

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CISCO SYSTEMS, INC.

Petitioner

v.

PORTSMOUTH NETWORK CORP.

Patent Owner

Case IPR2024-00505

U.S. Patent No. 6,917,986

DECLARATION OF HENRY H. HOUH, PH.D.

ON BEHALF OF PETITIONER

TABLE OF CONTENTS

	<u>Page</u>
I. Background.....	1
II. Relevant Law	8
A. Claim Construction	8
B. Obviousness.....	9
C. Ordinary Skill in the Art.....	11
III. Summary of Opinions.....	12
IV. Overview of the '968 Patent.....	12
A. Technology Background	12
1. Spanning Tree Protocol (STP).....	12
2. Root Switch.....	16
3. Access, Core, And Edge Switches.....	18
4. Computer Network Having Redundant Links	19
B. Purported Invention Of The '968 Patent.....	23
C. Prosecution History	25
V. Overview of the Prior Art.....	25
A. Gai ("US 6,032,194").....	25
1. Gai's Computer Network Implements A Spanning Tree Protocol	26
2. One Exemplary Failure In Gai's Access Switch 114	30

3.	Additional Embodiments of Gai Not Discussed In The '986 Patent	34
VI.	Level of Ordinary Skill in the Art	42
VII.	Claim Construction.....	43
VIII.	Specific Grounds for Challenge	43
A.	Ground IA: Gai (Downstream Embodiment) Renders Obvious Claims 1-2, 4-7, 9-14, 16-19, and 21-24	43
1.	Independent Claims 1 and 12.....	43
2.	Independent Claim 13	69
3.	Independent Claim 24	77
4.	Dependent Claims 2 and 14	80
5.	Dependent Claim 4	80
6.	Dependent Claims 5 and 17	84
7.	Dependent Claims 6 and 18	86
8.	Dependent Claim 7	88
9.	Dependent Claims 9 and 21	89
10.	Dependent Claims 10 and 22	91
11.	Dependent Claims 11 and 23	93
12.	Dependent Claims 16 and 19	93
B.	Ground IB: Gai (Upstream Embodiment) Renders Obvious Claims 1-2, 4-7, 9-14, 16-19, and 21-24	97
1.	Independent Claim 1	99

2.	Independent Claim 13	120
3.	Independent Claim 24	124
4.	Dependent Claims 2 and 14	126
5.	Dependent Claim 4	126
6.	Dependent Claims 5 and 17	129
7.	Dependent Claims 6 and 18	130
8.	Dependent Claim 7	130
9.	Dependent Claims 9 and 21	131
10.	Dependent Claims 10 and 22	131
11.	Dependent Claims 11 and 23	131
12.	Dependent Claims 16 and 19	132
IX.	Conclusion	133
X.	Availability for Cross-Examination	133
XI.	Right to Supplement	133
XII.	Jurat.....	134

I, Henry H. Houh, declare as follows:

1. My name is Henry H. Houh.
2. I have been retained as an expert witness on behalf of Petitioner Cisco Systems, Inc. to provide expert opinions on the validity of U.S. Patent No. 6,917,986 (the “’986 patent”).

I. Background

3. I received a Ph.D. in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology (“MIT”) in 1998. I also received a Master of Science degree in Electrical Engineering and Computer Science in 1991, a Bachelor of Science degree in Electrical Engineering and Computer Science in 1989, and a Bachelor of Science degree in Physics in 1990, all from MIT.

4. I am currently self-employed as an independent technical consultant. I am also president of a company, Einstein’s Workshop, that provides supplemental science, technology, engineering, and mathematics (“STEM”) education to children of all ages.

5. I first entered telecommunications in 1987 when I worked as a summer intern at AT&T Bell Laboratories as part of a five-year dual degree program at MIT. I continued to work at AT&T Bell Laboratories as part of this MIT program. While at MIT, I was a teaching assistant (“TA”) in the Electrical Engineering and Computer Science Department’s core Computer Architectures

course. I first was a TA as a senior for a role typically reserved for graduate students. I later became head TA. The course covered various topics in computer architectures, as well as programming. As a TA, I helped write homework assignments, lab assignments (including those that involved programming), and exams. I also taught in the recitation sections.

6. Later, as part of my doctoral research at MIT from 1991-1998, I was a research assistant in the Telemedia Network Systems (“TNS”) group at the Laboratory for Computer Science. The TNS group built a high-speed gigabit network and created applications that ran over the network. Example applications included ones for remote video capture, processing, and display of video on computer terminals. In addition to working on the design of core network components, designing and building the high-speed links, and designing and writing the device drivers for the interface cards, I also set up the group’s web server. Also, I helped to maintain, install, and upgrade the networking devices used within the group, along with other graduate students. Being a networking group, we regularly discussed topic of concern in networking, including network security and the kinds of attacks that were carried out over a network. I used several network scanning applications, including Nmap, to assess the level of security in our own network.

7. I also helped to build the web pages that initiated the above-mentioned video sessions via a web interface. Vice President Al Gore visited our group in 1996 and received a demonstration of—and remotely drove—a radio-controlled toy car with a wireless video camera mounted on it that was built by our group. This toy car device received commands transmitted over a network from a remote computer, and video data from the toy car was transmitted wirelessly then over a computer network back to the user controller. On occasion, we allowed users visiting our web site to drive the toy car from their remote computer while they watched the video on their computer. The video stream was encoded by TNS-designed hardware, streamed over the TNS-designed network, and displayed using TNS-designed software.

8. I defended and submitted my Ph.D. thesis, titled “Designing Networks for Tomorrow’s Traffic,” in January 1998. As part of my thesis research, I analyzed local area and wide area flows to show a more efficient method for routing packets in a network, based on traffic patterns at the time. The traffic flow data included Ethernet, IP, TCP or UDP, and RTP header information, which I analyzed to come to the conclusions in my thesis.

9. From 1997 to 1999, I was a Senior Scientist and Engineer at NBX Corporation, a start-up that made business telephone systems for streaming

packetized audio over data networks instead of using traditional telephone lines.

NBX was later acquired by 3Com Corporation.

10. As part of my work at NBX, I designed the core audio reconstruction software algorithms for the telephones, as well as the packet transmission algorithms. I also designed and validated the core packet transport protocol used by the phone system. The protocol was used for all signaling in the phone system, including for the setup of conference calls. The NBX system also featured a computer interface for initiating phone calls, which could also initiate conference calls. The NBX system also supported the Telephony Application Programming Interface (“TAPI”) that allowed other computer programs to integrate with our system telephony features. I also programmed the first prototype of our phone which communicated using IP, and I demonstrated our IP phones working over the Internet when we attended a trade show in California. The phone connected over the Internet to our headquarters in Andover, MA. I also programmed a data capture and analysis system that provided data on the operation of the system.

11. From 1999-2004, I was employed by Empirix or its predecessor company, Teradyne. Empirix was a leader in test tools for telecommunications protocols and systems, providing functional testing tools as well as load testing tools. From 2000-2001, I conceived and built a test platform for testing Voice-over-IP (VoIP). The first application on this new test platform was a cloud

emulator for simulating the effects of transmitting VoIP over a busy network. The test platform was based on a network processor chip, which could be programmed to cut-through packets while processing packet data such as various protocol layer headers including addresses included therein and even packet data contents. I also designed a protocol analyzer built on the same platform. The application captured and performed protocol decoding at various layers in the protocol stacks of captured packets, including detailed Ethernet header decoding, IP header decoding, TCP header decoding, UDP header decoding, RTP header decoding, and many other specified protocols. The application was also designed to reconstruct entire conversations that spanned multiple packets.

12. At Empirix, I also rearchitected the design of the Web Test division's core product, e-Test Suite. E-Test Suite was a software program designed to perform functional and load testing of web applications. I was technical lead on the project, and also worked on programming a Javascript interpreter for the product. Javascript is the flexible programming layer that allows users to interact with web pages through actions such as clicking and dragging that allows web page programmers to extend the functionality of a web page beyond clicking on links. I also designed and prototyped a web server security testing product, which scanned web servers for specific types of vulnerabilities, including looking for administrative logins which could be attacked with password attacks and buffer

overflow attacks, and attacks such as SQL query injection attacks. We used the prototype on some customer web sites, with their permission, and successfully detected vulnerabilities.

13. In 2006, as part of my role at BBN Technologies, I helped found PodZinger Inc., now known as RAMP Inc. PodZinger utilized BBN's speech recognition algorithms to search through the spoken words in audio and video segments. While I was Vice President of Operations and Technology, PodZinger followed its initial prototype with a full streaming audio and video search solution. I also created a social networking web site, which BBN sold to a venture-funded startup company. In the process of creating the web site, I designed and specified the authentication and authorization protocols.

14. Until October 2022, I was President of Einstein's Workshop, a STEM education company for kids. At Einstein's Workshop, we taught programming to children of all ages, starting with Scratch programming for young elementary students and advancing through programming in Python and for game development for high school students. I conceived many of the programming classes and have taught programming to novice and more advanced students. We also created an educational computer-aided design tool, called BlocksCAD. We received a grant, on which I was the principal investigator, to further develop BlocksCAD and successfully met our grant objectives. BlocksCAD was spun-off

from Einstein's Workshop and has been part of various incubator/accelerator programs in education including LearnLaunch and Mass Challenge as well as Play Labs at the MIT Media Lab. Today, BlocksCAD is used worldwide in makerspaces as well as in schools throughout the United States.

15. I have reviewed all of the exhibits cited in this declaration, including the following:

Exhibit No.	DESCRIPTION
1001	U.S. Patent No. 6,917,986 to Mor et al. ("986 patent")
1002	Prosecution History of U.S. Patent No. 6,917,986
1004	<i>Curriculum Vitae</i> of Dr. Houh
1005	U.S. Patent No. 6,032,194 to Gai <i>et al.</i> ("Gai")
1006	U.S. Patent No. 7,848,264 to Gai <i>et al.</i> ("Gai 264")
1007	"IEEE Standard for Information Technology, Telecommunications and Information Exchange between Systems, Local and Metropolitan Area Networks, Common Specifications, Part 3: Media Access Control (MAC) Bridges," published as ANSI/IEEE Standard 802.1D (1998) ("802.1D")
1011	U.S. Patent No. 6,330,229 to Jain et al. ("Jain")
1012	U.S. Patent No. 7,200,110 to Burns et al. ("Burns")
1013	U.S. Patent No. 5,995,608 to Detampel et al ("Detampel")
1014	U.S. Patent Publication No. 2003/0021223 to Kashyap ("Kashyap")

16. I understand the '986 patent has a filing date of January 7, 2002. Thus, I have considered certain issues from the perspective of a person of ordinary skill in the art as described below as of the January 7, 2002, filing date of the '986 patent.

17. I have been retained by the Petitioner as an expert in the field of information technology and networking. I am working as an independent consultant in this matter and am being compensated at my normal consulting rate of \$680 per hour for my time. My compensation is not dependent on and in no way affects the substance of my statements in this Declaration.

18. I have no financial interest in the Petitioner. I similarly have no financial interest in the '986 patent and have had no contact with any of the named inventors of the '986 patent.

II. Relevant Law

19. I am not an attorney. For the purposes of this declaration, I have been informed about certain aspects of the law that are relevant to my opinions. My understanding of the law is as follows:

A. Claim Construction

20. I have been informed that claim construction is a matter of law and that the final claim construction will ultimately be determined by the Board. For the purposes of my analysis in this proceeding and with respect to the prior art, I have been informed that IPRs are currently reviewed under “the *Phillips* standard.”

21. I have been informed that under the *Phillips* standard, claim terms are given their plain and ordinary meaning as understood by a person of ordinary skill

in the art at the time of the invention in light of the claim language and the patent specification.

22. I have been informed that embodiments described in the specification are generally encompassed by the claims.

23. I have been informed that the patentee can serve as their own lexicographer. As such, if a claim term is provided with a specific definition in the specific, I should interpret that claim term in light of the particular definition provided by the patentee.

B. Obviousness

24. I have been informed and understand that a patent claim can be considered to have been obvious to a person of ordinary skill in the art at the time the application was filed. This means that, even if all of the requirements of a claim are not found in a single prior art reference, the claim is not patentable if the differences between the subject matter in the prior art and the subject matter in the claim would have been obvious to a person of ordinary skill in the art at the time the application was filed.

25. I have been informed and understand that a determination of whether a claim would have been obvious should be based upon several factors, including, among others:

- the level of ordinary skill in the art at the time the application was filed;

- the scope and content of the prior art; and
- what differences, if any, existed between the claimed invention and the prior art.

26. I have been informed and understand that a claim can be unpatentable in light of a single prior art reference if it would have been obvious to modify that reference to come up with the claimed subject matter.

27. I have been informed and understand that the teachings of two or more references may be combined in the same way as disclosed in the claims, if such a combination would have been obvious to one having ordinary skill in the art.

28. In determining whether a combination based on either a single reference or multiple references would have been obvious, it is appropriate to consider, among other factors:

- whether the teachings of the prior art references disclose known concepts combined in familiar ways, which, when combined, would yield predictable results;
- whether a person of ordinary skill in the art would have implemented a predictable variation, and would have seen the benefit of doing so;

- whether the claimed elements represent one of a limited number of known design choices, and would have a reasonable expectation of success by those skilled in the art;
- whether a person of ordinary skill would have recognized a reason to combine known elements in the manner described in the claim;
- whether there is some teaching or suggestion in the prior art to make the modification or combination of elements claimed in the patent; and
- whether the claim applies a known technique that had been used to improve a similar device or method in a similar way.

