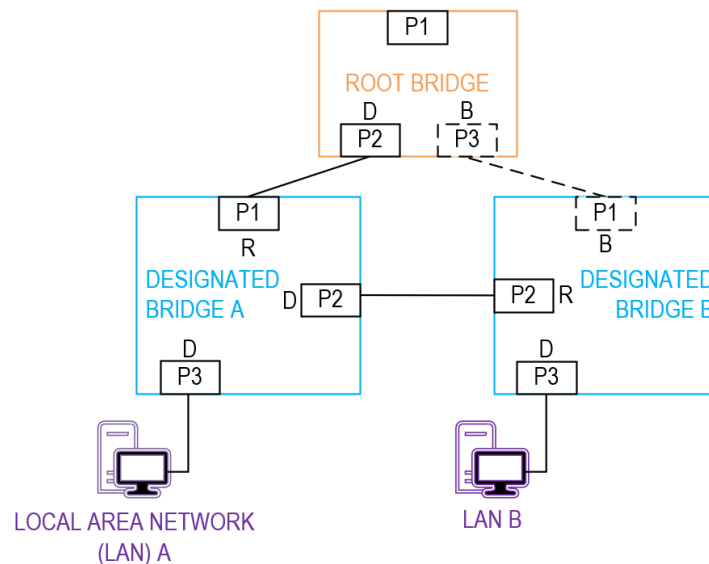


Figure 3: Roles of bridges and their ports in STP

Ports P1 of **designated bridge A** and P2 of **designated bridge B** above are root ports (“R”). Ports P2 of the **root bridge**, P2 and P3 of **designated bridge A**, and P3 of **designated bridge B** are designated ports (“D”). A designated port can connect to a LAN or to another bridge’s root port. Ports P3 of the **root bridge** and P1 of **designated bridge B** are alternate or backup ports, called “blocked ports” (“B”) which receive but do not forward messages. It would still comply with IEEE 802.1D for the root bridge’s port P3 to be a designated port since any messages sent therefrom would be discarded at port P1 of **designated bridge B**. McNair, ¶¶71.

3. STP Bridge Port States

The STP algorithm also assigns to ports a state: forwarding, learning, listening, and blocking. A port in the forwarding state receives and forwards data messages to other bridges. In Figure 3 above, the root ports and the designated ports are in the forwarding state. McNair, ¶¶72-73.

A port in the blocking state receives but does not forward messages. In Figure 3, ports P3 of the **root bridge** and P1 of **designated bridge B** are in the blocking state (though, as discussed above, port P3 of the root bridge could be in the forwarding state). Upon detection of a failure at a port in the forwarding state, one of the back-up ports can be transitioned from the blocking state to the forwarding state as explained immediately below. McNair, ¶¶74-75.

4. Maintaining STP Using Bridge Databases

In STP, a bridge stores information it learns about the spanning tree in a filtering database. A bridge's filtering database determines which port to use when forwarding a data message with a given destination media access control (MAC) address. The filtering database maps destination MAC addresses to ports (*i.e.*, a root port or a designated port) through which a message to that MAC address should be sent. McNair, ¶¶76-78.

If a bridge receives a data message having a destination MAC address in its filtering database, the bridge forwards the data message using the bridge's port identified in the filtering database that corresponds to the destination MAC address. If a bridge receives a data message having a destination MAC address not identified in its filtering database, the bridge updates its filtering database to include a new mapping between the received port and the source MAC address such that the bridge can use the new mapping when it receives future data messages directed to the source of the data message. Additionally, the bridge forwards the data message to another bridge, which causes the receiving bridge to update its own filtering database as the transmitting bridge did. McNair, ¶79.

B. STP Self-Healing Process

Links between nodes in a computer network may fail, which can isolate portions of the network. To mitigate the effects of a link failure, the IEEE 802.1D standard defines STP to carry out self-healing recovery mechanisms to reconnect network portions that are disconnected by a link failure. In general, an STP bridge recovers from a link failure detected at its root port by selecting a different port, *e.g.*, a blocked port, to be a candidate new root port. STP causes the previously-blocked port to transition through states (*i.e.*, from the blocking state to the forwarding state) and, if successful, selects the previously-blocked port to be the new active root port. Network operation resumes by the bridge receiving and forwarding data messages through the new root port. McNair, ¶80.

1. Link Failure Detection

STP's self-healing recovery mechanisms use a Bridge Protocol Data Unit (BPDU). A BPDU is a control message that is periodically transmitted by bridges to maintain the spanning tree by monitoring for network changes such as link failures. In a default STP configuration, a BPDU is transmitted every 2 seconds. A bridge detects a link failure at its root port when the bridge does not receive a BPDU on its root port in an expected time frame (*i.e.*, 20 seconds by default). McNair, ¶¶81-85.

2. Link Failure Recovery

According to STP, when a bridge detects a link failure, the bridge triggers a process to failover to a new root port, which is shown in Figures 4-5 below.

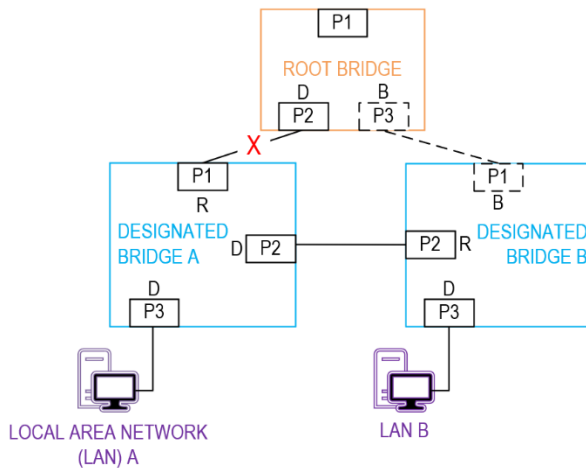


Figure 4: STP link failure

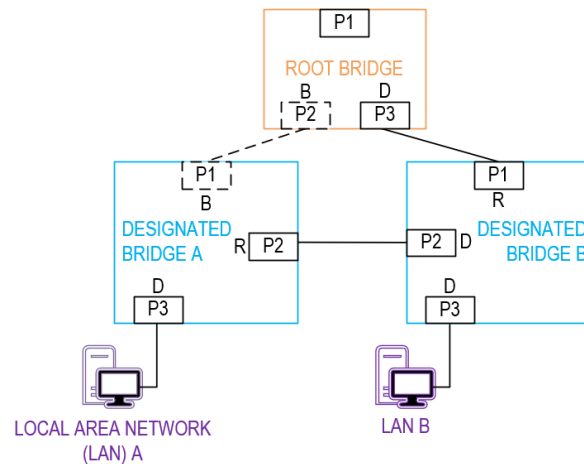


Figure 5: STP link recovery

In Figure 4 above, **designated bridge A** detects a failure on its root port (identified by a red “X” through the link between that root port and the designated port on the **root bridge** upstream) because it does not timely receive a BPDU on its root port. As shown in Figure 5, responsive to the failure, **designated bridge A** transitions its port P1, through intermediate states, into the blocking state, and marks its port P2 as a candidate new root port. Placing port P1 into the blocking state deactivates the previously-active link. The newly-deactivated link is shown in Figure 5 as a dashed line between the **root bridge** and **designated bridge A**.

McNair, ¶¶87-89.

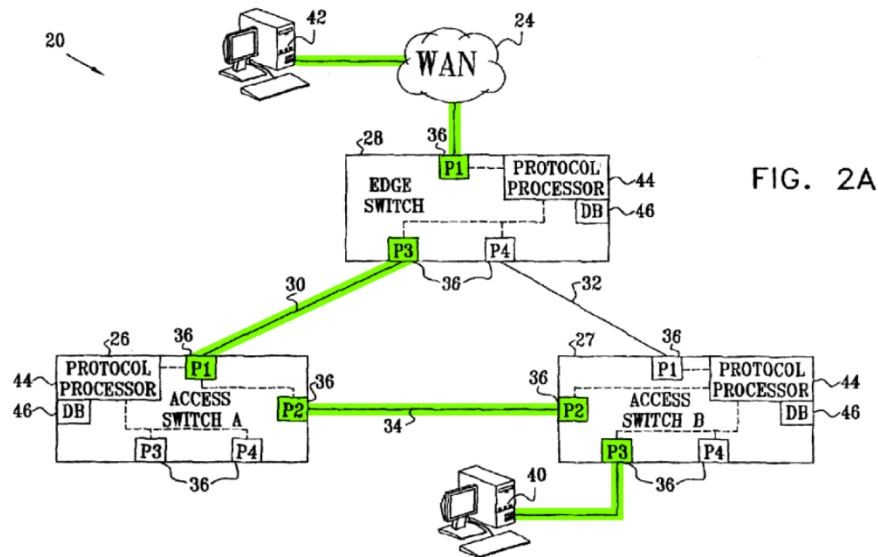
Designated bridge A transmits BPDUs via its port P2 to notify designated bridge B of the change. Designated bridge B transitions its port P1 from the blocking state, through intermediate states, to the forwarding state, activating the previously-inactive link between designated bridge B and the root bridge (solid line in Figure 5). Network operation then resumes, with designated bridge B receiving and forwarding messages on the newly-active link. The overall delay from designated bridge A receiving the last BPDU until reconfiguration is lengthy (approximately 50 seconds), frustrating timely communication. McNair, ¶90.

V. THE '986 PATENT

The '986 patent purports to provide “rapid reconfiguration in response to network topology changes” (3:58-60) in Spanning Tree networks. 8:1-9.² In operation, a link is placed in an active state via connection to an active port of a network switch. 8:10-13. An active port is placed in a forwarding state in which the port receives and transmits data frames. 2:39-41. A link that is redundant to the active link is connected to a port in a blocked state until the active link connected to the active port fails. 9:26-28. Following link failure, the '986 patent purports to reconfigure a network more quickly than standard STP self-healing by immediately activating a back-up port and using a dummy frame mechanism. 3:1-14, 4:13-31, 10:16-24. McNair, ¶103.

² Citations in Section V are to the '986 patent unless otherwise noted.

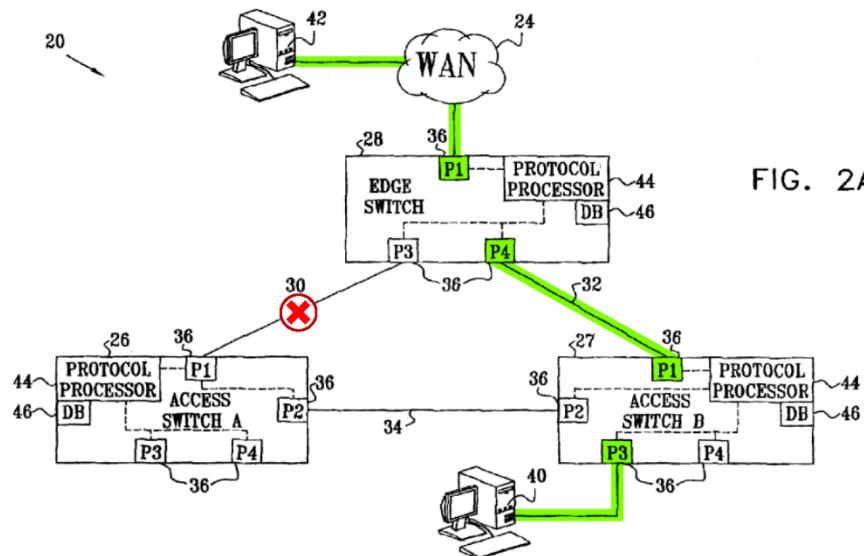
The network path for data frame forwarding prior to link failure is highlighted in green in an annotated version of Figure 2A, below.



Annotated Figure 2A of the '986 patent before link failure

When a failure is detected in an active link (*e.g.*, link 30), one of the blocked ports (*e.g.*, port P4) of the edge switch is chosen to serve as the new active link and the failed port (*e.g.*, port P3) is blocked. 9:26-35. So-called dummy frames are sent from the edge switch through the new active link (*e.g.*, link 32) to downstream access switches (*e.g.*, access switch 27). 9:42-44. Upon receiving dummy frames, the receiving access switches (*e.g.*, access switch 27) use the learning algorithm specified in the IEEE 802.1D standard (described *supra* §IV.A.4) to rebuild their filtering databases, which are used to forward data frames, so that upstream traffic will be directed to the new active link. 9:62-10:4. McNair, ¶¶105-106.

Responsive to the detection of the failed link, the updated network path for data frame forwarding is highlighted in green in the annotated version of Figure 2A below. The previously-active link between the edge switch and the downstream access switch A, now blocked after detecting the failure, is identified by a red X. The edge switch and the downstream access switch B exchange traffic over the newly-active link (e.g., link 32). McNair, ¶107.



Annotated Figure 2A of the '986 patent after link failure

Because various time-consuming aspects of STP are not used, the self-healing process of the '986 patent is not subject to the 50-second delay of STP. By using a dummy frame routing update, the '986 patent asserts that its self-healing process “can be completed in well under a second, after which normal operation of the network can resume.” 10:22-24. McNair, ¶108.