29. I understand that one of ordinary skill in the art has ordinary creativity and is not an automaton.

30. I understand that in considering obviousness, it is important not to determine obviousness using the benefit of hindsight derived from the patent being considered.

C. Ordinary Skill in the Art

31. I understand that, in the context of an invalidity analysis, a person having ordinary skill in the art (“POSITA”) is a hypothetical person who looks to prior art at the time of the invention.

32. I further understand that the factors that may be considered in determining the level of ordinary skill in the art include: (1) the problems encountered in the art; (2) the prior art solutions to the problems encountered in the art; (3) the rapidity of innovation; (4) the sophistication of the technology; and (5) the education level of active workers in the field. I understand that these factors need not all be taken into account for the analysis and that one or more of these factors may control.

III. Summary of Opinions

33. It is my opinion that claims 1-2, 4-7, 9-14, 16-19, and 21-24 (“Challenged Claims”) of the ’986 patent are rendered obvious by the prior art identified below:

Ground	Reference	Basis (35 U.S.C.)	Invalid Claims
1A	Gai (downstream embodiment)	§103	1-2, 4-7, 9-14, 16-19, and 21-24
1B	Gai (upstream embodiment)	§103	1-2, 4-7, 9-14, 16-19, and 21-24

IV. Overview of the ’968 Patent

A. Technology Background

1. Spanning Tree Protocol (STP)

34. Computer networks include interconnected devices, such as computers or end stations that can exchange, e.g., transmit and receive, data with

each other. EX1005, 1:20-23 (“A computer network typically comprises a plurality of interconnected entities. An entity may consist of any device, such as a computer or end station, that “sources” (i.e., transmits) or “sinks” (i.e., receives) data frames.”). “One or more intermediate devices are often used to couple LANs together and allow the corresponding entities to exchange information. For example, a switch may be utilized to provide a ‘switching’ function for transferring information, such as data frames, among entities of a computer network. Typically, the switch is a computer and includes a plurality of ports that couple the switch to the other entities.” EX1005, 1:36-43.

35. Computer networks often provide redundant communication paths between the switches in the network to protect the network against failures, for example, if a failure happens, the network will still operate as intended and no particular network portion would be isolated. EX1005, 2:14-16 (“Most computer networks include redundant communications paths so that a failure of any given link does not isolate any portion of the network.”); EX1001, 2:7-9. The figure below illustrates two examples of a *first redundant path* and a *second redundant path* between two pairs of *switches* (*switches 122* and *114* and *switches 116* and *121*).

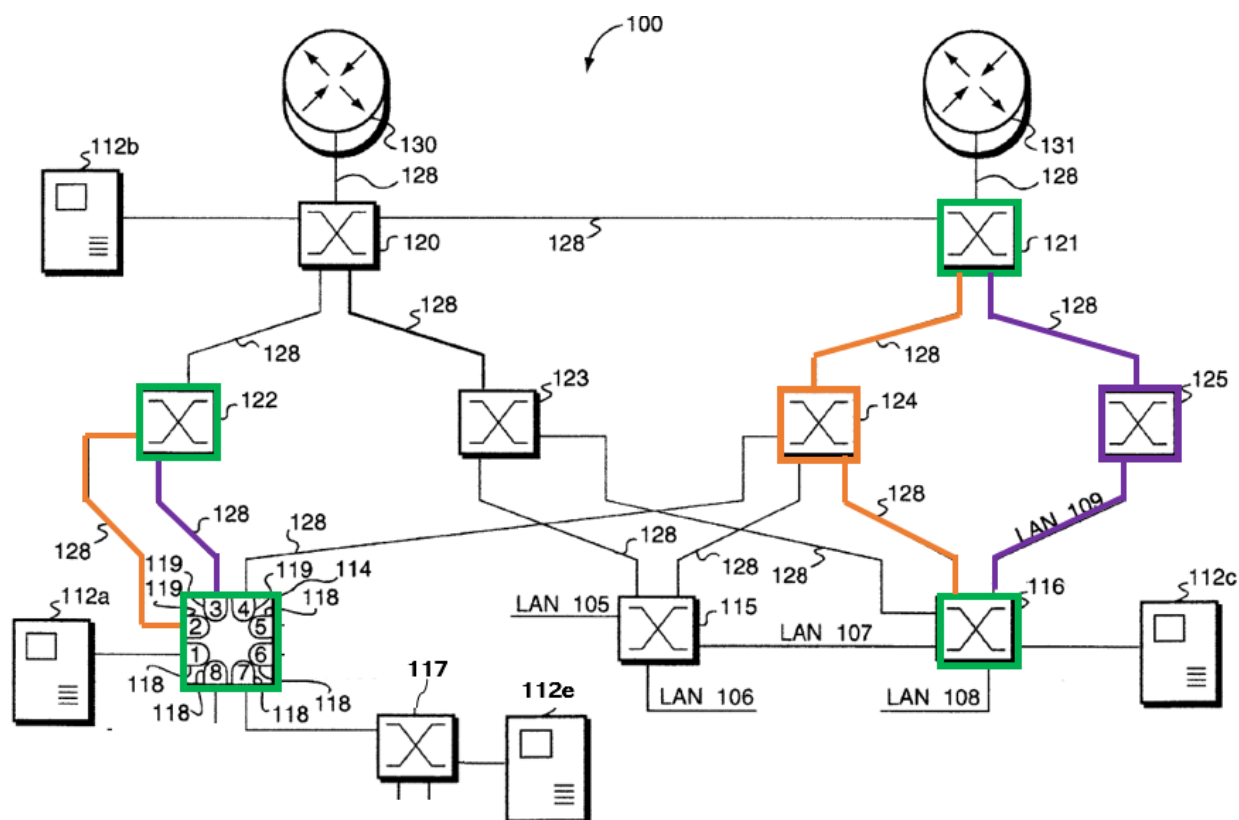


FIG. 1

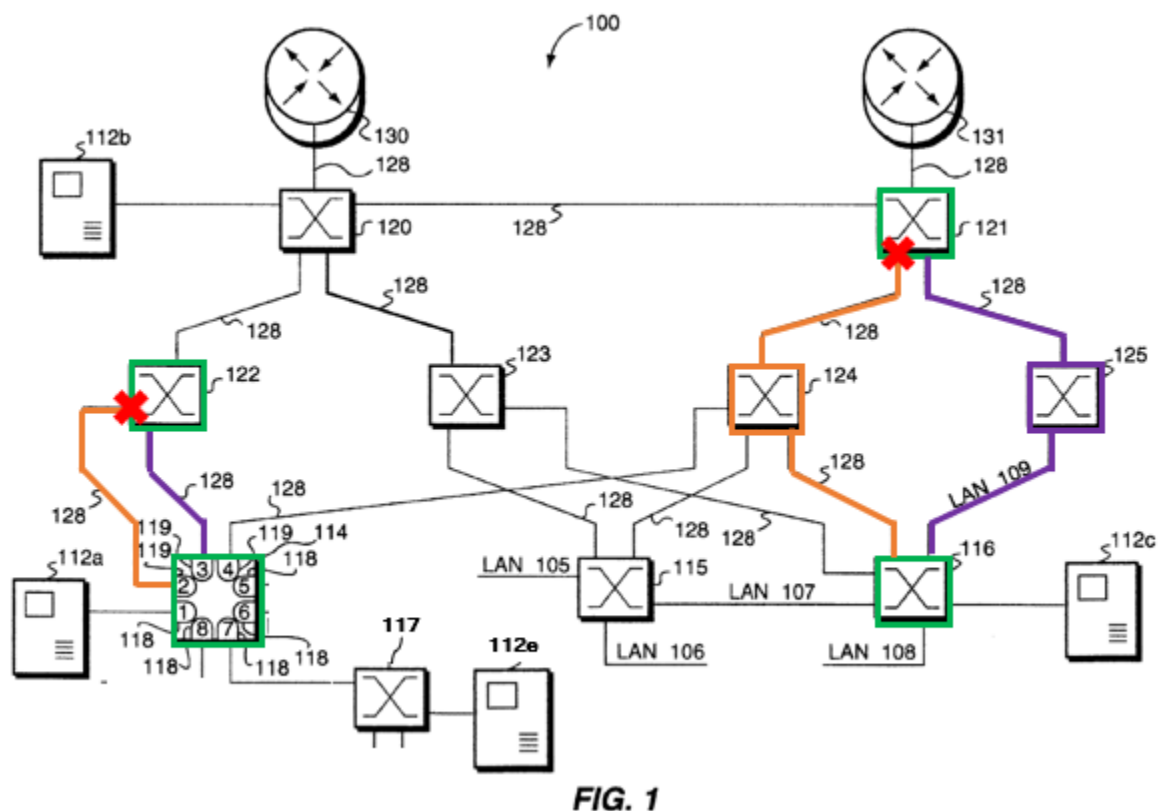
EX1005, Figure 1.

36. The issue with redundant paths is that they create loops within the network that are undesirable because data can traverse the network indefinitely within the loops and this can cause network congestion. EX1005, 2:18-27 (“The existence of redundant links, however, may cause the formation of circuitous paths or ‘loops’ within the network. Loops are highly undesirable because data frames may traverse the loops indefinitely. Furthermore, ... devices ... replicate (i.e., flood) frames ... resulting in a proliferation of data frames along loops. The resulting traffic effectively overwhelms the network.”); EX1001, 2:23-29 (“A

problem with redundant topologies ... is that ... broadcast packets may be forwarded repeatedly by the switches in the loop, causing a serious problem of network congestion...”).

37. To avoid the formation of loops, networks execute a protocol known as the “spanning tree protocol” (STP) that blocks certain redundant paths to ensure only a single communication path exists between any two network entities. EX1005, 2:30-41 (“To avoid the formation of loops, devices, such as switches or bridges, execute a spanning tree algorithm. This algorithm effectively ‘severs’ the redundant links within the network. Specifically, switches exchange special messages called bridge protocol data unit (BPDU) frames *that allow them to calculate a spanning tree* or active topology, which is a subset of the network that is loop-free (i.e., a tree) and yet connects every pair of LANs within the network (i.e., the tree is spanning).”); EX1001, 2:30-46 (“In order to avoid these problems, the 802.1D standard defines a communication protocol, known as the spanning tree protocol (STP), for managing the creation and updating of the network topology. STP ensures that all data paths in a network of bridges are free of loops by disabling forwarding of packets through certain interfaces... The spanning tree algorithm and protocol thus create a tree-shaped active topology from the arbitrarily-connected components of the network.”); EX1007, 76 (“The [spanning tree] algorithm and protocol described in this clause reduce the Bridged LAN

topology to a single Spanning Tree.”); EX1007, Figures 8-1, 8-2. For example, in the prior example, the spanning tree protocol can block certain ports (shown in “X” in the Figure below) leaving only a single communication path (the path shown in *purple*).



EX1005, Figure 1; *see also*, EX1007, 76-103.

2. Root Switch

38. The spanning tree protocol specifies that one switch in the network will act as the “root” switch, which is the switch logically located at “root” of the tree. Any network switch can be elected as the root switch. For example, in the example above, switch 120 can be elected as the “root” switch (“the root (e.g.,

backbone switch 120)” EX1005, 15:28), for example, if it is the switch with the lowest switch identifier or if it reduces the average distance between other switches (e.g., 121-125 and 114-116) and network entities (e.g., entities 112a, 112b, 112c, etc.) or based on a network administrator designating, e.g., forcing, a switch as the root. EX1005, 2:42-47 (“[E]xecution of the spanning tree algorithm causes the switches to elect a single switch, among all the switches within each network, to be the ‘root’ switch. Each switch has a unique numerical identifier (switch ID) and ***the root is the switch having the lowest switch ID numeric value...***”); EX1005, 10:29-33 (“The conventional spanning tree algorithm, however, simply ***selects the switch having the lowest ID to become the root***, regardless of where in the network 100 this switch is located. As a result, an access switch is often elected to be the root...”); EX1007, 78 (“The Bridge with the highest priority Bridge Identifier is the Root (for convenience of calculation, this is the identifier with the lowest numerical value).”); EX1005, 3:17-39 (“Depending on the configuration of a given network, the location of the root can significantly affect the distance that messages must travel. ... To reduce the average distance [between end stations and the primary server] and avoid partitioning of the backbone switches, it is desirable to locate the root at a backbone switch... To identify which switch should be the designated switch, switches... compare information in received BPDUs with their stored information.”); EX1001, 2:48-49 (“The spanning tree algorithm defines one

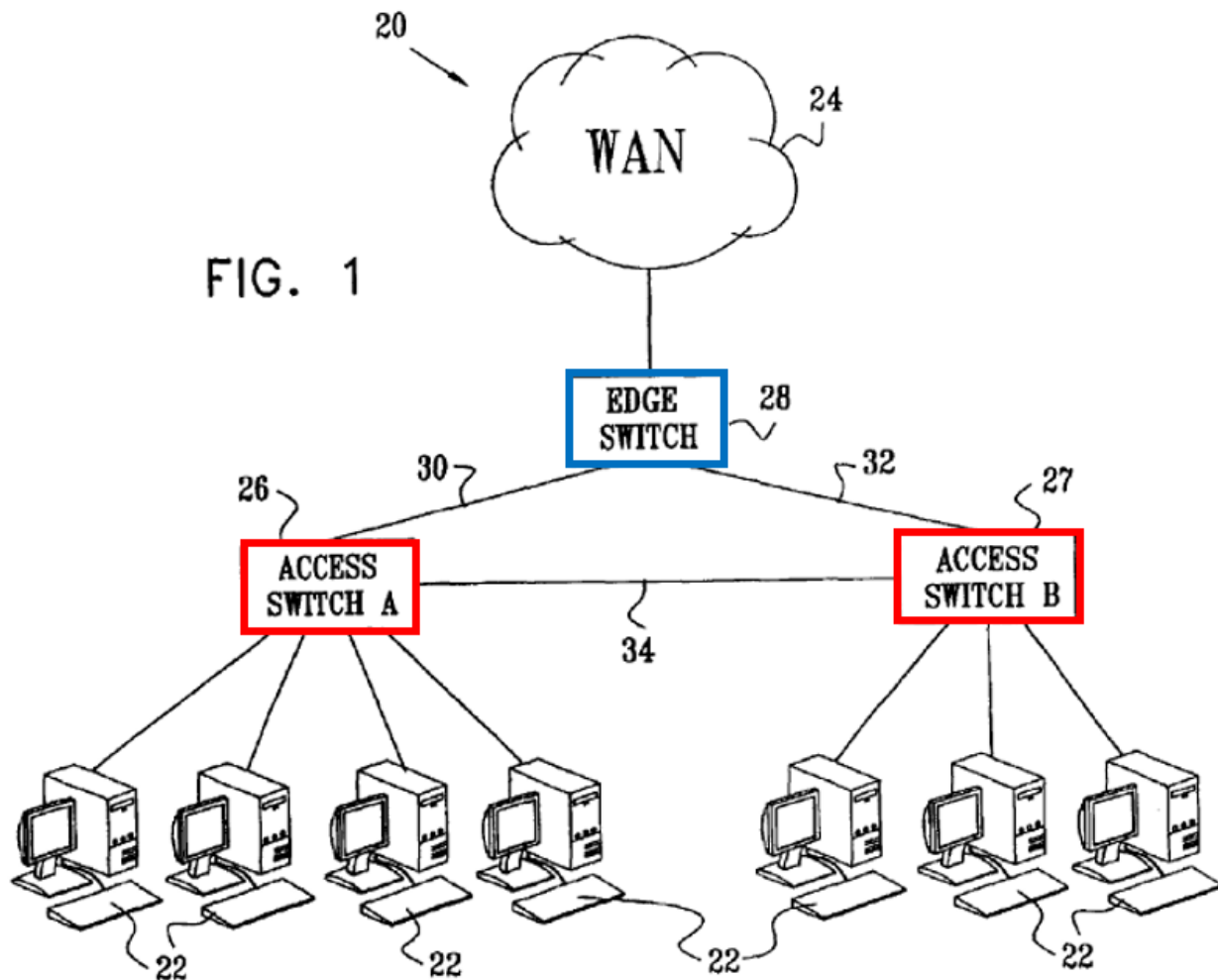
bridge in the network as the root bridge.”); EX1005, 3:33-36 (“[A] network administrator may ‘force’ the network to choose the selected switch as the root.”).

3. Access, Core, And Edge Switches

39. As the '986 Patent explains, switches can be “classified as access switches or core switches.” EX1001, 1:30-31. “An access switch has ports that connect to [local area networks] LANs and/or to other network endpoints, such as user computer systems.” EX1001, 1:31-33. “Core 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.” EX1001, 1:33-36. Moreover, the '986 Patent classifies as edge switches the core switches that are directly connected to access switches. EX1001, 1:36-38 (“Core switches that are directly connected to access switches are referred to herein as edge switches.”).

40. In the exemplary network below from the '986 Patent, **switches 26** and **27** are access switches because they have ports that connect to LANs and/or to other network endpoints, such as user computer systems 22. *See also*, EX1001, 7:12-14 (“Network 20 serves multiple users 22, who are connected via access switches 26 and 27 (marked access switches A and B) to a WAN 24.”). **Switch 28** can act as a root, core, and an edge switch. First, it can be a root switch if it has the lowest switch identifier. Second, it is a core switch because it interconnects **access**

switches 26 and **27** into a WAN by direct connection between the core switches and access switches and/or by connection to other core switches. Third, **switch 28** is also an *edge switch*, because it is a core switch that is directly connected to an access switch, e.g., **access switches 26** and **27**.

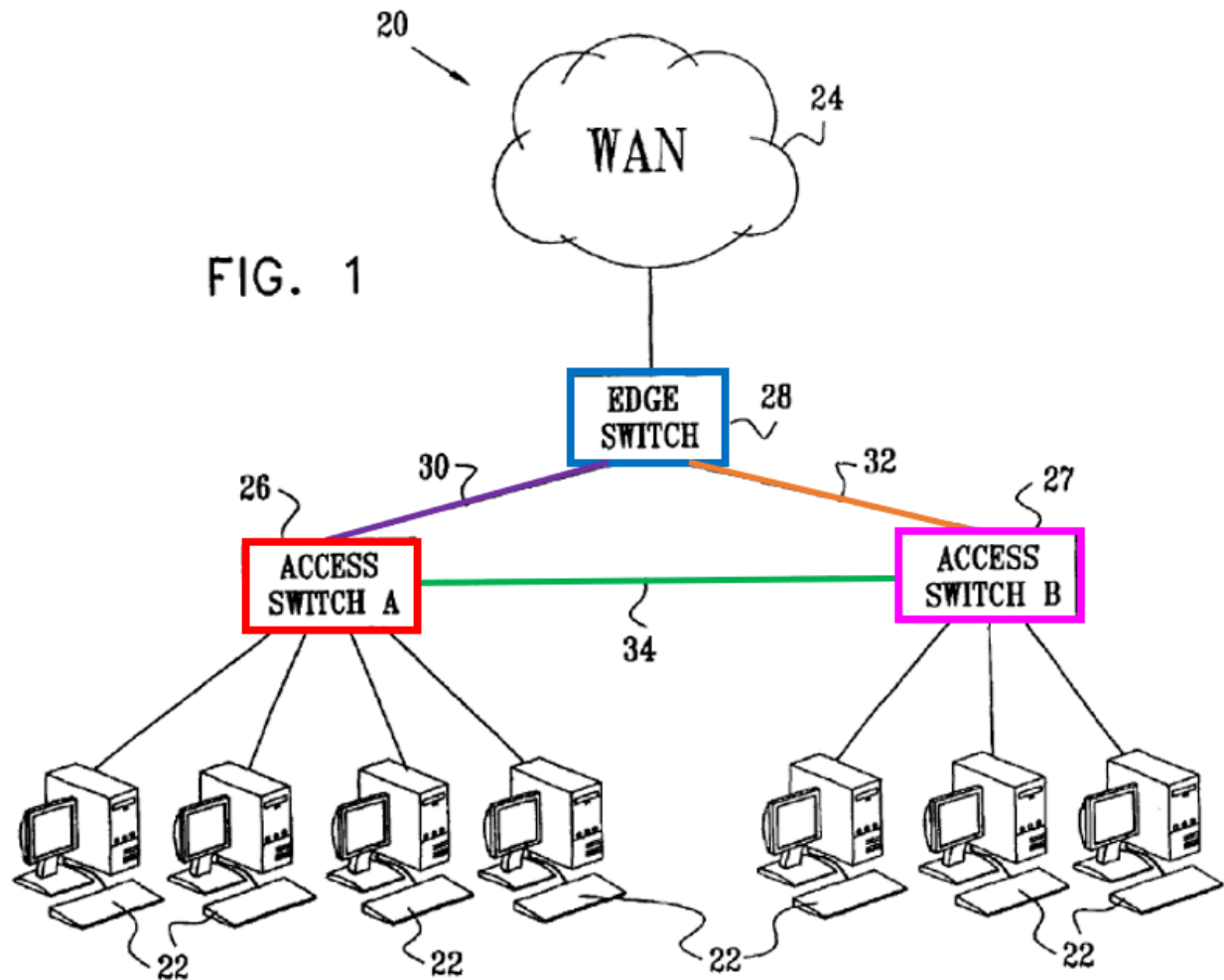


EX1001, Figure 1.

4. Computer Network Having Redundant Links

41. Figure 1 of the '986 Patent, which is reproduced below, shows **edge switch 28** and access switches **26** and **27** communicating with redundant links to

form network 20 that serves multiple uses 22. Specifically, “FIG. 1 is a block diagram that schematically illustrates a computer network 20, which is configured with redundant links for fast protection of user access in case of network failure...” EX1001, 7:8-11. As shown, “[n]etwork 20 serves multiple users 22, who are connected via **access switch[] 26** and **[access switch] 27** (marked access switches **A** and **B**) to a WAN 24.” EX1001, 7:12-14. Further, “[e]ach of the access switches is connected by a respective **access link 30** or **[access link] 32** to an **edge switch 28**, which is in turn connected to the core WAN [24].” EX1001, 7:14-16. “A lateral link 34 between access switches **26** and **27** is used to provide protection in case of a failure in one of access links **30** or **32** or in the corresponding ports of the access switches or edge switch.” EX1001, 7:16-20.

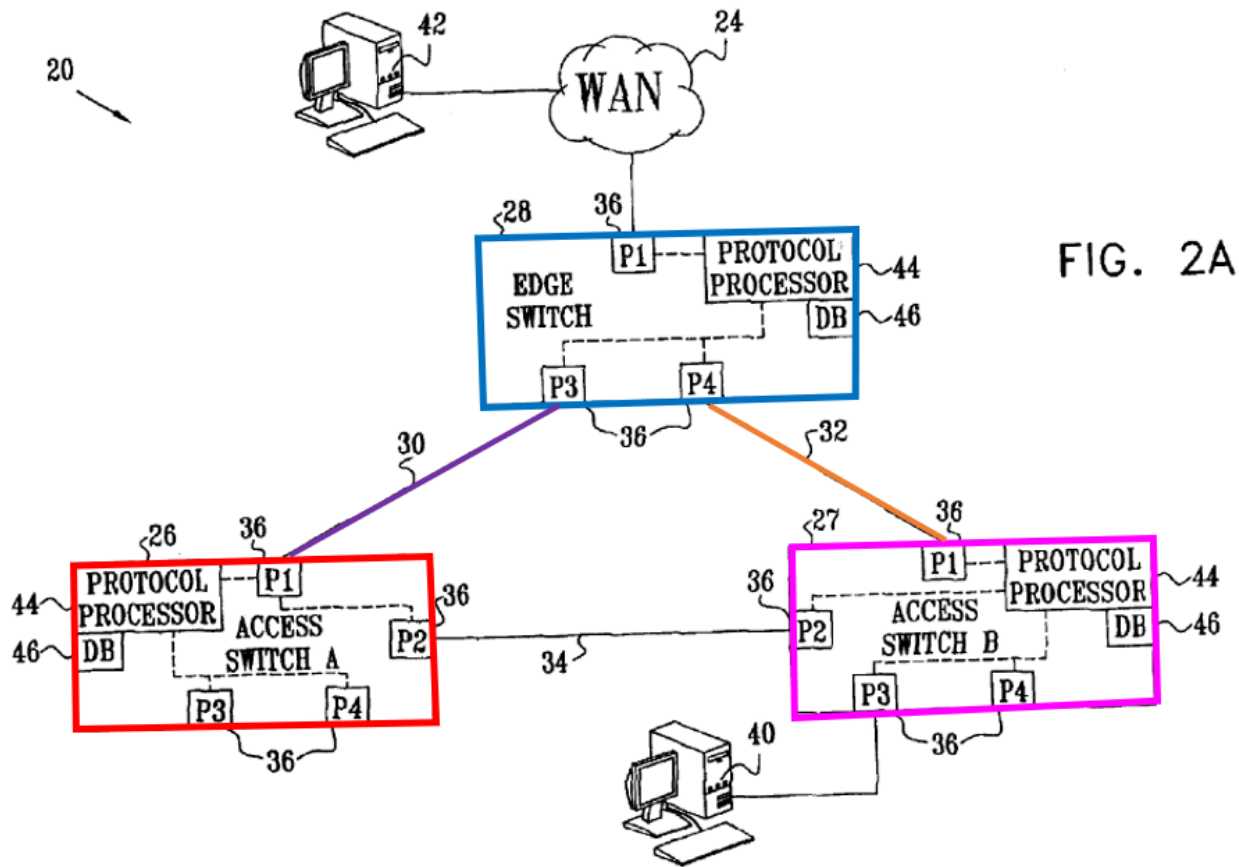


EX1001, Figure 1.