A. The Challenged Claims

The challenged claims are reproduced in the Claim Listing (*infra* §X).

Independent claim 1 is reproduced below with labeling added for reference.

1[PRE] A method for network communication, comprising:

- [1A]** coupling an edge switch in a network to one or more access switches via at least first and second redundant links, respectively connected to first and second ports of the edge switch;
- [1B]** placing the first port in an active state, while placing the second port in a blocking state;
- [1C]** conveying communication traffic over the first link while the first port is in the active state, thereby causing each of the switches to build a respective database for use in forwarding the traffic;
- [1D]** responsive to a failure associated with the first link, placing the second port in the active state and the first port in the blocking state; and
- [1E]** sending dummy traffic from the edge switch over the second link to the one or more access switches, so as to cause each of the one or more access switches to modify its respective database responsive to the second port being in the active state.

B. Prosecution History

The challenged claims were filed on January 7, 2002. EX1002, 1. The Patent Examiner issued no Office Action on the merits. The challenged claims were granted in U.S. Patent No. 6,917,986 on July 12, 2005. IEEE 802.1D, published in 1998 (EX1007; EX1008, 5), was of-record. EX1001, 1.

C. Claim Interpretation

Claim terms are construed herein according to their ordinary meaning in light of the specification and file history. 37 C.F.R. §42.100(b). Because the prior art discloses each element of the challenged claims under any plausible construction, no outer-boundary constructions are necessary. *Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co.*, 868 F.3d 1013, 1018-19 (Fed. Cir. 2017).

D. Level of Ordinary Skill in the Art

A person of ordinary skill in the art (POSA) with respect to the '986 patent would have a bachelor's degree in electrical engineering or a related field, plus at least two years of industry experience or graduate-level academic experience with computer networking. McNair, ¶111.

VI. TABLE OF GROUNDS

Petitioner requests cancellation of claims 1-24 as follows.

Ground	Reference(s)	Claims	Basis
1	Gai in view of IEEE 1999	1-3, 6-15, 18-24	§103
2	Gai in view of IEEE 1999 and Jain	4-5, 16-17	§103

VII. UNPATENTABILITY GROUNDS

A. Ground 1: Claims 1-3, 6-15, and 18-24: Obvious Over Gai in View of IEEE 1999

1. U.S. Patent No. 6,032,194 (“Gai,” EX1005)

Gai is prior art to the '986 patent under pre-AIA 35 U.S.C. §102(b) because it was granted (and hence published) on February 29, 2000, more than a year before the '986 patent's filing on January 7, 2002.

Gai, like the '986 patent, is directed to rapidly reconfiguring a Spanning Tree network. EX1005, 5:35-39. Gai recognizes several disadvantages of the STP algorithm including that, during the time-consuming reconfiguration process, message delivery is delayed, which network users may perceive as a service disruption, presenting significant problems on networks intended to be highly reliable. EX1005, 4:66-5:7. Additionally, following the reconfiguration process, BPDUs may flood the network, thereby consuming substantial network resources. EX1005, 5:7-13. Gai addresses these issues by immediately transitioning port status and utilizing a dummy packet routing update to quickly inform the network of newly-activated ports, without the delays related to STP's use of BPDUs and STP's requirement that ports transition through multiple states. EX1005, 6:1-5, 14:54-63. McNair, ¶113.

a. Port Reconfiguration After Link Failure

After detecting a link failure, Gai and the '986 patent both reconfigure their ports in the same manner.³ EX1001, 3:26-35 (describing Gai's operation); 4:14-30 (describing the '986 patent's operation). In Gai, a **downstream switch** detects failure of an **active link 128** at an active port (*i.e.*, port 3), which is illustrated below in an annotated version of Gai's Figure 1. McNair, ¶122.

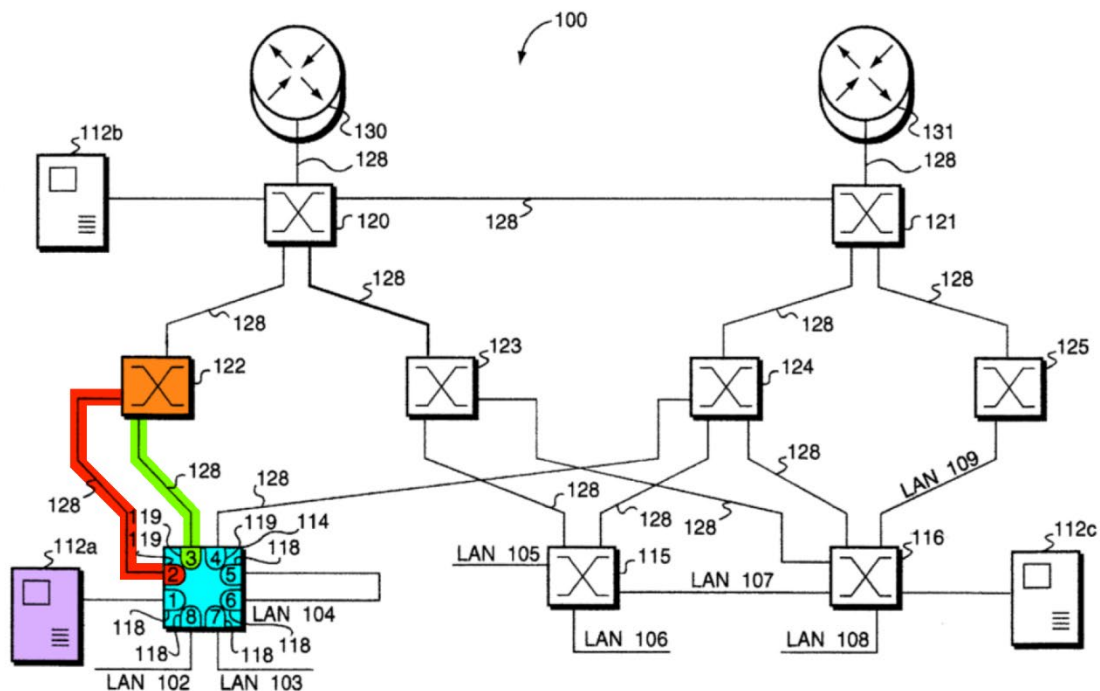
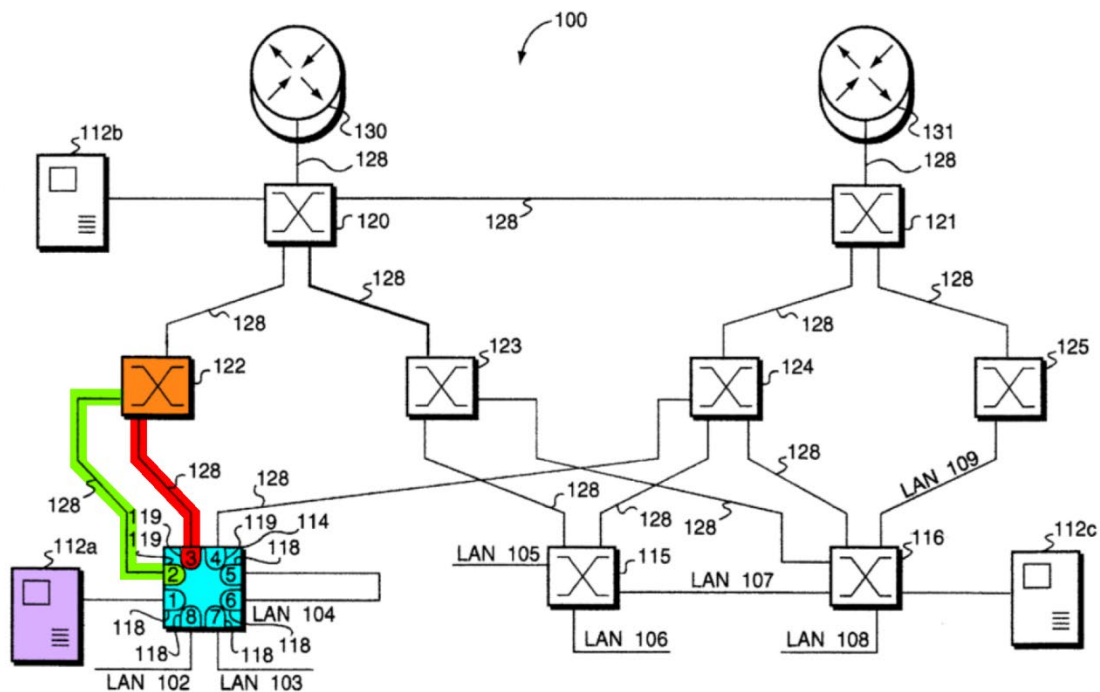


FIG. 1

Annotated Figure 1 of Gai before link failure

³ The '986 patent incorporates Gai for this very disclosure. EX1001, 3:15-35.

Below, another annotated version of Figure 1 shows port reconfigurations of the **downstream switch** after link 128 fails. After detecting the failure, the **downstream switch** places the previously-active port in a blocking state, changing previously-active link 128 to an **inactive link**. EX1005, 14:40-51. The previously-blocked port is placed in an active state, changing the previously-inactive link 128 to an **active link**. EX1005, 11:48-52. Like the '986 patent, by immediately transitioning the ports' states without progressing through intermediary states (*e.g.*, listening and learning STP states), Gai reduces the time needed to effectuate the network topology change. EX1005, 5:51-53, 6:50-53. McNair, ¶123.

**FIG. 1**

Annotated Figure 1 of Gai after link failure

b. Dummy Traffic

Gai and the '986 patent both transmit dummy traffic to notify other switches of port configuration changes. EX1001, 3:29-35 (describing Gai's operation), 4:17-28 (describing the '986 patent's operation). To notify upstream switches, Gai's **downstream switch** transmits dummy traffic upstream through the **new active link** to update the filtering database of the **upstream switch**. EX1005, 12:56-66. In summary, Gai discloses setting up a spanning tree network, performing port reconfiguration after link failure, and transmitting dummy traffic, all as described in the '986 patent. McNair, ¶¶104, 124-125.

c. Network Initialization Commands

The '986 patent purports to differentiate itself from Gai by not requiring, as part of the network self-healing process, certain network initialization commands discussed in Gai. EX1001, 3:35-39, 4:36-43. However, POSAs would recognize that Gai's self-healing technique does not require these commands, and that Gai discloses them merely as an example implementation for pre-configuring a network into an optimized spanning tree structure to streamline self-healing in the event of a link failure by taking into consideration server loads in a network, something not addressed in the STP algorithm. EX1005, 8:59-65, 10:29-36. Even setting aside this optimization teaching, the '986 patent's claims do not exclude such commands in any event. McNair, ¶¶126-131.

2. IEEE 1999 (EX1009)

IEEE 1999 is prior art under pre-AIA 35 U.S.C. §102(b) because it was published in March 1999 and accessible to POSAs, including by keyword searching, no later than May 1999—more than a year before (indeed, almost three years before) the '986 patent was filed. EX1010; EX1011; McNair, ¶132.

IEEE 1999 addresses the same challenges as Gai and the '986 patent with respect to the same IEEE 802.1D standard. EX1009, Abstract (paper discloses a “fast spanning tree reconfiguration algorithm...a significant improvement over the standard IEEE 802.1d implementation”), 161, 162, 164, 165 (the “fault recovery mechanism has to be fast, so that the affected connection can be reestablished with minimal interruption in service”). McNair, ¶¶100, 133.

IEEE 1999 explains that switches both upstream and downstream of a link failure detect the failure. EX1009, 166 (“the fault would be independently detected by switches both upstream as well as downstream of the fault”). Each switch that detects a failure takes steps to rebuild the spanning tree. EX1009, 166 (“The fault recovery mechanism on each switch that detects the fault...[r]ebuild[s] the spanning tree”). As part of the spanning tree reconfiguration, to improve recovery time, each switch that detected a link failure sends “teardown messages” to both upstream and downstream switches to “terminate the connections through a failed link.” EX1009, 167. McNair, ¶¶101-102, 134.

POSAs would have understood that a switch that receives a “teardown message” updates its filtering database such that the failed connection is terminated. Thus, IEEE 1999 discloses to POSAs that, to improve recovery time, a switch, responsive to detecting a failure upstream or downstream of itself, sends messages both upstream and downstream to cause other switches to update their respective filtering databases. McNair, ¶135.

3. Gai in View of IEEE 1999

The '986 patent admits Gai discloses sending dummy traffic upstream from a switch that detects a link failure on its root port. EX1001, 3:26-33. Gai discloses sending dummy traffic to reduce link failure recovery time. EX1005, 6:42-53, 15:21-25. Aside from the network initialization commands discussed above (which, as noted above, are not excluded by the '986 patent's claims), the '986 patent purports to distinguish Gai by sending dummy traffic downstream, rather than upstream as in Gai. EX1001, 4:35-43. McNair, ¶136.