42. Switches have ports that connect to communication links. For example, Figure 2A of the '986 Patent, which is reproduced below, shows that “[e]ach of the switches has multiple ports 36, labeled P1, P2, P3, P4, as well as a protocol processor 44, which is responsible for configuring and controlling the ports.”. EX1001, 7:55-58. The '986 Patent explains that “[t]he functions of processor 44 include building and updating a filtering database (DB) in a memory

46 and running STP.” EX1001, 8:1-3. In each switch, “the database includes entries associated with a plurality of the ports...” EX1001, 5:34-35.

43. The '986 Patent also explains that switches that implement the spanning tree protocol can manage the redundant links in the network. Specifically, “processor 44 in **edge switch 28** is responsible for managing redundant links **30** and **32**,” (EX1001, 8:1-5), and can “select[] one of links **30** and **32** to serve as its active link to access switches **26** and **27**, while the other link is placed on standby.” EX1001, 8:11-13.



EX1001, Figure 2A.

44. For example, the '986 Patent assumes in the example discussed in the Specification that **link 30** is the active link. EX1001, 8:13-14 (“For the sake of example, let **link 30** be the active link.”) The '986 Patent further describes that:

Processor 44 accordingly puts port P3 of **switch 28** in the forwarding state, while putting port P4 in blocking state. This means that P4 will transmit no frames over **link 32**, and will discard any frames it receives on the link. ... Therefore, when STP is run by processors 44, no traffic passes through **link 32**, and thus no loop exists, so that STP maintains port P1 of **switch 27** in the forwarding state. Since **switch 27** will receive no frames on port P1, it will also create no entries for this port in its filtering database.

EX1001, 8:14-24.

45. Because no data frames pass through **link 32**, all traffic directed to **switch 27** passes through **link 30** and link 34. EX1001, 8:62-64 (“[W]hen users 40 and 42 attempt to communicate over network 20, all the traffic between them will pass through **switch 26**, via links **30** and 34.”)

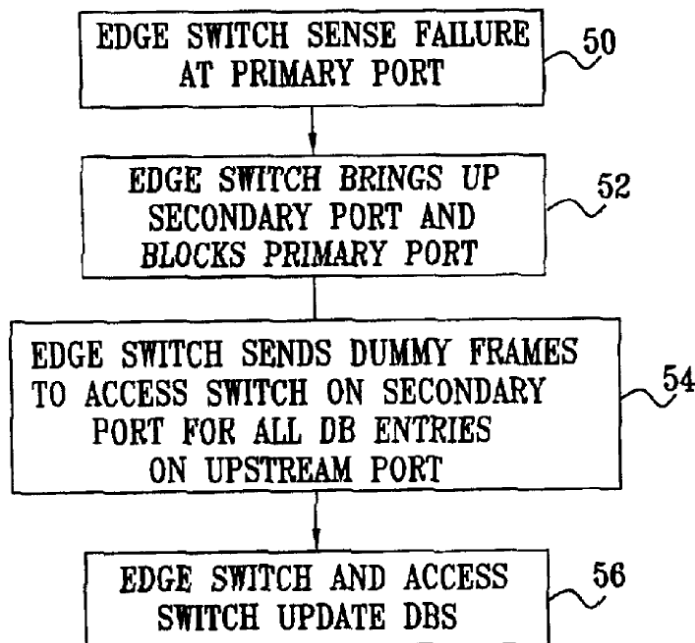
B. Purported Invention Of The '968 Patent

46. The '986 Patent purportedly provides “methods and devices for providing reliable, fault-resistant network access.” EX1001, 1:6-8.

47. Figure 3 of the '986 Patent, which is reproduced below, “is a flow chart that schematically illustrates a method by which **edge switch 28** responds to a failure in **link 30**...” EX1001, 9:11-13. The method depicted on Figure 3 shows

that processor 44 of **edge switch 28** senses a failure at the primary port P3 (step 50) and then activates the secondary port P4, while blocking port P3 (step 52. *See* EX1001, 9:13-18 (“The method is invoked when processor 44 of switch 28 senses a failure at port P3, at a failure sensing step 50. This port is referred to here as the primary port, i.e., the port that is used originally at switch 28, and which presumably continues to be used until a failure occurs.”); EX1001, 9:25-30 (“Upon sensing the failure, processor 44 of **edge switch 28** blocks its [primary] port P3 and activates its secondary port P4 instead...”)).

FIG. 3



EX1001, Figure 3.

48. Then, “**edge switch 28** sends dummy frames downstream to **access switch 27** via **link 32**, at a dummy transmission step 54. For every entry in the filtering database of the **edge switch** that is not associated with the failed port (P3 in the present example), the **edge switch** sends a dummy frame over **link 32** to **access switch 27**...”. EX1001, 9:42-48. “When **switch 27** receives the dummy frame, it also forwards it to **switch 26**.” EX1001, 9:56-57.

49. “Once the dummy frame has been sent, switches **26**, **27** and **28** update their filtering databases to reflect the change in the active link from **link 30** to **link 32**, at a database update step 56. **Edge switch 28** simply replaces the failed port (P3) in the entries in its database with the new active port (P4) that has taken its place.” EX1001, 9:61-67.

C. Prosecution History

50. The '986 Patent was filed on January 7, 2002. EX1001, 1. I understand that the Examiner did not issue any Office Actions rejecting the application claims. Instead, the Examiner issued a Notice of Allowance on March 10, 2005. EX1002, 59.

V. Overview of the Prior Art

A. Gai (“US 6,032,194”)

51. Gai is a U.S. Patent by Cisco filed on December 24, 1997, and issued on February 29, 2000, and therefore is prior art to the '986 Patent, because both the

filing and issuance dates predate the priority date of January 7, 2002, of the '986 Patent.

52. I understand that Gai was discussed by the Applicant of the '986 Patent in the background section of the '986 Patent. *See* EX1001, 3:15-41. Specifically, the background section of the '986 Patent describes that “US. Pat. No. 6,032,194 [“EX1005” or “Gai”] ... describes a method for rapidly reconfiguring a computer network, based on certain modifications to the standard STP.” EX1001, 3:15-18. Specifically, as the '986 Patent admits Gai is directed to a “method that rapidly reconfigures a computer network having a plurality of devices executing the spanning tree algorithm” (EX1005, Abstract) and “more specifically, to a method and apparatus for rapidly reconfiguring a computer network *following a network change.*” EX1005, 1:14-17.

1. Gai's Computer Network Implements A Spanning Tree Protocol

53. Figure 1 of Gai, which is reproduced below, shows exemplary “network 100 [that] comprises a plurality of local area networks (LANs) 102-109 each of which is coupled to one or more access switches 114-116. **Access switches 114-116** are interconnected with each other through a plurality of **backbone switches 120-125.**” EX1005, 7:4-9. “Each **access switch 114-116** and **backbone switch 120-125**, moreover, includes a plurality of ports which, if

coupled to a LAN 102-109, are referred to as local ports 118 and if coupled to links 128 are referred to as trunk ports 119.” EX1001, 7:14-18.

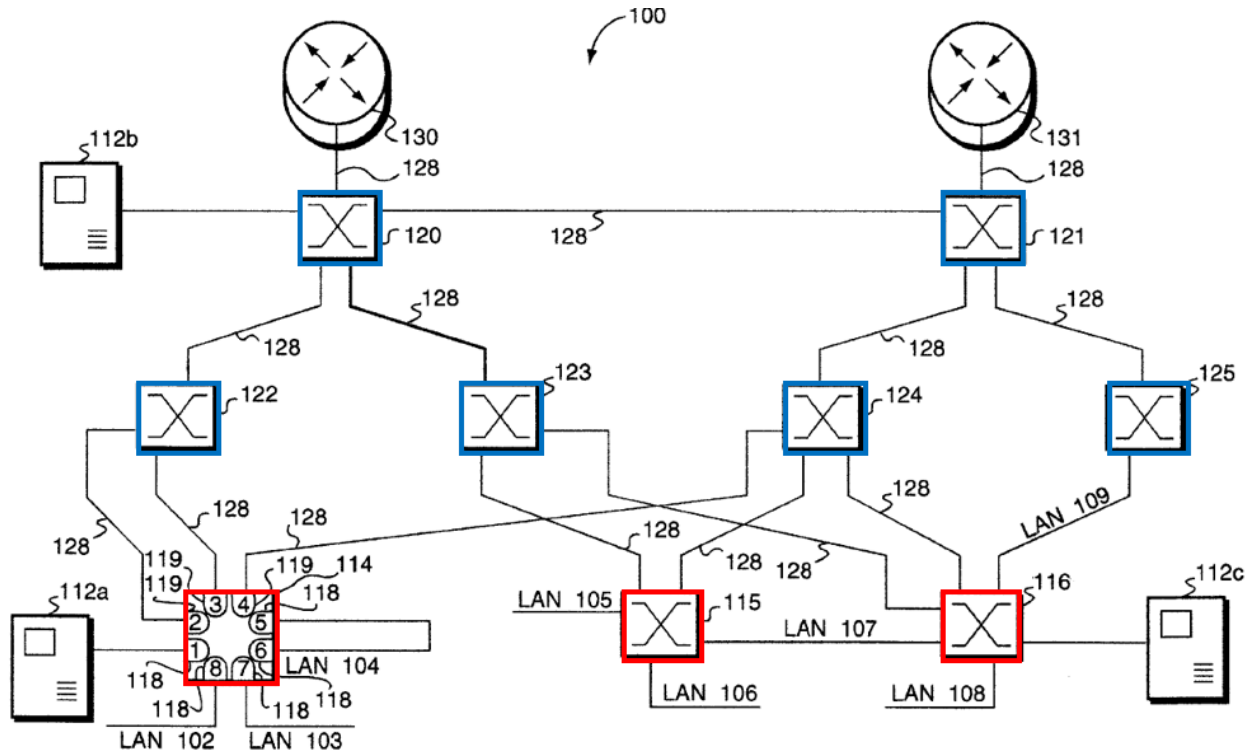


FIG. 1

EX1005, Figure 1.

54. In Gai, the **access switches** are connected to the **backbone switches** through point-to-point links 128 or shared media, such as LANs. EX1005, 7:9-13 (“**[A]access switches 114-116** are generally connected to the **backbone switches 120-125** through a series of links 128, such as point-to-point links, although they may be connected via a shared media (e.g., LAN 109).”); *see also id.*, 7:43-51 (“**[O]ne or more distribution switches** may be interleaved between the **backbone**

switches and the **access switches**, two **access switches** may be directly connected, etc. It will also be understood to those skilled in the art that there is no distinction from the spanning tree point of view between local and trunk ports or point-to-point links and shared links (e.g., LANs). The distinction between local and trunk ports is made for purposes of explanation only.”)

55. Gai discloses connecting the **access switches** to the **backbone switches** using *redundant communication paths*. EX1005, 7:53-55 (“As shown, network 100 includes *redundant communication paths* interconnecting the access and backbone switches 114-116 and 120-125.”). The *redundant communication paths* provide alternative links to portions of a network, but may result in loops, which are undesirable. EX1005, 7:59-61 (“The existence of such redundant links prevents portions of the network 100 from becoming isolated should any constituent link or device fail. Such redundancy, however, also results in the creation of loops, which, as described above, are highly undesirable.”). As such, Gai teaches that in order “[t]o avoid the creation of loops, these switches execute the spanning tree algorithm...” EX1005, 7:59-61.

56. For example, similar to the '986 Patent, Gai explains that only one switch port and its associated path that provides access to the root will be forwarding. EX1005, 11:8-11 (“Significantly, for each access switch 114-116, *only one port (local or trunk) that represents a path from the access switch to the*

root (i.e., provides connectivity to the root through links, shared media, switches, etc.) **will be forwarding**.) In addition, “[a]ll other ports (local or trunk) that represent paths from the access switch to the root will be blocked. In other words, only one port at each access switch 114-116 that provides connectivity to the root will be forwarding.” EX1005, 11:8-15.

57. In the exemplary network shown in Figure 1 of Gai below, for **access switch 114** having a path to root switch 121, the active port may be designated as port 3, highlighted below with a yellow circle. See EX1005, 11:48-49 (“With reference to switch 114, *for example*, assume that port number three, [is] in the forwarding state....”).