In view of IEEE 1999, POSAs would have had reason to modify Gai's technique by sending Gai's dummy traffic downstream after a downstream link failure (as the '986 patent describes), in addition to sending dummy traffic upstream after an upstream link failure (as Gai describes). This has the same benefit already described in Gai, *i.e.*, reducing the time needed for network self-healing. EX1005, 6:42-53, 15:21-25. McNair, ¶137.

As noted above, IEEE 1999 discloses sending such messages both upstream and downstream, after detecting a failure either upstream or downstream, for the same purpose that Gai describes sending dummy traffic: increasing the speed of self-healing following link failure in an 802.1D network. *Supra* §VII.A.2. In view of IEEE 1999, POSAs had reason to use Gai's technique when a failure is detected downstream of a switch, just as when a failure is detected upstream of a switch. While IEEE 1999 describes the messages as "teardown messages," their function is the same as Gai's "dummy messages." McNair, ¶¶134-135, 138.

Gai discusses two types of switches: access switches and backbone switches. "Access switches are preferably coupled to entities (e.g., LANs, end stations, etc.) whereas backbone switches provide the interconnections between access switches." EX1005, 6:9-11. Gai discloses entering an Enable_Uplinkfast command at an access switch (e.g., access switch 114) to block a failed port (e.g., the root port) and configure the access switch to send dummy traffic upstream through a new active port. EX1005, 6:39-46. When modified in view of IEEE 1999's disclosure that self-healing network traffic can advantageously be sent both upstream and downstream, the command would be entered at a backbone switch (e.g., backbone switch 122) to block a failed port (e.g., a designated port connected to access switch 114) and configure the backbone switch to send dummy traffic downstream to an access switch (e.g. access switch 114) through a new active port.

While Gai uses its protocol signaling command, Enable_Uplinkfast, to send dummy traffic through a new root port, POSAs would deem it an obvious implementation detail to modify the parameters of the command to be applicable to other ports, such as a designated port on a backbone switch. McNair, ¶139.

Claim 1 of the '986 patent requires an “edge switch” to send dummy traffic downstream to an access switch. Gai’s backbone switch 122 is an “edge switch.” Per the '986 patent, “[c]ore switches interconnect the access switches into a Wide Area Network (WAN) by direct connection between the core switch and the access switches and/or by connection to other core switches.” EX1001, 1:33-36. Gai’s backbone switch 122 is a “core switch,” as it connects directly to access switch 114 and, via another backbone switch 120, to router 130, which POSAs would have understood connects to a WAN. EX1005, Fig. 1; McNair, ¶140. The '986 patent further explains: “Core switches that are directly connected to access switches are referred to herein as edge switches.” EX1001, 1:36-38. Because Gai’s backbone switch 122 connects directly to access switch 114, it is an “edge switch.” McNair, ¶141. Further, the '986 patent admits that Gai’s backbone switches (which the '986 patent calls “trunk switches”) are “comparable to the core and edge switches” of the '986 patent. EX1001, 3:21-24. In short, the modified Gai technique involves an “edge switch” sending dummy traffic downstream to an “access switch” as claim 1 requires. McNair, ¶142.

Further, some claims of the '986 patent (*e.g.*, claim 12) use the term “bridge” instead of “edge switch” and “access switch.” “Bridge” is a general term encompassing both edge and access switches. EX1001, 1:26-38 (“[t]he most common type of MAC bridge is a switch...Switches can be roughly classified as access switches or core switches...”). Thus, for the same reasons described above, the modified Gai technique involves a “bridge” that sends dummy traffic downstream to another “bridge” as required, for example, by claim 12 of the '986 patent. McNair, ¶¶54, 143.

In short, when modified in view of IEEE 1999’s disclosure that it was advantageous to send self-healing messages both upstream and downstream after a link failure either upstream or downstream, Gai’s backbone switches (*i.e.*, the claimed “edge switches” and “bridges”) send Gai’s dummy traffic both upstream and downstream, regardless of whether a link failure is detected upstream of the switch or downstream of the switch. This means that modified Gai’s backbone switch / edge switch / bridge sends dummy traffic downstream to Gai’s access switch / bridge, as recited in the challenged claims. Doing so achieves Gai’s goal (shared by IEEE 1999) of reducing the time taken to rectify network failures, including between an edge switch and an access switch. EX1005, 11:53-55, 11:64-12:12. McNair, ¶144.

Finally, a POSA would have had a reasonable expectation of success in modifying Gai to have its backbone switches / edge switches / bridges send dummy traffic both upstream and downstream because signaling protocols are routinely modified to provide for different control signaling, such that there would have been no additional technical challenge in sending those messages both upstream and downstream. Further, Gai's mechanism for sending dummy traffic (*i.e.*, rapid reconfiguration entity 234) is implemented by both its backbone switches and its access switches. *Infra* §VII.A.4.f. Since Gai's modified technique applies to backbone switches, which qualify as the claimed "edge switches" and "bridges," the combination of Gai with IEEE 1999 renders obvious the '986 patent's claims as explained below. McNair, ¶145.

4. Claim 1

a. [1PRE] A method for network communication, comprising:

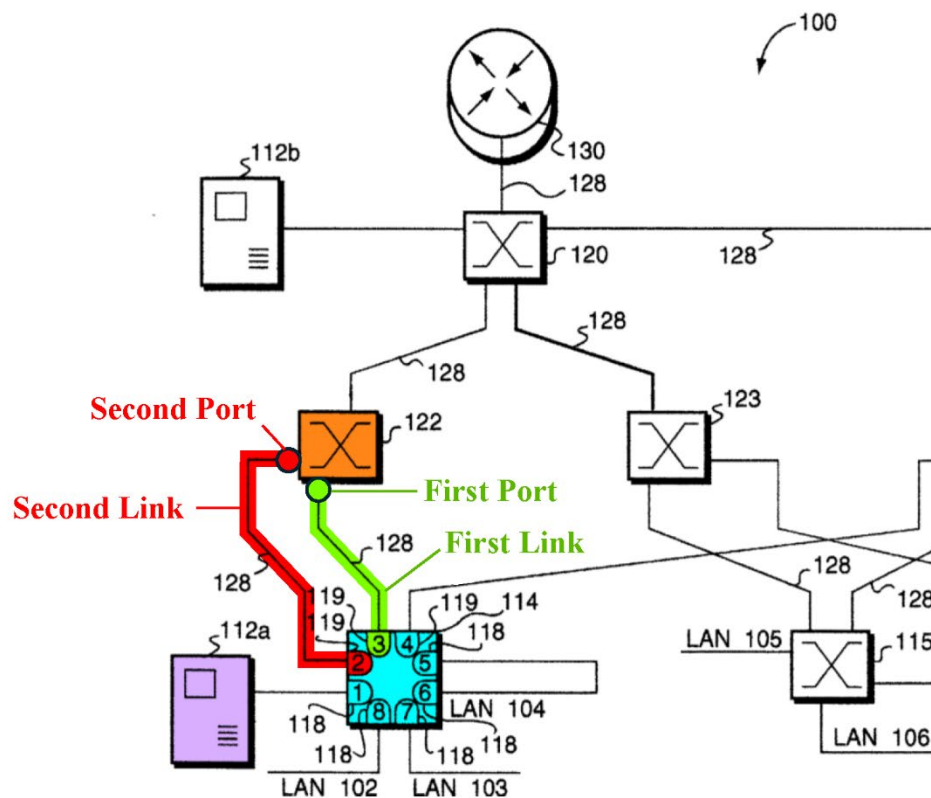
Gai discloses "a method...for rapidly configuring a computer network following a network change." EX1005, 1:15-17. Gai discloses "a method...for balancing message traffic across several links of a computer network." EX1005, 5:29-30. McNair, ¶147.

- b. [1A] coupling an edge switch in a network to one or more access switches via at least first and second redundant links, respectively connected to first and second ports of the edge switch;**

Gai discloses a computer network including backbone switches 120-125 and access switches 114-116. EX1005, 7:3-9; *see also* EX1001, 3:15-21 (acknowledging that Gai discloses “access switches”). The backbone switches 120-125 and the access switches 114-116 are interconnected through redundant communication paths that “prevent portions of the network 100 from becoming isolated should any constituent link or device fail.” EX1005, 7:53-57. Each of the backbone switches 120-125 and access switches 114-116 have multiple ports. EX1005, 7:34-37 (“[E]ach switch and router, such as access switch 114, identifies its local and trunk ports 118 and 119 by numbers (e.g., port number one, port number two, port number three, etc.).”), Fig. 1. McNair, ¶148.

The '986 patent explains: (i) “[c]ore switches interconnect the access switches into a Wide Area Network (WAN), by direct connection between the core switch and access switches and/or by connection to other core switches” and (ii) “[c]ore switches that are directly connected to access switches are referred to herein as edge switches.” EX1001, 1:33-38. McNair, ¶149.

As shown in annotated Fig. 1 of Gai below, **backbone switch 122** is an edge switch because it meets both (i) and (ii) above. **Backbone switch 122** meets (i), and thus is a “core switch,” because it interconnects **access switch 114** into a Wide Area Network (WAN) (implicitly or obviously connected to router 130) by direct connection between **backbone switch 122** and **access switch 114** and via backbone switch 120 and router 130. EX1005, 7:26-30, Fig. 1, 6:10-12. **Backbone switch 122** meets (ii), and thus is an “edge switch” as claimed, because it directly connects to **access switch 114**. McNair, ¶¶140, 150.



Annotated Figure 1 of Gai before link failure

Gai thus describes a network topology meeting [1A]. As shown in the annotated version of Gai's Figure 1 above, **backbone switch 122** (*i.e.*, a claimed "edge switch") is coupled to **access switch 114** via at least first and second redundant communication links 128, the "first" of which is an **active link** while the "second" of which is an **inactive link**. EX1005, 11:41-52. The **active link** is between port 3 of **access switch 114** and a "first" port of **backbone switch 122** (green circle). The **inactive link** is between port 2 of **access switch 114** and a "second" port of **backbone switch 122** (red circle). These ports on backbone switch 122 are disclosed and, at the very least, obvious. McNair, ¶¶86, 151.

In addition, the '986 patent specification effectively admits that Gai meets [1A]. EX1001, 3:18-24. The '986 patent specification describes that Gai's network has both core switches and access switches; some of which are called "trunk switches." The '986 patent admits "[t]hese trunk switches are comparable to the core switches and edge switches defined above." EX1001, 3:18-24. The '986 patent also describes that the Gai "manage[s] redundancies in their upstream ports" and that Gai's access and core switches have "multiple trunk ports providing connectivity to the STP root." EX1001, 4:40-42; 3:18-21. Accordingly, the '986 patent effectively admits that Gai discloses switches, such as edge and access switches, that are coupled through redundant links. McNair, ¶152.

c. [1B] placing the first port in an active state, while placing the second port in a blocking state;

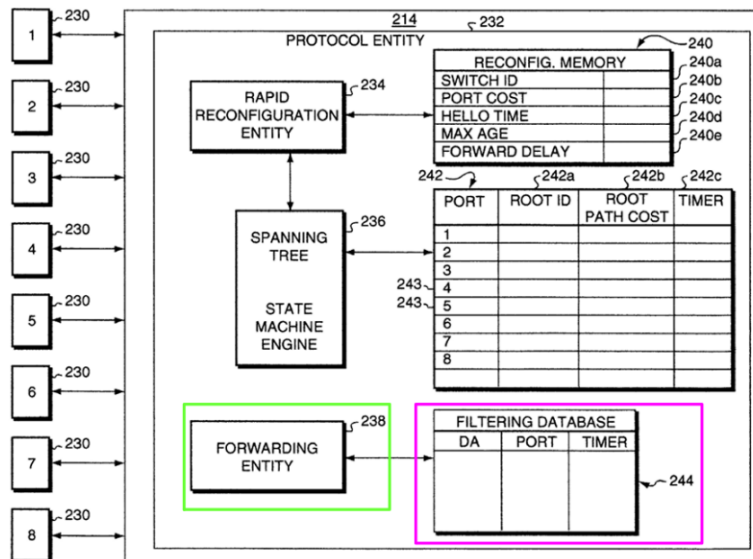
Gai explains that its switches place their ports in either a forwarding or blocking state. “The network preferably includes a plurality of devices executing the spanning tree algorithm so as to elect a root and place ports of the devices in either a forwarding or blocked state...one trunk port is in the forwarding state and the other trunk ports are in the blocked state.” EX1005, 5:36-43. With reference to the annotated version of Gai’s Figure 1 above, “port number three, which is coupled to **backbone switch 122**, is the root port for **switch 114** and thus in the forwarding state...and port numbers two and four, which also connect **switch 114** to the backbone switches (*i.e.*, **backbone switches 122** and 124), are in the blocked state.” EX1005, 11:42-48. The corresponding ports on backbone switch 122 are placed in corresponding states, or at least it would have been obvious to do so. By placing a port (*i.e.*, “first” port) of **backbone switch 122** that is connected to port 3 of **access switch 114** in the active state while placing a port (*i.e.*, “second” port) of **backbone switch 122** that is connected to port 2 of **access switch 114** in the blocking state, Gai meets [1B]. McNair, ¶¶153-154.

Further, the ’986 patent admits that Gai meets [1B]. EX1001, 3:24-26 (“One of the trunk ports of each switch is placed in the forwarding state, while the others are designated as back-up ports.”). McNair, ¶155.

- d. [1C] conveying communication traffic over the first link while the first port is in the active state, thereby causing each of the switches to build a respective database for use in forwarding the traffic;

Gai conveys communication traffic over the first link identified above (between port 3 on access switch 114 and a corresponding port on backbone switch 122) while those ports are in the active state. EX1005, 7:30-33. McNair, ¶156.

As discussed *supra* §IV.A.4, databases for forwarding traffic are known from conventional STP. EX1007; EX1005, 1:50-58. Gai discloses such databases as shown in a pink box in an annotated version of Gai's Figure 2 below. Figure 2 is a partial block diagram of a switch 214 (which "corresponds to access switch 114 of Fig. 1," EX1005, 7:62-64), including filtering database 244, "configured to store address information corresponding to the entities of network 100." EX1005, 7:62-64; 8:31-33. McNair, ¶157.



Annotated Figure 2 of Gai

Gai discloses building **filtering database 244** using information from traffic received at a port in an active state (*i.e.*, a forwarding state). In operation, ports 230 receive messages (*i.e.*, traffic) and pass them along to **forwarding entity 238**. EX1005, 11:16-18. **Forwarding entity 238** looks up the destination address (“DA”) of the messages in the **filtering database 244**. If the destination address is found in the **filtering database 244**, a port (*e.g.*, port 4) corresponding to the destination address is identified. The message is switched out on the port (*e.g.*, port 4). EX1005, 11:19-23. If the destination address is not found in **filtering database 244**, the **forwarding entity 238** looks up the source address in **filtering database 244**. If the source address is not found in **filtering database 244**, a new entry is formed in **filtering database 244** (*i.e.*, building **filtering database 244**). EX1005, 11:24-33. McNair, ¶158.

Gai discloses that each of its switches, including its backbone switches (hence the claimed “edge switch”), include filtering databases. EX1005, 1:53-54 (“This information is then stored by each switch in a block of memory referred to as a filtering database.”), 13:67-14:2 (“upon receipt of these multicast messages 400, the other devices within network 100 update their corresponding filtering databases”). Thus, limitation [1C] is satisfied. McNair, ¶159.

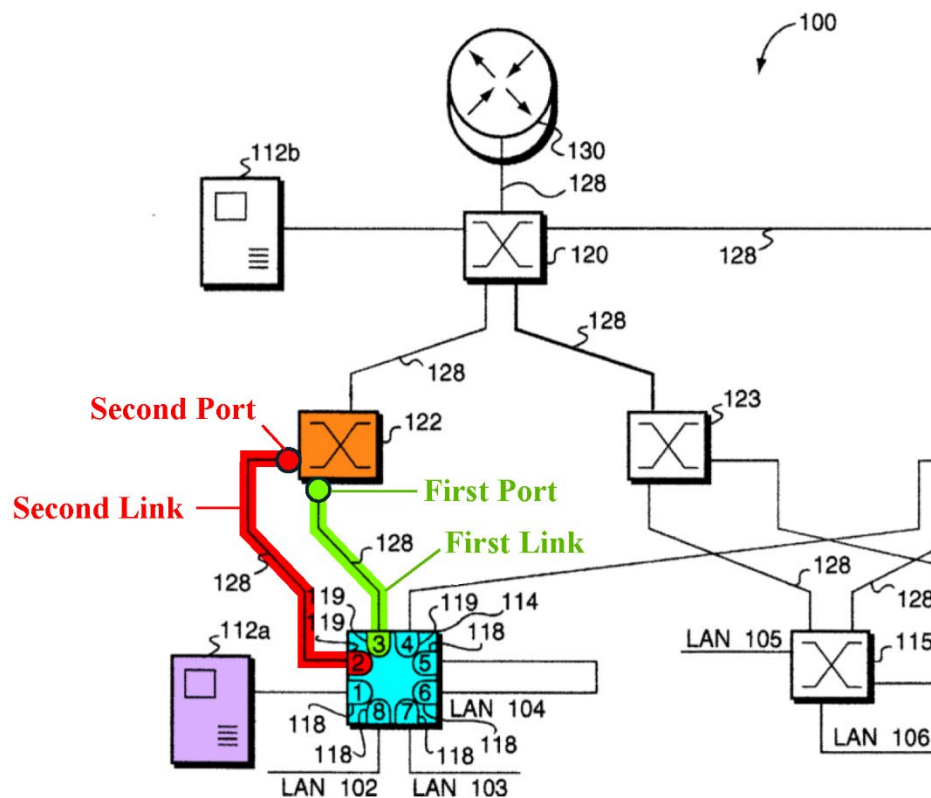
- e. **[1D] responsive to a failure associated with the first link, placing the second port in the active state and the first port in the blocking state; and**

The '986 patent admits that Gai discloses detecting a failure associated with a link and thereafter placing a back-up port in the forwarding state to become the new active port. EX1001, 3:26-29 (“Upon detection of a failure at the active forwarding port, the switch immediately places one of the back-up trunk ports in the forwarding state, so that it becomes the new active port.”). McNair, ¶160.

The '986 patent's admission is consistent with Gai's own disclosure. Gai discloses that “access switches 114-116 and backbone switches 120-125 detect failures.” EX1005, 11:64-65. Switches in Gai can detect failures using two different techniques. First, a switch (*e.g.*, an access switch or a backbone switch) identifies a link failure “[i]f a switch stops receiving BPDUs on its root port.” EX1005, 4:1-3. Second, a switch identifies a link failure by performing a link integrity test that “typically exchanges test messages across the respective link 128 at a relatively high rate (*e.g.*, every ten milliseconds).” EX1005, 12:4-9. POSAs would have understood that a failed link between a backbone switch (*e.g.*, backbone switch 122) and an access switch (*e.g.*, access switch 114) would be recognized both by the backbone switch (as a downstream link failure) and by the access switch (as an upstream link failure). McNair, ¶161.

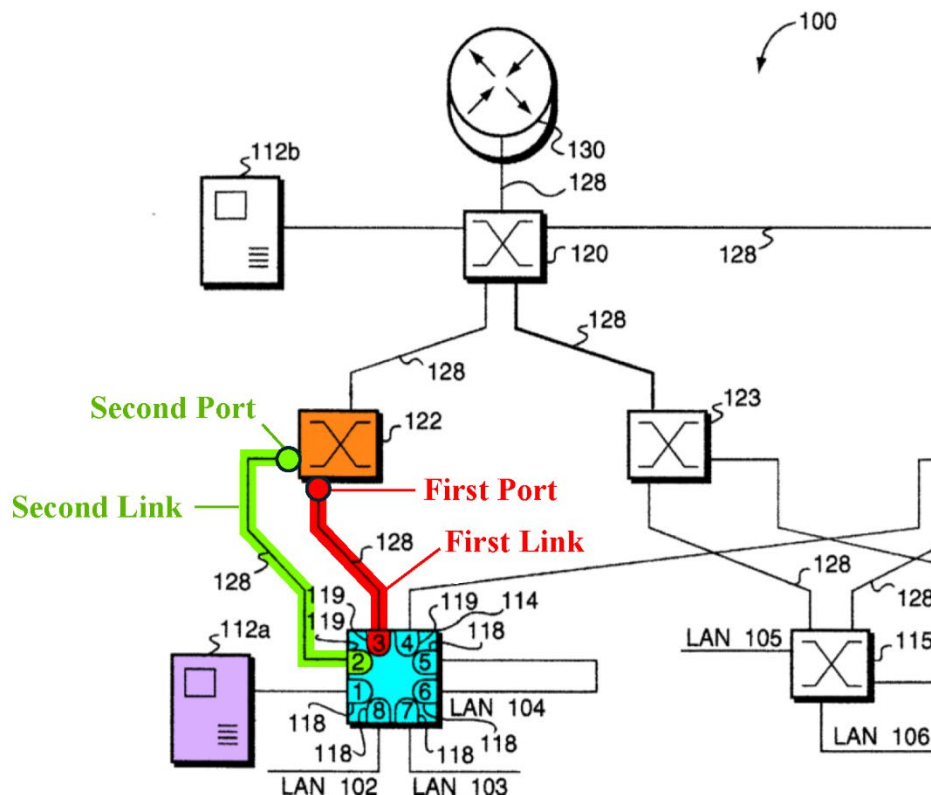
Gai discloses port reconfiguration responsive to a link failure. In an annotated version of Gai's Figure 1 below showing the configuration before link failure, port 3 of **access switch 114** is in the forwarding state and is connected to a corresponding first (active) port (identified by a green dot) of **backbone switch 122** via an **active link**, which is the “active link” of [1D]. Port 2 of **access switch 114** is in the blocking state and is connected to a corresponding second (blocked) port (identified by a red dot) of **backbone switch 122** via the **inactive link**.

McNair, ¶162.



Annotated Figure 1 of Gai before link failure

After detecting a failure of the **active link**, **access switch 114** places its port 2 in the forwarding state and its port 3 in the blocking state. EX1005, 11:48-52; 14:40-51. The inactive (red) link shown above thus becomes the new **active link**, which is green below in another annotated version of Gai's Figure 1. The first port on **backbone switch 122** (connected to port 3 on **access switch 114**) is placed in the blocking state (identified below by a red dot), and the active link shown above becomes an **inactive link** in red below. The second port on the **backbone switch 122** (connected to port 2 on **access switch 114**) is placed in the forwarding state (*i.e.*, the active state) (identified below by a green dot). McNair, ¶163.



Annotated Figure 1 of Gai after link failure

As explained above in [1A], **backbone switch 122** qualifies as the claimed “edge switch.” As explained immediately above in this subsection, **backbone switch 122** can detect a failure associated with the **active link** and, responsive to such a failure, **backbone switch 122** places a port previously in the blocking state (*i.e.*, its second port as identified above with respect to [1B]) in the forwarding state and places another port previously in the active state (*i.e.*, its first port as identified above with respect to [1B]) in the blocking state, meeting and/or rendering obvious [1D]. *Supra* §§VII.A.4.b, VII.A.4.c, McNair, ¶164.

- f. **[1E] sending dummy traffic from the edge switch over the second link to the one or more access switches, so as to cause each of the one or more access switches to modify its respective database responsive to the second port being in the active state.**

The '986 patent and Gai send the same dummy traffic between switches. The '986 patent admits that Gai discloses sending dummy multicast messages (*i.e.*, dummy traffic) upstream using a root port after detection of a link failure and, “[u]pon receiving these messages, other devices in the network learn to use the new active port.” EX1001, 3:26-35. McNair, ¶165.

Gai’s dummy multicast messages meet [1E]’s “dummy traffic.” Gai’s Figure 4 (reproduced below) shows a dummy multicast message 400 that “includes a destination address (DA) field 410 and a source address (SA) field 412 that may be compatible with the Media Access Control (MAC) layer of the protocol stack.”

EX1005, 13:1-4. Gai discloses generating a dummy multicast message by “load[ing] SA field 412 with the address of an entity directly coupled to switch 114.” EX1005, 13:14-16. The address of an entity directly coupled to switch 114 would be in the filtering database of switch 114. EX1005, 8:31-39 (“filtering database 244 configured to store address information corresponding to the entities of network 100”). Loading SA field 412 in this way matches the ’986 patent, which describes “[f]or every entry in the filtering database of the edge switch that is not associated with the failed port...the edge switch sends a dummy frame...with a source MAC address equal to the destination MAC address listed for that entry.” EX1001, 9:44-49. Put another way, the source MAC address of the dummy frame in the ’986 patent is one of the destination MAC addresses in a filtering database, which is the same as Gai loading an address from a filtering database into SA field 412. Thus, a source address of Gai’s dummy multicast message is the same as the source address of the dummy frame of the ’986 patent. Gai also discloses loading the DA field “with a multicast address that causes message 400 to be received by all devices (e.g., access switches 114-116 and backbone switches 120-125) within the network 100.” EX1005, 13:9-13. This matches the ’986 patent, which says “[t]he destination MAC address of the dummy frame is a multicast address or other special address selected to ensure that the dummy frame is flooded to all the switches downstream from the edge switch.” EX1001, 9:53-56. McNair, ¶166.

In summary, the dummy multicast message disclosed in Gai meets [1E]’s dummy traffic because both are data messages that are loaded with source and destination addresses in the same manner. McNair, ¶167.

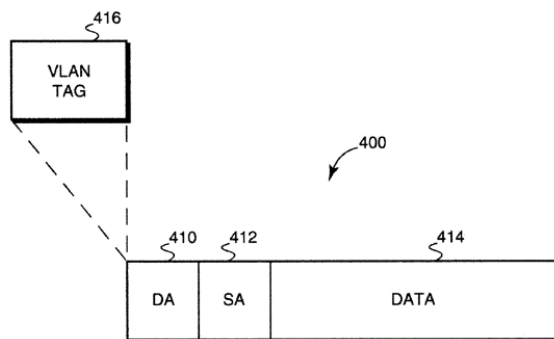


FIG. 4

Figure 4 of Gai

Gai discloses that dummy multicast messages can be sent by any switch. Figure 2 of Gai (annotated version reproduced below) shows a switch 214 that corresponds to access switch 114. EX1005, 7:62-64. McNair, ¶168.