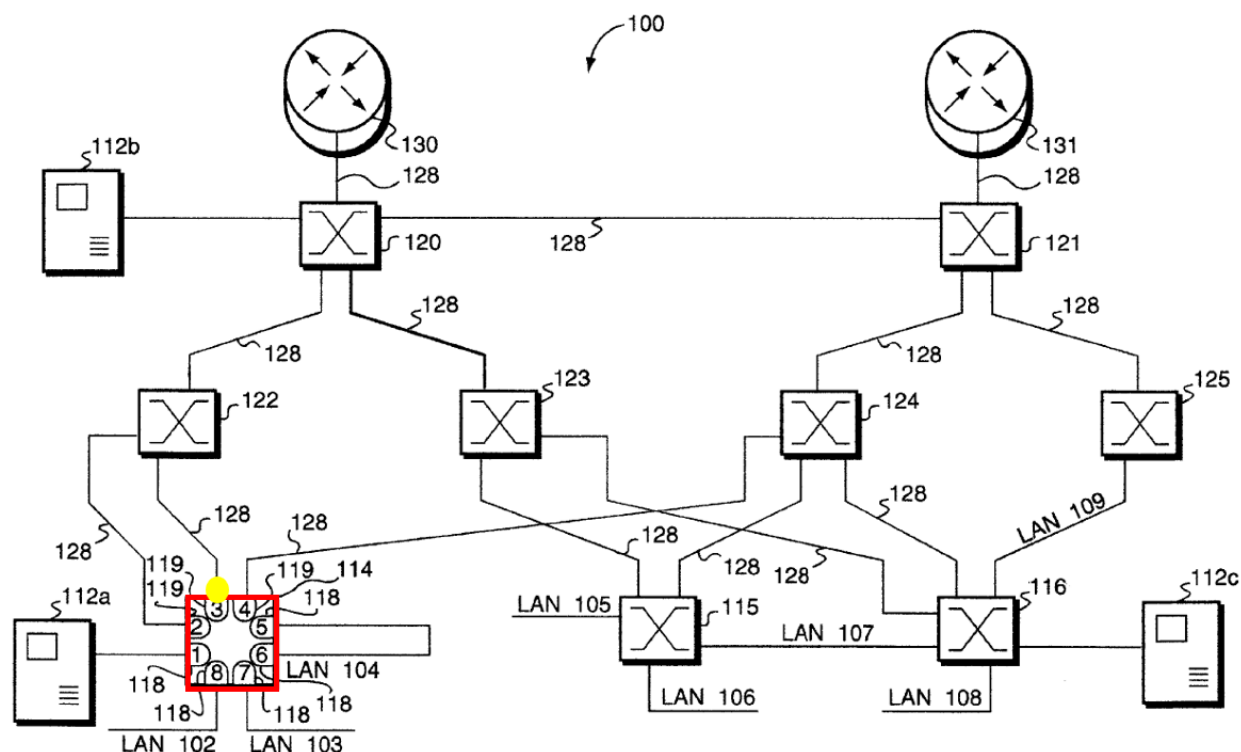


FIG. 1

EX1005, Figure 1.

2. One Exemplary Failure In Gai's Access Switch 114

58. Gai explains that failures can happen to links or ports of switches in the network. *See e.g.*, EX1005, 2:14-16 (“[M]ost computer networks include redundant communications paths so that *a failure of any given link* does not isolate any portion of the network”); EX1005, 4:1-3 (“If a switch stops receiving BPDUs on its root port, indicating a possible link or device failure, the corresponding timer will expire and the information will be discarded”); EX1005, 5:35-47 (“The network preferably includes *a plurality of devices* executing the

spanning tree algorithm so as to elect a root and place the ports of the devices in either a forwarding or blocked state. In accordance with the method, *one or more devices* are configured and arranged so that one trunk port is in the forwarding state and other trunk ports are in the blocked state. Additionally, one or more of the blocked ports are designated as back-up ports. *Upon detection of a failure at the active forwarding port*, the state of one of the back-up ports immediately transitions from blocked to forwarding, thereby becoming the new active port for the device.”); EX1005, 12:4-6 (“Rather than wait for the corresponding BPDU information to time out, a link failure may alternatively be detected by a link integrity test...”).

59. Specifically, Gai provides one example of a failure at the link (link 128) that is coupled to the forwarding port (port number 3) of access switch 114. *See e.g.*, EX1005, 11:36-38 (“Should a change occur in the network 100, *such as* a failure disabling the link coupled to a root port of an access switch...”); EX1005, 11:41-44 (“With reference to switch 114, *for example*, assume that port number three, which is coupled to backbone switch 122, is the root port for switch 114 and thus in the forwarding state.”); EX1005, 11:48-49 (“With reference to switch 114, *for example*, assume that port number three, [is] in the forwarding state.... *If* the link 128 coupled to the port number three at switch 114 fails...”).

60. In this particular example, Gai explains that “[u]pon 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.” EX1001, 3:26-28. Specifically, either back-up trunk **ports 2 or 4** highlighted in red dots in the figure below will become the new active port of **switch 114**. EX1005, 11:48-52 (“If the link 128 coupled to the port number three at switch 114 fails, either one of these two other ports (i.e., port numbers two or four) will immediately transition to the forwarding state and begin receiving and sending messages.”).

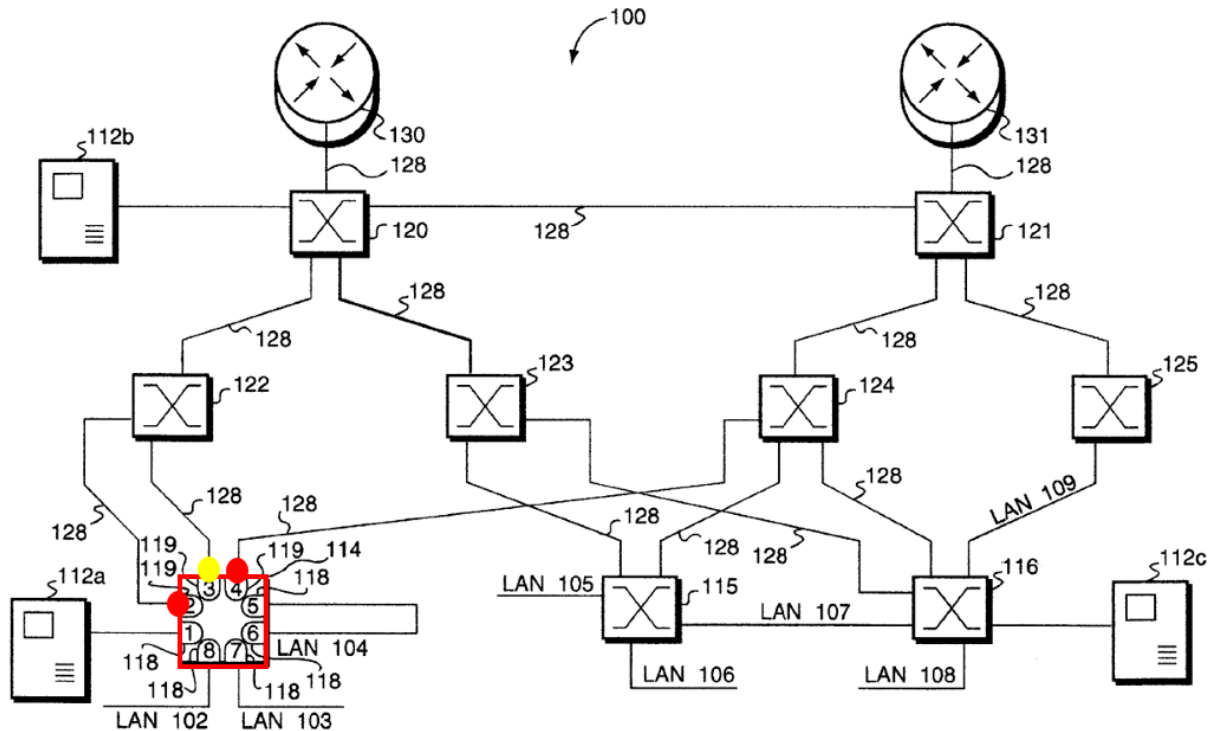


FIG. 1

EX1005, Figure 1.

61. Gai further discloses that “[u]pon transition to the new forwarding port, the device begins transmitting ‘dummy’ multicast messages through the new port” (EX1005, 5:54-56) to inform the network that a new topology exists for transmission of messages through access switch 114. Specifically, Gai explains that “[a]s ports transition between the blocked and forwarding states, entities may appear to move from one port to another. To prevent switches from distributing messages based upon incorrect information, switches quickly age-out and discard the ‘old’ information in their filtering databases.” EX1005, 4:30-34. This is

achieved by transmitting “dummy multicast message 400” and optionally, an additional “Topology Change Notification Protocol Data Unit (TCN-PDU) frame.”
See EX1005, 15:12-18; EX1005, 4:34-37.

62. Specifically, Gai explains that “the transmission of dummy multicast messages 400, as described above, quickly informs devices of newly activated ports, thereby reducing the flooding that otherwise occurs when the contents of filtering databases 244 are discarded.” EX1005, 15:21-25.

3. Additional Embodiments of Gai Not Discussed In The '986 Patent

63. In the Specification of the '986 Patent, the Applicant distinguished Gai from the purported invention of the '986 Patent by representing that the '986 Patent “manage[s] its redundant *downstream* ports in a novel way” (EX1001, 4:37-38), e.g., by “sending *dummy traffic...downstream*” (EX1001, 12:4-5). Specifically, the Applicant contrasted the transmission of dummy traffic in the *downstream direction* in the '986 Patent from Gai, which the Application represented that transmitted dummy messages in the *upstream direction*, i.e., toward the root. *See* EX1001, 4:38-42 (“This modification [in the '986 Patent] is in contrast to the methods described in [Gai], which require network entities to be aware of and manage redundancies in their *upstream ports*...”); EX1001, 3:30-31

(“The switch then transmits *dummy multicast messages* toward the root (*upstream*) through the newly-active trunk port.”).

64. This distinction is true when considering only the *example* of a failure on an *upstream* port and sending dummy traffic *upstream* (EX1005, 11:36-14:3). However, the claims of the '986 Patent are not limited to only sending dummy traffic downstream. For example, only claims 11, 12, 23 and 24 recite such “*downstream*” transmission of dummy traffic. See EX1001, 11:53-55; 12:4-5; 13:7-14:15. The scope of claim 1, which does not recite “downstream,” includes sending dummy traffic upstream. In addition, claim 11, which depends from claim 1, recites “wherein sending the dummy traffic comprises sending data frames downstream from the edge switch toward the access switches.” EX1001, 11:53-55. I have been informed and understand, that because claim 11—as a dependent claim—should further narrow the scope of claim 1, this further confirms that the scope of claim 1 should cover both sending downstream and upstream data frames. This is because—as I have been informed—the scope of claim 1 should cover the scope of claim 11, but also be broader. Therefore, because upstream transmission is the only alternative to downstream transmission, a scope of claim 1 that is broader of the scope of claim 11 should cover both the upstream and the downstream transmission of dummy frames.

65. Indeed, Figure 1 of Gai, which is reproduced below, shows annotations of **switch 124**, **access switch 115** and **access switch 116** that are similar to the '986 Patent's **edge switch 28**, **access switch 115** and **access switch 116**. See, e.g., EX1001, 3:18-24 ("The method assumes that the switches in the network, including both *core and access switches*, have multiple trunk ports providing connectivity to the STP root via trunk switches. (These trunk switches are *comparable to the core switches and edge switches defined above...*)."). Applicant did not discuss any of **edge switch 28**, **access switch 115** and **access switch**.

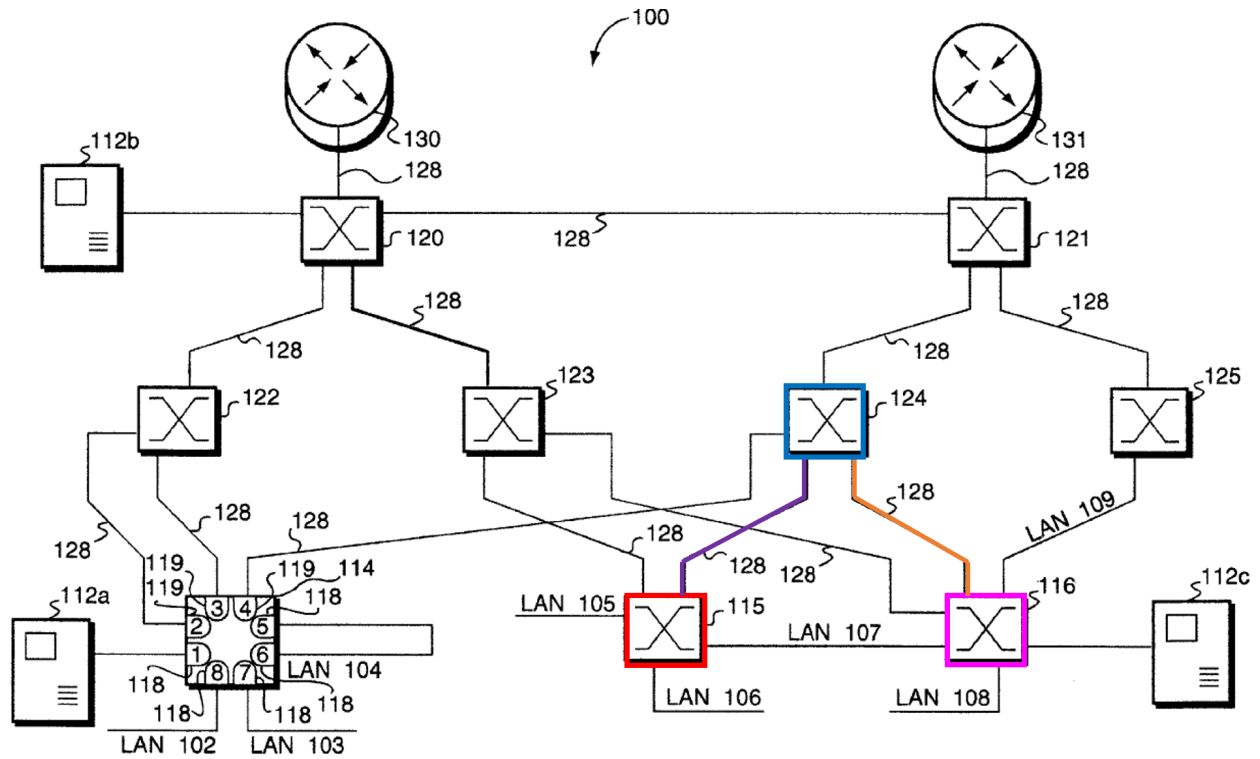
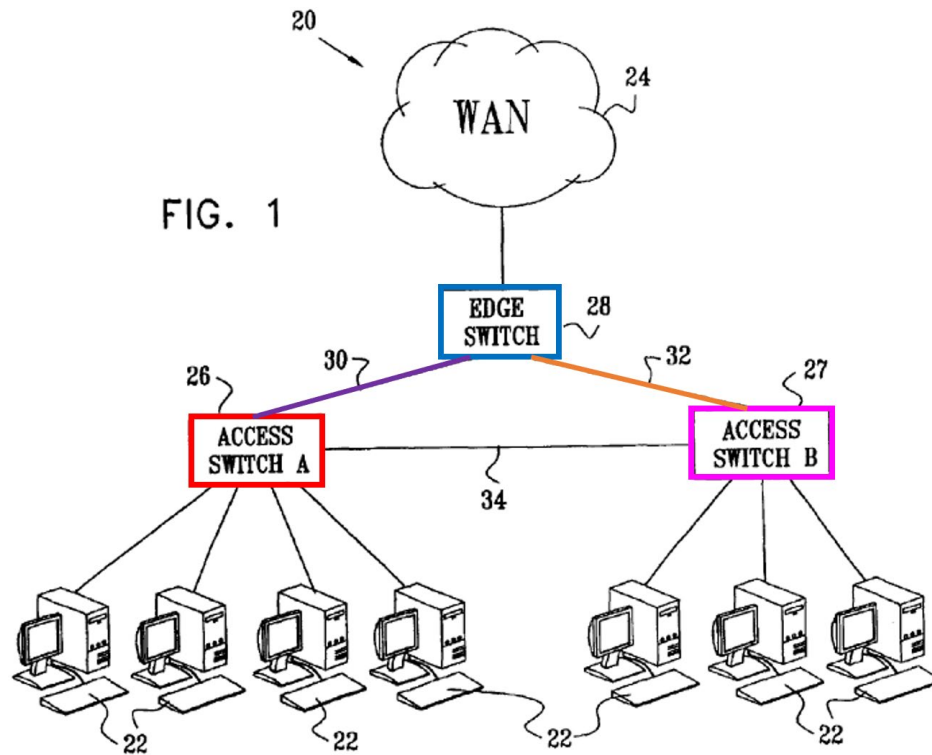