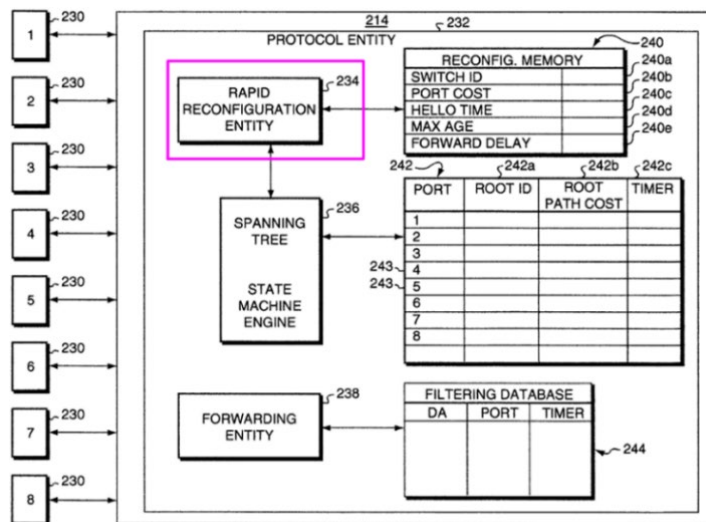


FIG. 2

Annotated version of Figure 2 of Gai

The switch implementation shown in Gai's Figure 2 includes a **rapid reconfiguration entity 234**. EX1005, 8:12-14. **Rapid reconfiguration entity 234** is Gai's dummy multicast message mechanism that informs other devices of a new forwarding port by transmitting dummy multicast messages through the new port. EX1005, 12:59-66. **Rapid reconfiguration entity 234** is also implemented by at least backbone switch 120. EX1005, 8:66-9:3 ("the rapid reconfiguration entity 234 of the selected backbone switch 120"). Therefore, the switch implementation shown in Figure 2, which generates and transmits dummy multicast messages using rapid reconfiguration entity 234, also applies to backbone switch 120. Thus, POSAs would have understood that the switch implementation shown in Figure 2 of Gai applies to any switch in Gai, or at least that it would have been obvious to use it with any switch in Gai. Therefore, POSAs would have understood that both access switches and backbones switches in Gai can generate and transmit dummy multicast messages. McNair, ¶169.

The purpose and benefit of Gai's dummy multicast message and the dummy traffic in [1E] are the same. Gai discloses that "[u]pon transition to the new forwarding port, the device begins transmitting 'dummy' multicast messages through the new port" (*i.e.*, "responsive to the second port being in the active state," as claimed). EX1005, 5:54-56. McNair, ¶170.

Gai further discloses that through dummy multicast messages, “other devices within the network learn to utilize the new forwarding port, rather than the failed port, when directing messages to these entities”—reconfiguring the network “substantially sooner than the time required by the conventional spanning tree algorithm.” EX1005, 5:65-6:5; 6:50-54. In Gai, when other devices within the network receive dummy multicast messages, they enter this new information in their respective filtering databases. EX1005, 13:34-38. This matches the ’986 patent, which says “[u]pon receiving these dummy frames, the access switches use the standard MAC learning algorithm to rebuild their own filtering databases so that upstream traffic is immediately directed to the new active link,” purportedly completing network reconfiguration in less than one second. EX1001, 4:24-30. McNair, ¶171.

Gai discloses sending dummy multicast messages to upstream devices after a link failure. As an example, Gai discloses [access switch 114](#) selecting port 4 as a new root port after a link failure, as shown in annotated Figure 1 of Gai below. [Access switch 114](#) then sends a dummy multicast message with a source address of server 112a through a new [active link](#) to [backbone switch 124](#). EX1005, 13:14-23. [Backbone switch 124](#) updates its filtering database by replacing previous information for server 112a with information referencing new port 4 of [access switch 114](#). EX1005, 13:34-38. McNair, ¶172.

In summary, as explained above in §VII.A.3, when modified in view of IEEE 1999's disclosure that it was advantageous to send self-healing messages both upstream and downstream, Gai's backbone switches send Gai's dummy traffic both upstream and downstream, regardless of whether a link failure is detected upstream or downstream of the switch. Thus, modified Gai's backbone switch (the claimed "edge switch") sends dummy traffic downstream to Gai's access switch, causing the access switch to update its filtering database. These effects occur in response to the second port being placed in the active state, after a link failure is detected, thus satisfying every portion of [1E]. McNair, ¶174.

In short, Gai renders obvious claim 1 in view of IEEE 1999. McNair, ¶175.

5. Claim 2: A method according to claim 1, wherein placing the second port in the blocking state comprises configuring the first and second ports so that no loop is created in the network between the edge switch and the one or more access switches.

Gai discloses switches that execute a spanning tree algorithm, which calculates a spanning tree that is loop-free. EX1005, 2:30-41; 7:59-61. For each switch, only one root port is placed in the forwarding state while the remaining root ports on that switch are blocked. EX1005, 6:33-38; 11:8-13. Accordingly, for an access switch that is connected to a backbone switch, only one access switch port connected to the backbone switch is in the forwarding state while the other access switch ports connected to the backbone switch are blocked.

Gai explains that configuring only one port on each access switch for connectivity to the backbone switch avoids loops. EX1005, 12:43-52. McNair, ¶176.

As explained above in [1B] (*supra* §VII.A.4.c), Gai discloses, or at least renders obvious, placing the “first” port of **backbone switch 122** that is connected to port 3 of **access switch 114** in the active state while placing the “second” port of **backbone switch 122** that is connected to port 2 of **access switch 114** in the blocking state. Among the designated ports of backbone switch 122 that are connected to **access switch 114**, only the designated port that is connected to root port 3 of **access switch 114** is in the forwarding state, while the remaining designated port(s) connected to **access switch 114** are blocked. EX1005, 6:33-35. By placing only one designated port of **backbone switch 122** that is connected to **access switch 114** in the forwarding state while placing other designated ports that are connected to **access switch 114** in the blocking state, no loop is created in the network between **backbone switch 122** and **access switch 114**. McNair, ¶177.

6. Claim 3: A method according to claim 2, wherein configuring the first and second ports comprises configuring the edge switch so as to decrease a radius of a spanning tree established in the network.

The '986 patent describes reducing a radius of a spanning tree by running the standard spanning tree algorithm over fewer branching communication links. EX1001, 4:44-48. The '986 patent reduces the number of such links by managing redundant downstream ports in a purportedly novel manner. EX1001, 4:35-37.

The '986 patent manages redundant downstream ports by placing one in the forwarding state while blocking the others. EX1001, 3:67-4:4. Likewise, Gai discloses, or at least renders obvious, placing one port of **backbone switch 122** that is connected to the root port of **access switch 114** in the forwarding state while blocking other backbone switch ports connected to the **access switch 114** (*supra* §VII.A.4.c), which—for the same reasons explained in the '986 patent—decreases a radius of a spanning tree in Gai's network, thus meeting claim 3. McNair, ¶178.

7. **Claim 6: A method according to claim 1, wherein sending the dummy traffic comprises sending data frames originating at the edge switch while having source addresses corresponding to network addresses of other entities in the network that are accessible to the access switches via the edge switch.**

As explained above in [1A] (*supra* §VII.A.4.b), **backbone switch 122** is an edge switch. As explained above in [1E] (*supra* §VII.A.4.f), it would have been obvious for **backbone switch 122** (*i.e.*, the edge switch) to send dummy multicast messages and thereby originate them. Such “dummy multicast messages carry the source address of each entity that is directly coupled to the device with the new active port or downstream thereof (relative to the root) and are received by other devices in the computer network.” EX1005, 5:54-60. Since such entities directly coupled to the device (*e.g.*, **backbone switch 122**) or downstream thereof are entities in the network and are accessible to access switches via the device, Gai's **backbone switch 122** sends the dummy traffic as claimed. McNair, ¶179.

8. Claim 7: A method according to claim 6, wherein sending the data frames comprises copying the source addresses for the data frames from the database of the edge switch.

Gai discloses that each switch (*e.g.*, **backbone switch 122**) has a filtering database. EX1005, 1:53-54. Gai discloses that “filtering database 244 [is] configured to store [addresses] corresponding to the entities of network 100.” EX1005, 8:31-38. Gai discloses a switch loading a source address field of a dummy multicast message 400 with “the address of an entity directly coupled to switch 114.” EX1005, 13:1-4; 13:14-16; 13:50-57. Since a switch stores these addresses in its filtering database 244, it would have been obvious—as the simplest and fastest way of obtaining the address—for the switch to load (*i.e.*, copy) an address from its filtering database 244 into the source address field of the dummy multicast message 400, meeting claim 7. McNair, ¶180.

9. Claim 8: A method according to claim 7, wherein

a. [8A] the edge switch has three or more ports, including the first and second ports, and

Gai meets, or at least renders obvious, [8A] because **backbone switch 122**, as shown below in an annotated version of Figure 1 of Gai, has at least three ports (identified by the yellow dots): a first port connected to port 3 of **access switch 114** (the “first” port), a second port connected to port 2 of **access switch 114** (the “second” port), and a third port connected to backbone switch 120. McNair, ¶181.

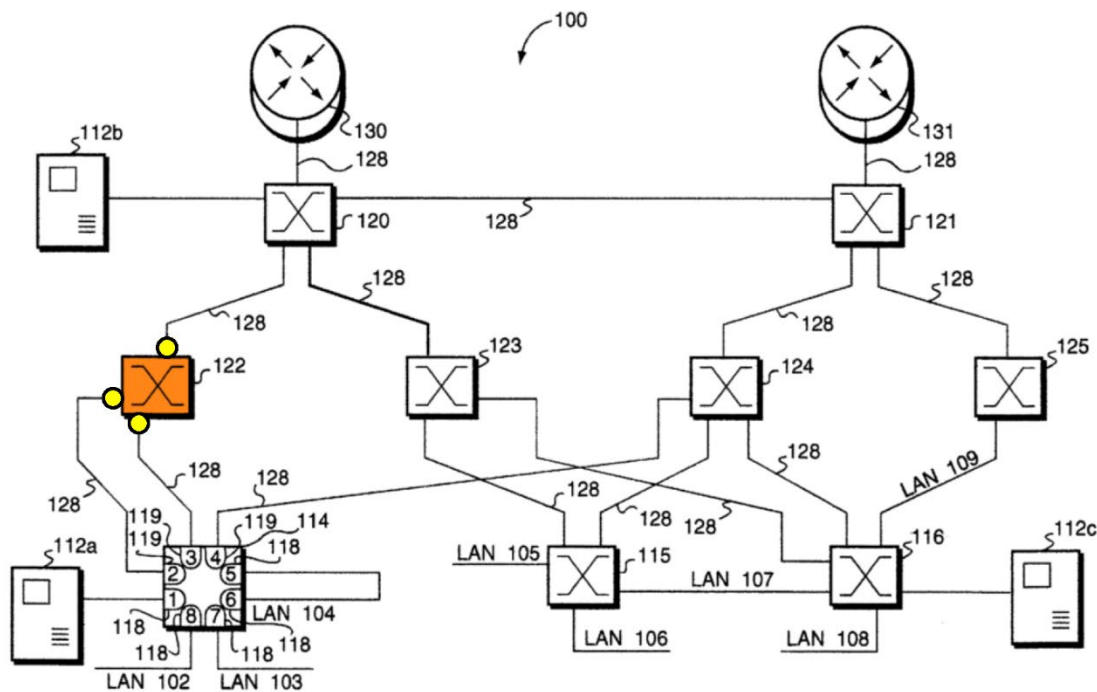


FIG. 1

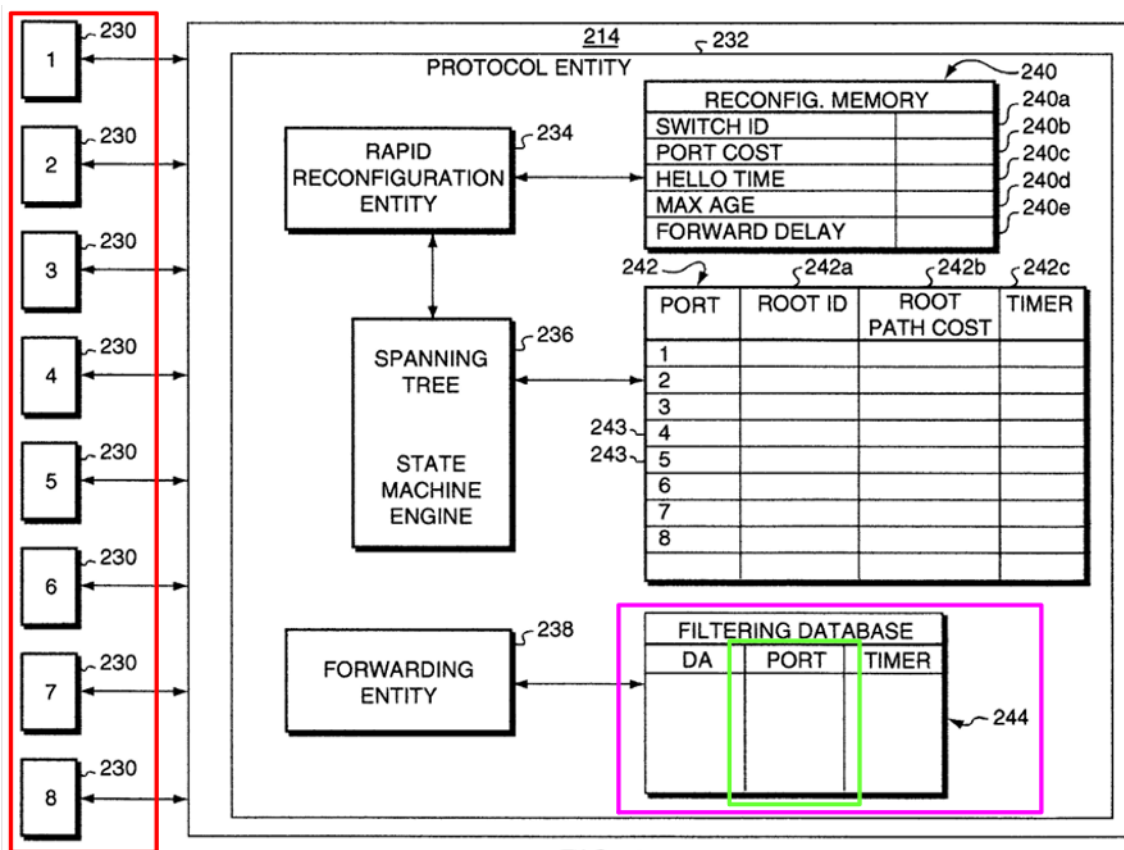
Annotated version of Figure 1 of Gai

- b. **[8B] wherein prior to the failure, the database comprises entries associated with a plurality of the ports, and**

Gai meets [8B] because **backbone switch 122**, like every switch, has a **filtering database 244** that comprises entries associated with a plurality of its ports. EX1005, 1:53-54. As shown below in an annotated version of Gai's Figure 2, **filtering database 244** has a "plurality of columns or cells, including a destination address cell, a destination port cell and a corresponding timer cell." EX1005, 8:32-37; 1:53-54. "Each row or entry in the filtering database 244 preferably corresponds to a particular entity." EX1005, 8:37-39. McNair, ¶182.

Each row or entry in the destination port cell (enclosed in the green box) includes destination port information for a plurality of ports (shown as ports 230 in Figure 2 below and enclosed in the red box) for backbone switch 122, and each row or entry is thereby associated with the plurality of the ports, meeting [8B].

McNair, ¶¶183-184.



Annotated version of Figure 2 of Gai

- c. **[8C] wherein copying the source addresses comprises using the network addresses in all the entries that are not associated with the first port as the source addresses of the data frames.**

As discussed in [8B] (*supra* §VII.A.9.b), Gai discloses using the network addresses in **filtering database 244** as the source addresses of dummy multicast messages. Gai further discloses “generat[ing] and transmit[ing] dummy multicast messages 400 for all entities directly coupled to switch 114 or downstream thereof (relative to the root) through the new forwarding port (i.e., port number four).” EX1005, 14:55-58. Each dummy multicast message has a source address corresponding to one such entity. EX1005, 13:1-4; 13:14-33. Sending the dummy messages from an edge switch would have been obvious. *Supra* §VII.A.4.f. A device that receives the dummy multicast message, which corresponds to one such entity, notes that the message is received from a different source port than the receiving device previously associated with the entity. As such, each dummy multicast message has a network address associated with the new forwarding port and not the previous forwarding port, which meets [8C]. McNair, ¶185.

10. Claim 9: A method according to claim 6, wherein sending the data frames comprises incorporating a Virtual Local Area Network (VLAN) tag in a header of the data frames.

Gai discloses incorporating a virtual local area network (VLAN) tag 416 to a header of dummy multicast message 400 as shown below in an annotated version of Gai’s Figure 4, meeting claim 9. EX1005, 17:32-41. McNair, ¶186.

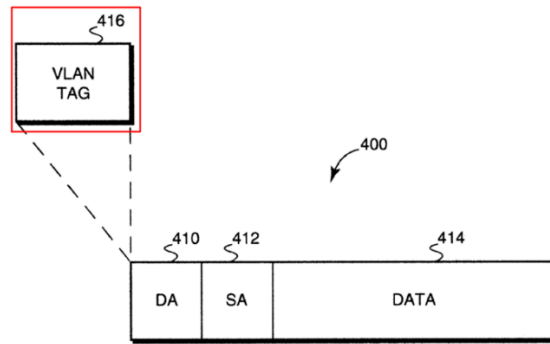


FIG. 4

Annotated version of Figure 4 of Gai

- 11. Claim 10: A method according to claim 6, wherein sending the data frames comprises multicasting the data frames to all of the one or more access switches.**

Gai discloses “load[ing] DA field 410 with a multicast address that causes [dummy] message 400 to be received by all devices (e.g., access switches 114-116...)” in the network, meeting claim 10. EX1005, 13:9-13. McNair, ¶187.

- 12. Claim 11: A method according to claim 1, wherein sending the dummy traffic comprises sending data frames downstream from the edge switch toward the access switches.**

As explained in [1E] (*supra* §VII.A.4.f), it would have been obvious for all of Gai’s switches, including **backbone switch 122** (the claimed “edge switch”) to send dummy messages, including downstream from **backbone switch 122** toward **access switch 114**. Per the ’986 patent, the claimed “dummy traffic” refers to dummy data frames. EX1001, 5:24-25, 12:51-52. Data frames are messages. EX1005, 11:16. Therefore, Gai’s “dummy messages” are “dummy traffic” comprising “data frames,” meeting claim 11. McNair, ¶¶104, 188.

13. Claim 12

- a. **[12A] A method for network communication, comprising: coupling a first bridge in a network to a second bridge located downstream of the first bridge, by connecting at least first and second redundant links to respective first and second ports of the first bridge so as to communicate with the second bridge;**

Gai meets [12A] for the same reasons as [1A]. The first and second bridges of [12A] are met by **backbone switch 122** and **access switch 114**, respectively.

Backbone switch 122 and **access switch 114** are coupled together in the claimed arrangement as shown by Figure 1 of Gai. *Supra* §VII.A.4.b. McNair, ¶189.

- b. **[12B] placing the first port in an active state, while placing the second port in a blocking state;**

Gai meets [12B] for the same reasons as [1B]. *Supra* §VII.A.4.c. McNair, ¶190.

- c. **[12C] conveying communication traffic over the first link while the first port is in the active state, thereby causing the bridges to build respective databases for use in forwarding the traffic;**

Gai meets [12C] for the same reasons as [1C]. *Supra* §VII.A.4.d. McNair, ¶191.

- d. **[12D] responsive to a failure associated with the first link, placing the second port in the active state and the first port in the blocking state; and**

Gai meets [12D] for the same reasons as [1D]. *Supra* §VII.A.4.e. McNair, ¶192.

- e. **[12E] sending dummy traffic from the first bridge over the second link downstream to the second bridge, so as to cause the second bridge to modify its database responsive to the second port being in the active state.**

Gai meets [12E] for the same reasons as [1E]. *Supra* §VII.A.4.f.

McNair, ¶193.

14. Claim 13

- a. **[13A] A network switching device, comprising: a network port, adapted to be coupled over a network trunk link to a core switch in a network so as to exchange communication traffic with the core switch;**

Gai discloses [13A] because **backbone switch 122** meets the claimed network switching device. **Backbone switch 122** comprises (or at least renders obvious, *supra* §VII.A.4.b) a network port (*i.e.*, a trunk port) that is coupled to backbone switch 120 (*i.e.*, a core switch) in the network via a network trunk link to exchange communication traffic with the backbone switch 120. The '986 patent uses "trunk port" synonymously with "root port." EX1001, 3:23-24 ("[T]he trunk ports are on the upstream side of each switch, i.e., the side leading toward the root."). This port on **backbone switch 122** is coupled to backbone switch 120, which is a "core switch" per the '986 patent because it connects access switches to a WAN via router(s). EX1001, 1:32-35; EX1005, Fig. 1. The purpose of the connection is to exchange communication traffic. EX1005, 7:31-34. Accordingly, Gai's **backbone switch 122** meets or renders obvious [13A]. McNair, ¶194.

- b. [13B] first and second access ports, adapted to be coupled via first and second redundant network access links to one or more access switches serving network users; and**

Gai meets [13B] for the same reasons as [1A]. **Backbone switch 122** has (or at least it would have been obvious for it to have) first and second ports coupled to **access switch 114** via first and second redundant network access links 128. *Supra* §VII.A.4.b. McNair, ¶195.

- c. [13C] a protocol processor, adapted to place the first access port in an active state, while placing the second access port in a blocking state, so that the communication traffic is conveyed over the first link while the first port is in the active state, thereby causing each of the access switches to build a respective access switch database for use in forwarding the traffic, the protocol processor being further adapted, responsive to a failure associated with the first link, to place the second access port in the active state and the first port in the blocking state and to send dummy traffic through the second access port over the second link to the one or more access switches, so as to cause each of the one or more access switches to modify its respective access switch database responsive to the second access port being in the active state.**

The capabilities of the “protocol processor” of [13C] match the steps recited [1B]-[1E]. Gai thus renders obvious [13C] for the same reasons set forth in [1B]-[1E], and because Gai discloses protocol entity 232 (a “protocol processor”) for performing these functions, including via rapid reconfiguration entity 234, logic circuits and “programmable processing elements.” EX1005, 8:12, 8:40-51.

These elements parallel the '986 patent's protocol processor 44, about which the '986 patent explains: "[A]lthough protocol processor 44 is shown here as a single, separate entity, its functions are in actuality distributed among a number of separate processors..." EX1001, 8:58-61. McNair, ¶196.

Switch 214, containing protocol entity 232, corresponds to [access switch 114](#), but just as any switch in Gai can include a rapid reconfiguration entity 234 (EX1005, 8:12-15) it would have been obvious to include protocol entity 232 in other switches, such as backbone switches, since rapid reconfiguration entity 234 is part of protocol entity 232. EX1005, 8:67-9:1. In summary, [13C] is obvious for the same reasons as [1B]-[1E]. *Supra* §§VII.A.4.c-VII.A.4.f. McNair, ¶197.

15. Claim 14: A device according to claim 13, wherein the protocol processor is adapted to configure the first and second access ports so that no loop is created in the network between the edge switch and the one or more access switches.

To the extent "edge switch" in claim 14 is intended to refer to the "network switching device" of claim 13,⁴ Gai renders obvious claim 14, via Gai's protocol entity 232, for the same reasons explained above with respect to claim 2. *Supra* §VII.A.5. McNair, ¶198.

⁴ Petitioner reserves the right to challenge claim 14 and its dependents for indefiniteness in litigation.

- 16. Claim 15: A device according to claim 14, wherein the protocol processor is further adapted to configure the first and second access ports so as to decrease a radius of a spanning tree established in the network.**

To the extent “edge switch” in claim 14 refers to claim 13’s “network switching device,” Gai renders obvious claim 15, via Gai’s protocol entity 232, for the same reasons explained above with respect to claim 3. *Supra* §VII.A.6. McNair, ¶199.

- 17. Claim 18: A device according to claim 13, wherein the dummy traffic comprises data frames originating at the edge switch while having source addresses corresponding to network addresses of other entities in the network that are accessible to the access switches via the edge switch.**

To the extent “edge switch” in claim 18 refers to the “network switching device” of claim 13, Gai renders obvious claim 18, via Gai’s protocol entity 232, for the same reasons explained above with respect to claim 6. *Supra* §VII.A.7. McNair, ¶200.

- 18. Claim 19: A device according to claim 18, and**
a. [19A] comprising a memory,

Gai’s protocol entity 232 meets [19A] because it comprises “memory 240 and tables 242 and 244 [that] may be implemented as content addressable memory (CAM) devices.” EX1005, 8:40-42, 1:53-54. McNair, ¶201.

- b. **[19B] wherein the protocol processor is adapted to build an edge switch database in the memory responsive to the communication traffic prior to the failure associated with the first link, and to copy the source addresses for the data frames from the edge switch database.**

To the extent “edge switch” in [19B] refers to the “network switching device” of claim 13, Gai’s protocol entity 232 meets [19B] for the same reasons explained above with respect to [1C], [13C], and claim 7. *Supra* §§VII.A.4.d, VII.A.14.c, VII.A.8. McNair, ¶202.