FIG. 1

EX1005, Figure 1.



EX1001, Figure 1.

66. Just like **access switches 26** and **27** of the '968 Patent, switches **115** and **116** of Gai are *access switches* because they connect to LANs and/or to other network endpoints, such as user computer systems. See EX1001, 1:31-33 (“An *access switch* has ports that connect to LANs and/or to other network endpoints, such as user computer systems.”).

67. Just like **edge switch 28** of the '968 Patent, **switch 124** of Gai is an *edge switch* because it is a core switch¹ that is directly connected to **access switches 115** and **116**. See EX1001, 1:35-38 (“Core switches that are directly connected to access switches are referred to herein as edge switches.”).

68. Also, similar to redundant links **30** and **32** of the '986 Patent, **first redundant link 128** couples **switch 124** to access **switch 115** and **second redundant link 128** couples **switch 124** to **access switch 116**.

69. Moreover, as shown below, “assum[ing] that backbone switch 121 (FIG. 1) is the root” (EX1005, 14:16-17), the links identified with a **green arrow** indicate possible paths from **access switch 116** to the root switch (identified with a **purple arrow**). That is, one possible path includes sequentially **access switch 116**, **access switch 115**, **switch 124**, root switch 121, and the links (LAN 107 and links 128) coupling those switches.

¹ **Switch 124** is a core switch because it interconnects access switches. See EX1001, 1:33-36 (“Core 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.”)

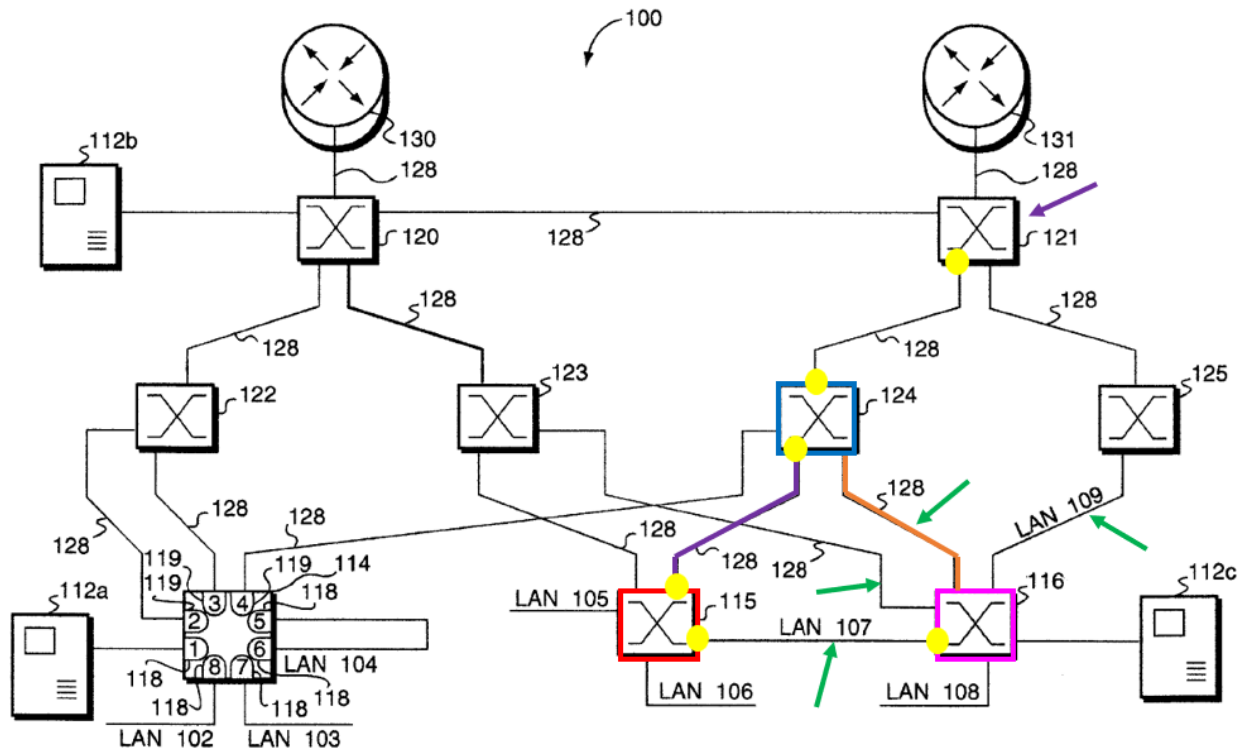


FIG. 1

EX1005, Figure 1.

70. Accordingly, when the STP algorithm is implemented, only the port identified with a yellow dot on the left side (as depicted) of **access switch 116** will be forwarding, while the other ports of **access switch 116** coupled to the other links identified with a **green arrow** will be blocked. *See* EX1005, 5:36-39 (“The network preferably includes a plurality of devices executing the spanning tree algorithm so as to elect a root and place the ports of the devices in either a forwarding or blocked state.”). And because Figure 1 implements a spanning tree algorithm (i.e., there is only one path from the root to a component in the network),

the ports identified with yellow dots and the corresponding links provide the path for a packet from the root port (yellow dot on switch 121) to the **access switch 116** and vice versa.

71. While Gai explicitly discusses the event of a failure in access switch 114, a POSITA would have readily recognized that failures can happen in any of the other switches, such as in one of the ports of **switch 124**. EX1012, 2:56-59 (“A ... *port may fail for various reasons*, including a software defect, an equipment failure in the call control infrastructure or a failure in the transmission link.”). In that case, the active port that failed would be blocked (see **purple X** below) and the port (see **red circle** below) to **access switch 116** would be unblocked making **link 128** available for transmission. And a POSITA would have readily understood that Gai teaches (or at least renders obvious) sending dummy traffic through the opened port (i.e., downstream) to inform the network that a new topology is now implemented for communication across the network.

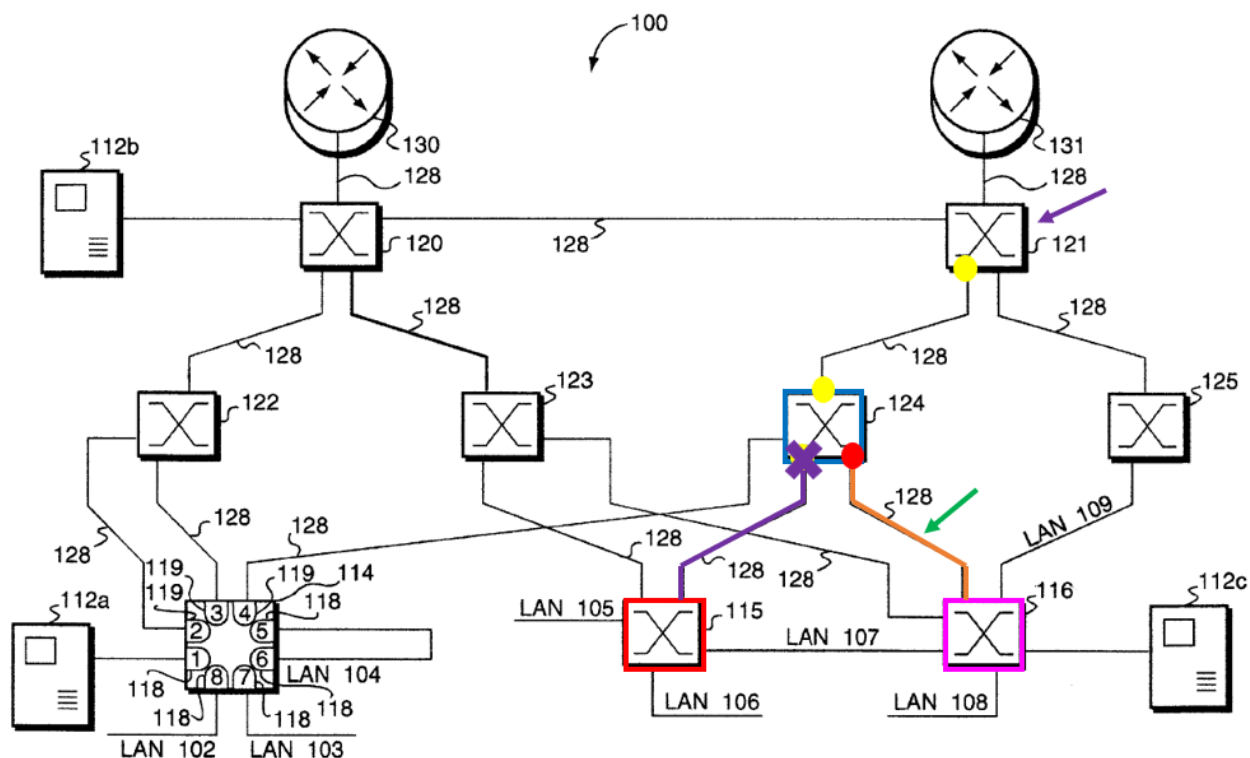


FIG. 1

EX1005, Figure 1.

VI. Level of Ordinary Skill in the Art

72. In my opinion, a person of ordinary skill in the art (“POSITA”) in the field of the ’986 patent as of its effective filing date of January 7, 2002, would have had a bachelor’s degree in computer science, computer engineering, electrical engineering, or in an equivalent field, or approximately two years of experience working in the field of information technology and networking. In my opinion, additional education could make up for less practical experience, and vice versa.

73. I met these requirements for POSITA at the time of the effective filing date of the '986 patent.

VII. Claim Construction

74. I have reviewed the claims, specification, and file history of the '986 patent.

75. I did not find any explicit definition of claim terms. I have applied the ordinary and customary meaning of the claim terms as understood by a POSITA in my analysis provided herein.

VIII. Specific Grounds for Challenge

A. Ground IA: Gai (Downstream Embodiment) Renders Obvious Claims 1-2, 4-7, 9-14, 16-19, and 21-24

1. Independent Claims 1 and 12

i. [1pre/12pre] A method for network communication, comprising:

76. I understand that the preamble may not be limiting. Nevertheless, Gai discloses and/or renders obvious the preamble. For example, Gai discloses a method for rapidly reconfiguring a computer network that has many devices that have ports and can communicate with a root switch. EX1005, Abstract (“***A method that rapidly reconfigures a computer network*** having a plurality of devices executing the spanning tree algorithm. First, one or more devices are configured and arranged so that one port, ***providing connectivity to the root***, is in

the forwarding state and the remaining ports, providing connectivity to the root, are in the blocked state. ... The method is also compatible with networks supporting virtual local area network (VLAN) designations and allows load balancing among different VLANs.”)