- 19. **Claim 20: A device according to claim 19, wherein prior to the failure, the edge switch database comprises entries associated with a plurality of the ports, and wherein the protocol processor is adapted to use the network addresses in all the entries that are not associated with the first access port as the source addresses of the data frames.**

To the extent “edge switch” in claim 20 refers to the “core switch” of claim 13, Gai renders obvious claim 20 for the reasons explained above with respect to claim 8. *Supra* §VII.A.9. McNair, ¶203.

- 20. **Claim 21: A device according to claim 18, wherein the protocol processor is adapted to incorporate a Virtual Local Area Network (VLAN) tag in a header of the data frames.**

Gai renders obvious claim 21 for the same reasons explained above with respect to claim 9. *Supra* §VII.A.10. McNair, ¶204.

- 21. Claim 22: A device according to claim 18, wherein the protocol processor is adapted to multicast the data frames to all of the one or more access switches.**

Gai renders obvious claim 22, via its protocol processor, for the same reasons explained above with respect to claim 10. *Supra* §VII.A.11.

McNair, ¶205.

- 22. Claim 23: A device according to claim 13, wherein the dummy traffic comprises data frames sent downstream from the edge switch toward the access switches.**

Gai renders obvious claim 23 for the same reasons explained above with respect to claim 11. *Supra* §VII.A.12. McNair, ¶206.

23. Claim 24

- a. [24A] A bridge device for use in a communication network having a spanning tree root, the device comprising: a designated port, adapted to be coupled over an upstream link in the network so as to communicate with the root;**

Gai discloses **backbone switch 122** (*i.e.*, the claimed bridge device) in a network topology that meets [24A]. The network topology includes backbone switch 120 as the spanning tree root. EX1005, 8:57-59. Further, as provided in [1A] (*supra* §VII.A.4.b), **backbone switch 122** has (or such would be obvious) a designated port coupled to **access switch 114** over link 128 (an upstream link), allowing **access switch 114** to communicate (via **backbone switch 122**) with the root. EX1005, Fig. 1. McNair, ¶207.

- b. **[24B] first and second downstream ports, adapted to be coupled via first and second redundant links to a further bridge device located downstream in the network relative to the spanning tree root; and**

As explained in [1A] (*supra* §VII.A.4.b), **backbone switch 122** has (or such would be obvious) first and second downstream ports coupled to **access switch 114** (*i.e.*, a further bridge device) over redundant links 128 to meet [24B].

McNair, ¶208.

- c. **[24C] a protocol processor, which is adapted to place the first downstream port in an active state, while placing the second downstream port in a blocking state, so that communication traffic is conveyed over the first downstream link while the first port is in the active state, thereby causing the bridge devices to build respective databases for use in forwarding the traffic, the protocol processor being further adapted, responsive to a failure associated with the first link, to place the second downstream port in the active state and to send dummy traffic over the second link downstream to the further bridge device, so as to cause the further bridge device to modify its database responsive to the second downstream port being in the active state.**

Modified Gai renders obvious [24C], via Gai's protocol entity 232, for the same reasons explained above with respect to [13C]. *Supra* §VII.A.14.c.

McNair, ¶¶209-210.

**B. Ground 2: Claims 4, 5, 16, and 17:
Obvious over Gai in View of IEEE 1999, Further in View of Jain**

1. U.S. Patent No. 6,330,229 (“Jain,” EX1006)

Jain discloses managing forwarding databases (*i.e.*, filtering databases) in case of link failure. EX1006, Abstract, 6:52-54. Like Gai, Jain discloses detecting a failure and transitioning a backup port to take over for the failed port. EX1006, 5:15-18. To reduce recovery time, the forwarding databases are updated by transferring information from the failed port to the backup port, which is now active. The updates are made before the port transition to maintain high network availability. EX1006, 4:28-33; 5:15-20. McNair, ¶211.

2. Gai in View of IEEE 1999, Further in View of Jain

Gai and Jain are both directed to addressing link failures in spanning tree networks while reducing link failure recovery time. As discussed previously, Gai discloses an access switch with a protocol entity, which includes a filtering database in which each row or entry stores address information of a particular network entity. EX1005, 8:31-38, Fig. 2. As explained *supra* §VII.A.14.c, it would have been obvious to use the protocol entity in any backbone switch and, as explained *supra* §VII.A.4.b, Gai’s backbone switches are “edge switches” as claimed. Therefore, it would have been obvious to use Gai’s protocol entity in “edge switches.” As discussed in Ground 1, sending dummy traffic downstream would have been obvious in view of IEEE 1999. *Supra* §VII.A.3. McNair, ¶212.

Gai discloses that, in response to a root port failure, the protocol entity transitions a back-up port to be the new root port and starts processing messages through the newly-activated port. EX1005, 12:15-18, 12:32-42. McNair, ¶213.

Gai expresses concern that messages may be misdirected or lost during the transition process. EX1005, 12:59-62, 15:18-21. POSAs would have been motivated to modify Gai by incorporating Jain's technique of updating a filtering database responsive to a link failure for the benefit of reducing the possibility of misdirected/lost data messages during the self-healing operation. As modified, Gai's protocol entity would update the filtering database of a switch (including backbone switch 122) after detecting a failure at a port (*e.g.*, a downstream port). The update would be implemented by remapping entries from the failed port to a backup port (*e.g.*, a back-up downstream port). The update would be implemented before receiving communication traffic on the backup port to reduce the possibility of the backup port misdirecting or losing received messages using outdated database information. The modification would result in a backbone switch's protocol entity transitioning a failed port in the forwarding state to the blocking state and a backup port from the blocking state to the forwarding state, all with a reduced possibility of data messages being misdirected or lost because the backbone switch's database would be updated prior to notifying other network switches of the newly-activated backup port. McNair, ¶214.

3. **Claim 4: A method according to claim 1, and comprising, responsive to placing the second port in the active state, updating the database of the edge switch so that entries in the database point to the second port instead of the first port.**

Gai transitions a backup port (*i.e.*, the second port) in the blocking state at a backbone switch (*e.g.*, **backbone switch 122**) to an active port in the forwarding state (*i.e.*, the active state) after detecting a failure at a port (*i.e.*, the first port). *Supra* §VII.A.4.e. Responsive to placing the backup port in the active state, Gai, modified in view of Jain, would update the filtering database of **backbone switch 122** by transferring the database entries that pointed to the failed port to point instead to the backup port, thus rendering claim 4 obvious. EX1006, 5:15-20, 14:13-26. McNair, ¶215.

4. **Claim 5: A method according to claim 4, wherein updating the database comprises changing the entries in the database to point to the second port before receiving the communication traffic from the one or more access switches on the second port.**

Modified in view of Jain, Gai would map, in **backbone switch 122**, the failed port forwarding data to the backup port after the backup port transitions from the blocking state to the forwarding state. EX1006, 14:13-15. “After mapping the forwarding data from the old root port to the new root port in block 703, ...a packet is sent on designated ports with the new root path cost data” to allow “descendant bridges to recompute their spanning tree configuration.” EX1006,

14:27-31. The descendant bridges (*e.g.*, one or more access switches) would not send communication traffic to the backup port until notified to do so. Thus Gai, modified in view of Jain, meets claim 5 because it updates the filtering database in **backbone switch 122** to point toward the backup port prior to the backup port receiving communication traffic from another network entity. McNair, ¶216.

5. Claim 16: A device according to claim 13, and

a. [16A] comprising a memory,

Gai's protocol entity 232 meets [16A] because it comprises "memory 240 and tables 242 and 244 [that] may be implemented as content addressable memory (CAM) devices." EX1005, 8:40-42, 1:53-54. McNair, ¶217.

b. [16B] wherein the protocol processor is adapted to build an edge switch database in the memory responsive to the communication traffic prior to the failure associated with the first link

To the extent "edge switch" in [16B] refers to the "network switching device" of claim 13, Gai's protocol entity 232 meets [16B] for the reasons explained above with respect to [1C] and [13C]. *Supra* §§VII.A.4.d, VII.A.14.c. McNair, ¶218.

- c. **[16C] and, responsive to placing the second access port in the active state, to update the edge switch database so that entries in the edge switch database point to the second access port instead of the first access port.**

To the extent “edge switch” in [16C] refers to the “network switching device” of claim 13, Gai’s protocol entity 232 meets [16C] for the reasons explained above with respect to claim 4. *Supra* §VII.B.3. McNair, ¶219.

6. **Claim 17: A device according to claim 16, wherein the protocol processor is adapted to update the entries in the database to point to the second access port before receiving the communication traffic from the one or more access switches on the second port.**

Gai’s protocol entity 232 meets claim 17 for the reasons explained above with respect to claim 5. *Supra* §VII.B.4. McNair, ¶220.

VIII. DISCRETIONARY DENIAL IS NOT APPROPRIATE

A. Fintiv

Under the Director’s Interim Procedure for Discretionary Denials,⁵ “the PTAB will not deny institution based on *Fintiv*⁶ if there is compelling evidence of unpatentability”—which this Petition provides. That should conclude the *Fintiv* analysis. *Nokia of Am. Corp. v. TQ Delta, LLC*, IPR2022-00471, Paper 11, 19-21 (Aug. 18, 2022).

⁵ <https://tinyurl.com/2s4rmz9x> (“Interim Procedure”).

⁶ *Apple Inc. v. Fintiv, Inc.*, IPR2020-00019, Paper 11 (Mar. 20, 2020) (precedential).

In any event, all *Fintiv* factors weigh against discretionary denial, or at least are neutral.

1. Potential Stay

Petitioner will seek a stay, but without “specific evidence” how the court will rule, this factor is neutral. *Sand Revolution II, LLC v. Cont’l Intermodal Grp.–Trucking LLC*, IPR2019-01393, Paper 24, 7 (June 16, 2020) (informative).

2. Timing of Trial

As of this Petition’s filing, the court has neither entered a schedule nor set a trial date. In any event, the Board considers medial time-to-trial (Interim Procedure, 8-9)—34.6 months in the District of Massachusetts, where the Juniper Litigation is pending. EX1012. This puts a trial involving Petitioner in approximately July 2026, ***well after*** a final written decision (“FWD”) herein.

While the Eastern District of Texas median time-to-trial statistics suggest the Cisco Litigation could reach trial before a FWD herein, a pre-FWD trial involving an unrelated defendant does not favor discretionary denial. *Dolby Labs. v. Intertrust Techs.*, IPR2020-00665, Paper 11, 15-17 (Feb. 16, 2021) (*Fintiv* Factor 2 weighed against denial, despite pre-FWD trial in Texas, because petitioner “Dolby is not a party to the Texas Actions”).

Further, Petitioner filed this petition diligently, approximately five months before the Section 315(b) deadline. EX1013. *Fintiv*, Paper 11, 11 (filing IPR petition “expeditiously...weigh[] against [denial]”); *Apple v. Seven Networks*, IPR2020-00235, Paper 10, 9-10 n.6 (July 28, 2020) (filing IPR petition four months before Section 315(b) deadline is “reasonably diligent”).

3. Litigation Investment

In the Juniper Litigation, no contentions have been exchanged and no discovery or claim construction proceedings have occurred. The lack of meaningful investment in the litigation disfavors denial. *Samsung Elec. Co., Ltd. v. Netlist, Inc.*, IPR2022-00063, Paper 13, 8 (May 5, 2022) (instituting IPR where no “claim construction and merits determinations” had occurred in litigation).

4. Overlap of Issues

Patent Owner’s amended complaint asserts only claim 12. EX1014. This IPR addresses all claims of the ’986 patent, including many that are not before the district court, weighing against denial. *Google LLC v. Geoscope Techs. Pte. Ltd.*, IPR2023-01212, Paper 14, 15 (Feb. 20, 2024).

5. Litigation Defendants

Petitioner is the litigation defendant, which is “far from an unusual circumstance.” *Sand Revolution II*, Paper 24, 12-13. This factor is neutral.

B. Section 325(d) and *Advanced Bionics*

Neither IEEE 1999 nor Jain were cited during prosecution of the '986 patent. Thus, *Advanced Bionics* Step 1 is not met as to either of these references. *Advanced Bionics, LLC v. MED-EL Elektromedizinische Geräte GmbH*, IPR2019-01469, Paper 6, 9-10 (Feb. 13, 2020) (precedential).

Although Gai is mentioned in the '986 patent specification, Gai was never applied during prosecution. Accordingly, *Advanced Bionics* Step 1 is not met with respect to Gai based on the '986 patent's prosecution. *Group III Int'l, Inc. v. Targus Group Int'l*, IPR2021-00371, Paper 21 (July 9, 2021) (Step 1 not met where petition relied on prior art in IDS not applied by Examiner to reject claims); *Align Technology, Inc. v. Dental Monitoring SAS*, IPR2024-00110, Paper 7, 10-11 (Apr. 19, 2024) (where prior art was cited in specification but not applied during prosecution, "the first part of the *Advanced Bionics* framework is not satisfied").

While the earlier-filed Cisco IPR relied on Gai, it has not been decided, so *Advanced Bionics* Step 1 is not met with respect to the Cisco IPR either. *AliveCor, Inc. v. Apple Inc.*, IPR2023-00950, Paper 12, 51 (Jan. 9, 2024) (instituting trial: "Since the 745 IPR has only been instituted and the Board has not yet reached a final decision, the prior art reference and arguments could not yet have been considered 'previously...presented to the Office.'").

Further, this Petition uses Gai differently. The Cisco IPR argues obviousness over Gai alone, while this Petition argues obviousness over Gai in view of IEEE 1999 (Ground 1), further in view of Jain (Ground 2). *AliveCor*, 52 (*Advanced Bionics* Step 1 not met where petition used same primary reference as earlier IPR, but employed different secondary reference).

Further, even if the Board were to conclude that Gai was “previously presented to the Office” per *Advanced Bionics* Step 1, discretionary denial would remain inappropriate because *Advanced Bionics* Step 2 plainly is not satisfied. *Advanced Bionics* Step 2 focuses on Becton Dickinson factors (c), (e), and (f) (*see Advanced Bionics*, 10)—which all weigh against discretionary denial.

With respect to the ’986 patent’s prosecution, for factor (c), while the specification mentions Gai, the Examiner never applied it. This weighs “strongly against” discretionary denial. *Code 2000 v. Bright Data Ltd.*, IPR2022-00353, Paper 8, 10 (July 1, 2022). Similarly, the Cisco IPR has not been decided. For factors (e) and (f), “if...the Office’s previous consideration of the art is not well developed or silent, then a petitioner may show the Office erred by overlooking something persuasive under factors (e) and (f).” *Advanced Bionics*, 10. Since Gai was not applied during prosecution, let alone in view of IEEE 1999 or Jain, and there has been no decision in the Cisco IPR, there is no prior analysis to review for error, other than allowance of claims that are unpatentable as explained herein.

Bowtech, Inc. v. MCP IP, LLC, IPR2019-00382, Paper 12, 13 (Aug. 6, 2019). The Office also lacked the benefit of expert testimony. This new evidence “undermines the case for discretionary denial.” *Pfizer Inc. v. ModernaTX, Inc.*, IPR2023-01358, Paper 19, 16 (Mar. 19, 2024)

Finally, this Petition challenges all claims of the ’986 patent, while the Cisco petition challenges only a subset of the claims. The fact that this Petition gives the Board an opportunity to evaluate the patentability of every ’986 patent claim weighs against discretionary denial.

C. Section 314(a) and *General Plastic*

The Director has explained that, under *General Plastic*,⁷ if a later and earlier petitioner “do not have a significant relationship, exercising discretion to deny the Petition is not justified.” *Videndum Prod. Sol’ns, Inc. v. Rotolight Ltd.*, IPR2023-01218, Paper 12, 7 (Apr. 19, 2024). The Board need go no further because the Petitioner, Juniper Networks, Inc., is independent of Cisco and the accused products in the two suits are different: Cisco routers in the Cisco Litigation and Juniper routers in the Juniper Litigation. EX1014; EX1015.

For completeness, Petitioner will briefly address the *General Plastic* factors.

⁷ *General Plastic Indus. Co. Ltd. v. Canon Kabushiki Kaisha*, IPR2016-01357, Paper 19 (Sept. 6, 2017) (precedential).

Factor 1 “weights strongly against discretionary denial” because Petitioner has not previously filed an IPR against the ’986 patent. *Mercedes-Benz USA, LLC v. Spero*, IPR2023-01040, Paper 10, 9 (Dec. 12, 2023).

Factors 2-5 are “substantially diminished” because “Petitioner did not previously challenge the challenged claims.” *Mercedes-Benz*, 9.

Factor 3 disfavors discretionary denial because, as of this Petition’s filing, neither the Patent Owner Preliminary Response nor the Board’s institution decision have been received in the Cisco IPR. *AliveCor*, 48.

Under Factors 2, 4 and 5, the fact that Juniper knew of Gai when Cisco filed its petition, and filed the present Petition about three months later, does not favor discretionary denial because the present Petition applies Gai differently than the Cisco petition and in view of new secondary references not used in the Cisco petition. *Arista Networks, Inc. v. Corrigent Corp.*, IPR2023-00839, Paper 9, 14 (Dec. 7, 2023) (granting institution and noting: “we do not find the three-month delay between the petition in the Cisco IPR and the instant petition to be excessive because Petitioner has asserted additional art and acquired supplementary declaratory testimony from a different declarant...the fourth and fifth factors are neutral.”).

Factor 6 (the Board's finite resources) does not justify discretionary denial. The fact that only this Petition gives the Board an opportunity to evaluate the patentability of every claim in the '986 patent weighs against discretionary denial. Further, while there are important differences between this Petition and Cisco's IPR petition, the burden on the Board is modest because the primary reference, Gai, is the same.

Factor 7 does not disfavor institution because there is no obstacle to the Board deciding this IPR within one year of institution.

IX. CONCLUSION

Petitioner respectfully requests cancellation of all the challenged claims.

Dated: May 3, 2024

Respectfully submitted,

Juniper Networks, Inc.

By its counsel,

/Michael N. Rader/

Michael N. Rader, Reg. No. 52,146
WOLF, GREENFIELD & SACKS, P.C.

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. §42.24, the undersigned certifies that the foregoing Petition for *Inter Partes* Review contains 14,000 words, including the signature block and excluding a table of contents, a table of authorities, Mandatory Notices under §42.8, a certificate of service or word count, or appendix of exhibits or claim listing. Petitioner has relied on the word count feature of the word processing system used to create this paper in making this certification.

Date: May 3, 2024

/ Virginia L. Weeks /
Virginia L. Weeks
Paralegal
Wolf, Greenfield & Sacks, P.C.

CERTIFICATE OF SERVICE

I certify that, on May 3, 2024, I will cause a copy of the foregoing document, including any exhibits or appendices filed therewith, to be served via United States Priority Mail Express International at the following correspondence address of record for the patent:

May Patents Ltd. c/o Dorit Shem-Tov
P.O.B 7230
Ramat-Gan, -5217102
ISREAL

I further certify that I will cause courtesy copies of the foregoing documents, including any exhibits and appendices filed herewith, to be served via email and United States Priority Mail Express to Patent Owner's litigation counsel at the following email and mail addresses:

Eric G. J. Kaviar, Esquire
Sunstein LLP
100 High Street
Boston, MA 02110
Email: ekaviar@sunsteinlaw.com

Date: May 3, 2024

/ Virginia L. Weeks /
Virginia L. Weeks
Paralegal
Wolf, Greenfield & Sacks, P.C.

X. CLAIM LISTING

The following claim listing assigns labels (e.g., [1PRE], [1A], [1B], etc.) to limitations of certain claims for clarity.

Claim 1
[1PRE] A method for network communication, comprising:
[1A] coupling an edge switch in a network to one or more access switches via at least first and second redundant links, respectively connected to first and second ports of the edge switch;
[1B] placing the first port in an active state, while placing the second port in a blocking state;
[1C] conveying communication traffic over the first link while the first port is in the active state, thereby causing each of the switches to build a respective database for use in forwarding the traffic;
[1D] responsive to a failure associated with the first link, placing the second port in the active state and the first port in the blocking state; and
[1E] sending dummy traffic from the edge switch over the second link to the one or more access switches, so as to cause each of the one or more access switches to modify its respective database responsive to the second port being in the active state.
Claim 2
A method according to claim 1, wherein placing the second port in the blocking state comprises configuring the first and second ports so that no loop is created in the network between the edge switch and the one or more access switches.
Claim 3
A method according to claim 2, wherein configuring the first and second ports comprises configuring the edge switch so as to decrease a radius of a spanning tree established in the network.
Claim 4
A method according to claim 1, and comprising, responsive to placing the second port in the active state, updating the database of the edge switch so that entries in the database point to the second port instead of the first port.

Claim 5
A method according to claim 4, wherein updating the database comprises changing the entries in the database to point to the second port before receiving the communication traffic from the one or more access switches on the second port.
Claim 6
A method according to claim 1, wherein sending the dummy traffic comprises sending data frames originating at the edge switch while having source addresses corresponding to network addresses of other entities in the network that are accessible to the access switches via the edge switch.
Claim 7
A method according to claim 6, wherein sending the data frames comprises copying the source addresses for the data frames from the database of the edge switch.
Claim 8
A method according to claim 7, wherein
[8A] the edge switch has three or more ports, including the first and second ports, and
[8B] wherein prior to the failure, the database comprises entries associated with a plurality of the ports, and
[8C] wherein copying the source addresses comprises using the network addresses in all the entries that are not associated with the first port as the source addresses of the data frames.
Claim 9
A method according to claim 6, wherein sending the data frames comprises incorporating a Virtual Local Area Network (VLAN) tag in a header of the data frames.
Claim 10
A method according to claim 6, wherein sending the data frames comprises multicasting the data frames to all of the one or more access switches.
Claim 11
A method according to claim 1, wherein sending the dummy traffic comprises sending data frames downstream from the edge switch toward the access switches.

Claim 12

[12A] A method for network communication, comprising: coupling a first bridge in a network to a second bridge located downstream of the first bridge, by connecting at least first and second redundant links to respective first and second ports of the first bridge so as to communicate with the second bridge;

[12B] placing the first port in an active state, while placing the second port in a blocking state;

[12C] conveying communication traffic over the first link while the first port is in the active state, thereby causing the bridges to build respective databases for use in forwarding the traffic;

[12D] responsive to a failure associated with the first link, placing the second port in the active state and the first port in the blocking state; and

[12E] sending dummy traffic from the first bridge over the second link downstream to the second bridge, so as to cause the second bridge to modify its database responsive to the second port being in the active state.

Claim 13

[13A] A network switching device, comprising: a network port, adapted to be coupled over a network trunk link to a core switch in a network so as to exchange communication traffic with the core switch;

[13B] first and second access ports, adapted to be coupled via first and second redundant network access links to one or more access switches serving network users; and

[13C] a protocol processor, adapted to place the first access port in an active state, while placing the second access port in a blocking state, so that the communication traffic is conveyed over the first link while the first port is in the active state, thereby causing each of the access switches to build a respective access switch database for use in forwarding the traffic, the protocol processor being further adapted, responsive to a failure associated with the first link, to place the second access port in the active state and the first port in the blocking state and to send dummy traffic through the second access port over the second link to the one or more access switches, so as to cause each of the one or more access switches to modify its respective access switch database responsive to the second access port being in the active state.

Claim 14
A device according to claim 13, wherein the protocol processor is adapted to configure the first and second access ports so that no loop is created in the network between the edge switch and the one or more access switches.
Claim 15
A device according to claim 14, wherein the protocol processor is further adapted to configure the first and second access ports so as to decrease a radius of a spanning tree established in the network.
Claim 16
A device according to claim 13, and comprising a memory, wherein the protocol processor is adapted to build an edge switch database in the memory responsive to the communication traffic prior to the failure associated with the first link and, responsive to placing the second access port in the active state, to update the edge switch database so that entries in the edge switch database point to the second access port instead of the first access port.
Claim 17
A device according to claim 16, wherein the protocol processor is adapted to update the entries in the database to point to the second access port before receiving the communication traffic from the one or more access switches on the second port.
Claim 18
A device according to claim 13, wherein the dummy traffic comprises data frames originating at the edge switch while having source addresses corresponding to network addresses of other entities in the network that are accessible to the access switches via the edge switch.
Claim 19
A device according to claim 18, and comprising a memory, wherein the protocol processor is adapted to build an edge switch database in the memory responsive to the communication traffic prior to the failure associated with the first link, and to copy the source addresses for the data frames from the edge switch database.
Claim 20
A device according to claim 19, wherein prior to the failure, the edge switch database comprises entries associated with a plurality of the ports, and wherein the protocol processor is adapted to use the network addresses in all the entries that are not associated with the first access port as the source addresses of the data frames.

Claim 21
A device according to claim 18, wherein the protocol processor is adapted to incorporate a Virtual Local Area Network (VLAN) tag in a header of the data frames.
Claim 22
A device according to claim 18, wherein the protocol processor is adapted to multicast the data frames to all of the one or more access switches.
Claim 23
A device according to claim 13, wherein the dummy traffic comprises data frames sent downstream from the edge switch toward the access switches.
Claim 24
[24A] A bridge device for use in a communication network having a spanning tree root, the device comprising: a designated port, adapted to be coupled over an upstream link in the network so as to communicate with the root;
[24B] first and second downstream ports, adapted to be coupled via first and second redundant links to a further bridge device located downstream in the network relative to the spanning tree root; and
[24C] a protocol processor, which is adapted to place the first downstream port in an active state, while placing the second downstream port in a blocking state, so that communication traffic is conveyed over the first downstream link while the first port is in the active state, thereby causing the bridge devices to build respective databases for use in forwarding the traffic, the protocol processor being further adapted, responsive to a failure associated with the first link, to place the second downstream port in the active state and to send dummy traffic over the second link downstream to the further bridge device, so as to cause the further bridge device to modify its database responsive to the second downstream port being in the active state.