77. Figure 1 of Gai, which is reproduced below, shows an exemplary network according to Gai's disclosure.

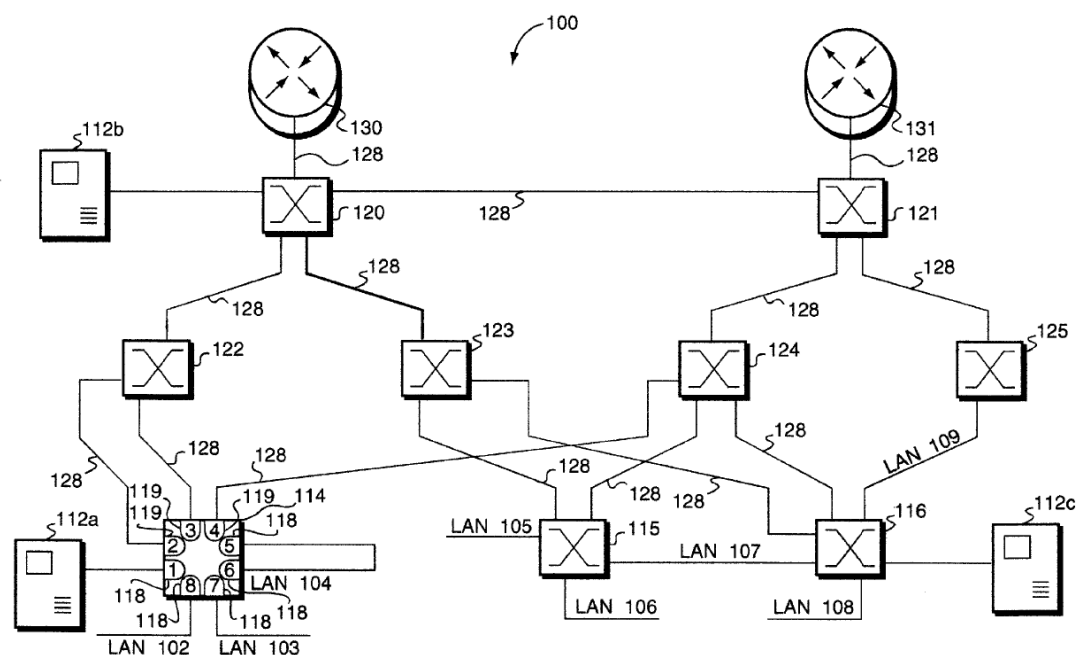


FIG. 1

EX1005, Figure 1.

ii. [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**;

[12a] 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**;

78. Gai discloses and/or renders obvious limitations [1a/12a].

79. *First*, as the '986 Patent explains, a “switch” is one particular type of a “bridge.” *See, e.g.*, EX1001, 1:27 (“The most common type of MAC bridge is a switch...”); 4:50-54 (“Although preferred embodiments are described herein with reference to certain switches, and particularly to edge switches and access switches used in a WAN, the principles of the present invention are generally applicable to bridges of other types and to other network configurations.”).

80. *Second*, Gai discloses an **edge switch** and one or more **access switches**. As the '986 Patent admits, Gai discloses the claimed **edge switch** and **access switch(es)**. EX1001, 3:16-24 (“[Gai] describes a method for rapidly reconfiguring a computer network, based on certain modifications to the standard STP. The method assumes that the switches in the network, including both core and **access switches**, have multiple trunk ports providing connectivity to the STP root via trunk switches. (These trunk switches are comparable to the core switches

and **edge switches** defined above, and the trunk ports are on the upstream side of each switch, i.e., the side leading toward the root.)”).

81. Indeed, as shown below, Gai’s “**backbone**” **switch 124** meets the claimed **edge switch/first bridge** and Gai’s **access switch 115** and **access switch 116** meet the claimed **access switches/second bridge**.

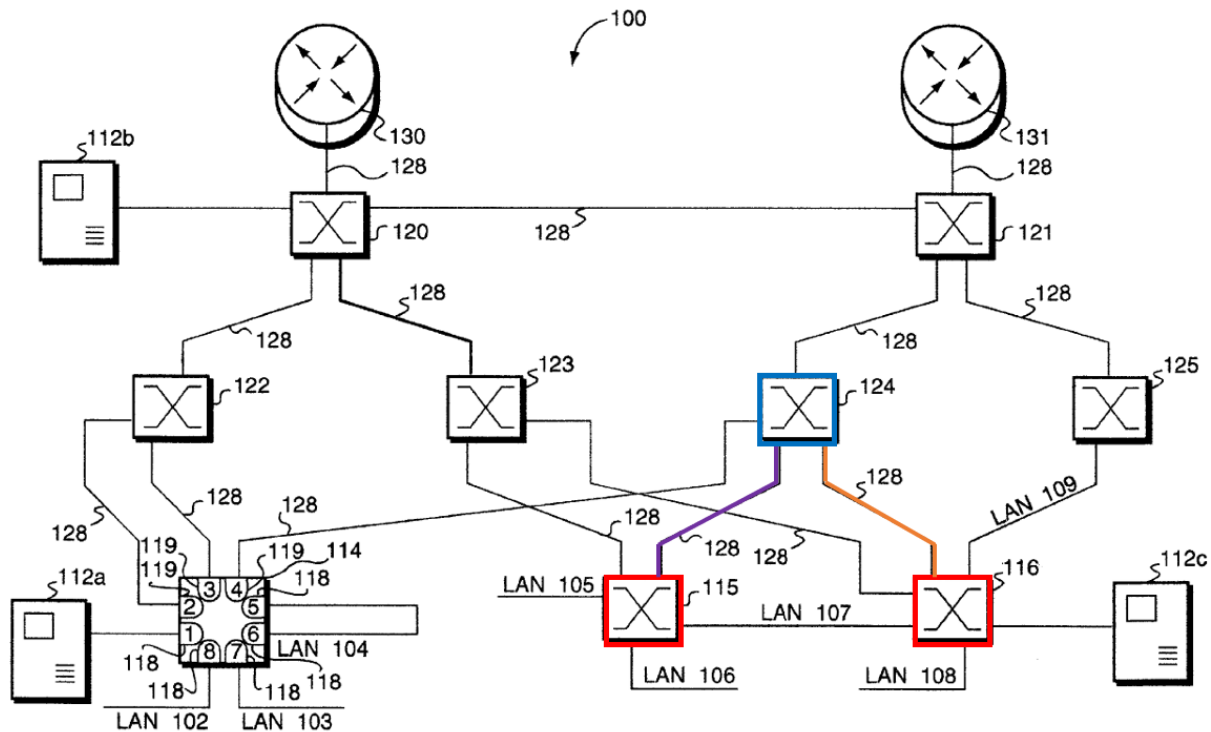


FIG. 1

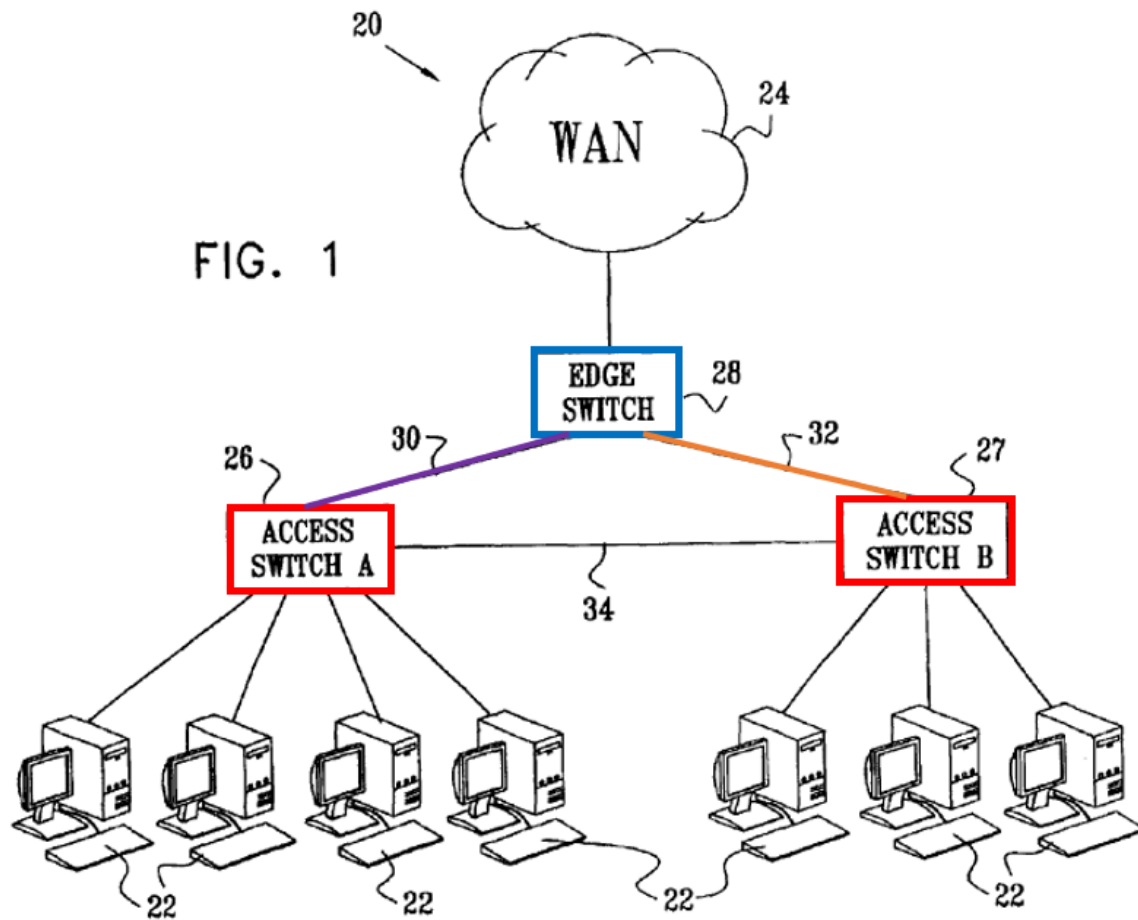
EX1005, Figure 1.

82. In Gai, **switches 115** and **116** have “ports that connect to LANs and/or to other network endpoints, such as user computer systems.” EX1001, 1:31-33; *see also*, EX1005, 7:4-7 (“The network 100 preferably comprises a plurality of

local area networks (LANs) 102-109 each of which is coupled to one or more access switches 114-116.”); EX1005, 6:10-12 (“Access switches are preferably coupled to entities (e.g., LANs, end stations, etc.)...”).

83. **Switch 124** satisfies the claimed **edge switch** because, as I explained in Section V.A, **edge switches** are “[c]ore switches that are directly connected to **access switches**.” EX1001, 1:36-38. *First*, as shown in Figure 1 of Gai, **switch 124** is directly connected to **access switches 115** and **116** through a first **link 128** and a second **link 128**, respectively. *See* EX1005, 7:9-14 (“**[A]ccess switches 114-116** are generally connected to the **backbone switches 120-125** through a series of links 128, such as point-to-point links....”). *Second*, **backbone switch 124** is a core switch because it “interconnect[s] the **access switch[es] [115 and 116]** into a Wide Area Network (WAN), by direct connection between the core switch and access switches and/or by connection to other core switches [such as switch 121].” EX1001, 1:33-36. And as Gai explains, interconnected LANs form a WAN. *See* EX1005, 1:31-34 (“[i]n many instances, several *LANs may be interconnected* by point-to-point links, microwave transceivers, satellite hook-ups, etc. *to form a wide area network (‘WAN’)*...”). Figure 1 of Gai shows several LANs (e.g., LANs 102-105) interconnected and, therefore, forming a WAN. Accordingly, Gai’s **backbone switch 124** corresponds to the claimed “**edge switch**” and “**first bridge**,” of claims 1 and 12, respectively.

84. *Third*, Gai discloses coupling the **edge switch** to the one or more **access switches** via at least **first** and **second** redundant links, respectively connected to first and second ports of the **edge switch**. As shown above in Figure 1 of Gai, **edge switch 124** is connected to **access switches 115** and **116**, similar to Figure 1 of the '986 Patent showing **edge switch 28** connected to **access switches 26** and **27**. Moreover, in Gai, **first redundant link 128** couples **switch 124** to access **switch 115** so as to communicate with **switch 115** and **second redundant link 128** couples **switch 124** to **access switch 116**, just like redundant links **30** and **32** of the '986 Patent, shown below.



EX1001, Figure 1.

85. Figure 1 of Gai only shows switch 114 as having ports (ports 118 and 119). Although Figure 1 of Gai does not explicitly show the other switches having ports, a POSITA would have understood, for example, that **switch 124** also has ports, because ports are used to connect the switch with the different links, e.g., **first redundant link 128** and **second redundant link 128**. In fact, a POSITA would have understood that **switch 124** can have the same eight ports shown in Figure 1 below, in connection with switch 114 (circled **red** below).

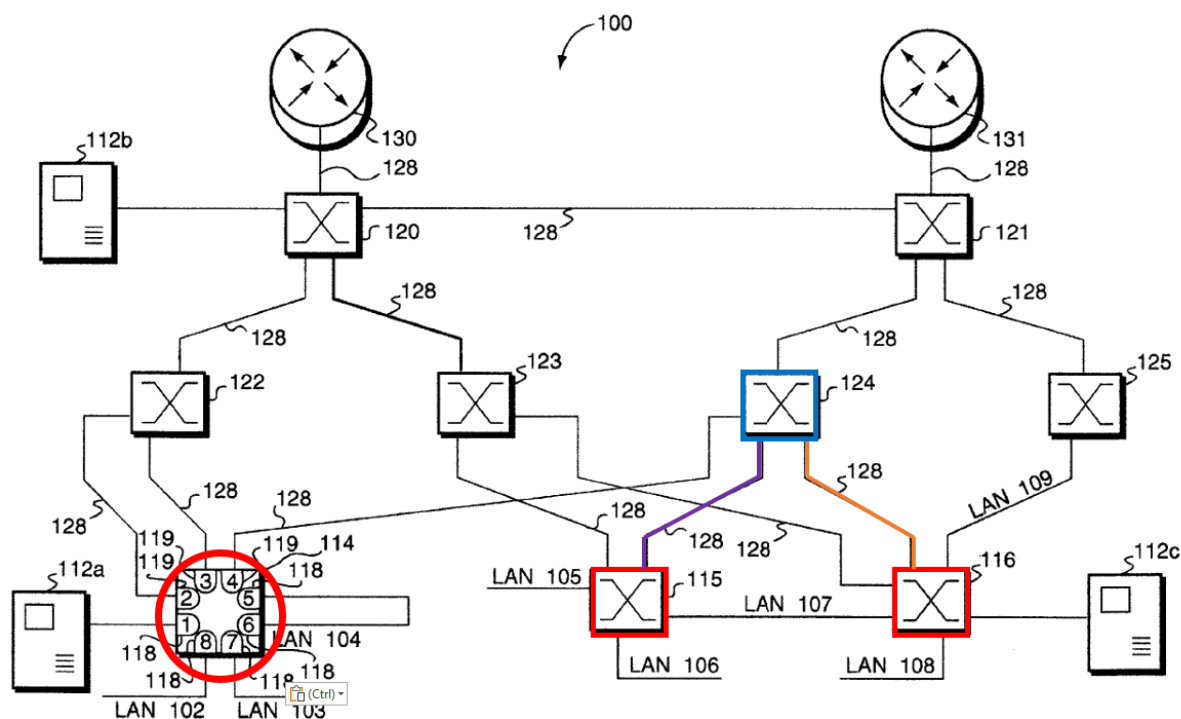


FIG. 1

EX1005, Figure 1.

86. Finally, and in connection with limitation [12a], **access switch/second bridge 115** is located downstream of the **edge switch 124** (*i.e.*, away from the root switch). As the '986 Patent explains, “upstream” corresponds to a direction toward the root bridge. *See* EX1001, 3:30-31 (“[T]oward the root (upstream)...”). In the network of Figure 1, **switch 121** is assumed to be the root switch. EX1005, 14:16-17 (“For example, assume that backbone switch 121 (FIG. 1) is the root...”). Accordingly, **second bridge 115** is located downstream of **first bridge 124**.

iii. [1b/12b] placing the first port in an **active state**, while placing the second port in a **blocking state**;

87. Gai discloses and/or renders obvious limitations [1b/12b].

88. *First*, as the '986 Patent admits, Gai discloses placing the first port in an **active state**, while placing the second port in a **blocking state**. 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*.”). In the example of Figure 1 below, the active forwarding port is identified using a **yellow dot** and the back-up port that is blocked is identified using a **red dot**.

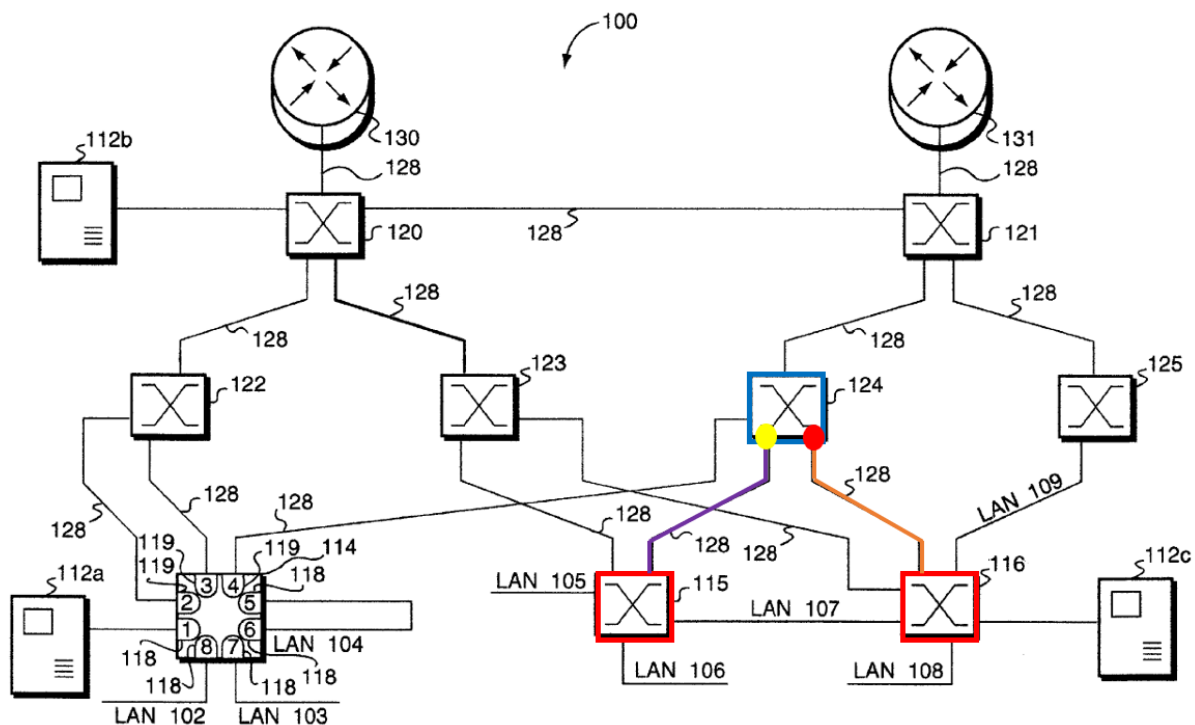


FIG. 1

EX1005, Figure 1.

89. Specifically, Gai discloses:

The network preferably includes a plurality of devices executing the spanning tree algorithm so as to elect a root and place the ports of the devices *in either a forwarding or blocked state*. In accordance with the method, one or more devices are configured and arranged so *that one trunk port is in the forwarding state and other trunk ports are in the blocked state*. Additionally, one or more of the blocked ports are designated as back-up ports.”

EX1005, 5:36-44; *see also*, EX1005, Abstract (“[O]ne or more devices are configured and arranged so that *one port*, providing connectivity to the root, *is in the forwarding state and the remaining ports*, providing connectivity to the root, *are in the blocked state*.”)

90. Because Gai’s network implements the spanning tree protocol, only the ports that are associated to links that are part of the tree (active links) will be forwarding, while the other ports would be in a blocked state. EX1005, 2:56-62 (“The root ports and designated switch *ports* are selected for inclusion in the spanning tree and are placed in a *forwarding state* so that data frames may be forwarded to and from these ports and thus onto the corresponding paths or links. *Ports not included within the spanning tree are placed in a blocked state*.”).

91. Accordingly, Gai teaches (or at a minimum suggests) that because the **first redundant link** is part of the STP, the first port (**yellow dot**) would be placed

in an active state while the second port (**red dot**) tied to the **second redundant link 128** would be placed in a blocking state.

- iv. **[1c/12c] conveying communication traffic over the first link while the first port is in the active state, thereby causing [each of the switches/the bridges] to build a respective database for use in forwarding the traffic;**

92. Gai discloses and/or renders obvious limitations [1c/12c].

93. *First*, Gai discloses “conveying communication traffic over the first link while the first port is in the active state.” For example, Gai discloses that the ports of the switches that are part of the spanning tree are placed in a forwarding state, e.g., active state, so that data frames, e.g., communication traffic, can be conveyed to and from these ports and over the corresponding links:

The root ports and designated *switch ports* are selected for inclusion in the spanning tree and *are placed in a forwarding state* so that data frames may be forwarded to and from these ports and thus *onto the corresponding paths or links*. Ports not included within the spanning tree are placed in a blocked state. When a port is in the blocked state, data frames will not be forwarded to or received from the port.

EX1005, 2:56-63; *see also*, EX1005, 11:21-24 (“Assuming an entry is found and a corresponding destination port (e.g., port number four) is identified, the message is switched out onto this port, *assuming this port is in the forwarding state*.”).

94. *Second*, Gai discloses “thereby causing each of the switches/the bridges to build a respective database for use in forwarding the traffic.” For

example, Gai discloses that switches learn which destination port to use for sending a message to a particular entity by inspecting the last message received from the particular entity and identifying which port the last message was received from and saving this information in a database. EX1005, 1:50-54 (“Switches typically learn which destination port to use in order to reach a particular entity by noting on which source port the last message originating from that entity was received. *This information is then stored by each switch in a block of memory referred to as a filtering database.*”). For example, Figure 2, which is reproduced below, shows the filtering database for switch 214.

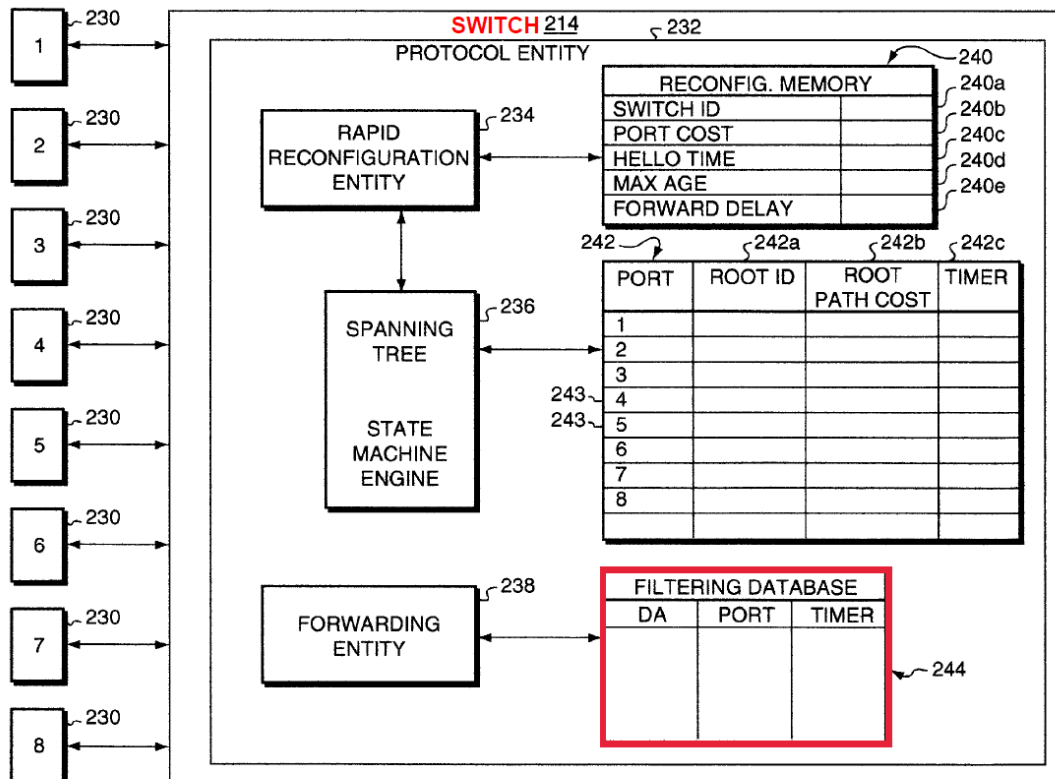


FIG. 2

EX1005, Figure 2.

95. Gai also discloses that when a message is received at the switch and is addressed to a particular entity, the switch uses the port information stored in the database to direct the message to the appropriate output port. EX1005, 1:55-62 (“Thereafter, when a message addressed to a given entity is received on a source port, *the switch looks up the entity in its filtering database and identifies the appropriate destination port to utilize in order to reach that entity.*”); see also, EX1005, 11:21-24 (“Assuming an entry is found and a corresponding destination port (e.g., port number four) is identified, the message is switched out onto this

port, assuming this port is in the forwarding state.”); *see also* EX1005, 8:32-39. If there is no entry in the filtering database, a new entry is formed for the received message. EX1005, 11:16-33 (“If no entry is found for the source address, a new entry is formed and the source port (e.g., port number eight) on which the message was received is entered as the destination port in the corresponding port column.”), Fig. 2, 8:32-39.

v. [1d/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

96. Gai discloses and/or renders obvious limitations [1d/12d].

97. *First*, the '986 Patent admits that Gai discloses this limitation.

Specifically, when describing Gai, the '986 Patent explains that in Gai, “[u]pon 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.” EX1001, 3:26-29.

98. Indeed, Gai discloses that “[*u*]pon *detection of a failure* at the active forwarding port [indicated in the Figure below with a **purple X**], the state of one of the back-up ports [in **red dot**] *immediately transitions from blocked to forwarding*, thereby becoming the new active port for the device.” EX1005, 5:44-47; *see also*, EX1005, 5:20-24 (“It is a further object of the present invention to provide a method and apparatus for defining a series of back-up ports which *may*

immediately begin forwarding data messages following a failure at an active port.”); 11:36-41; Abstract.

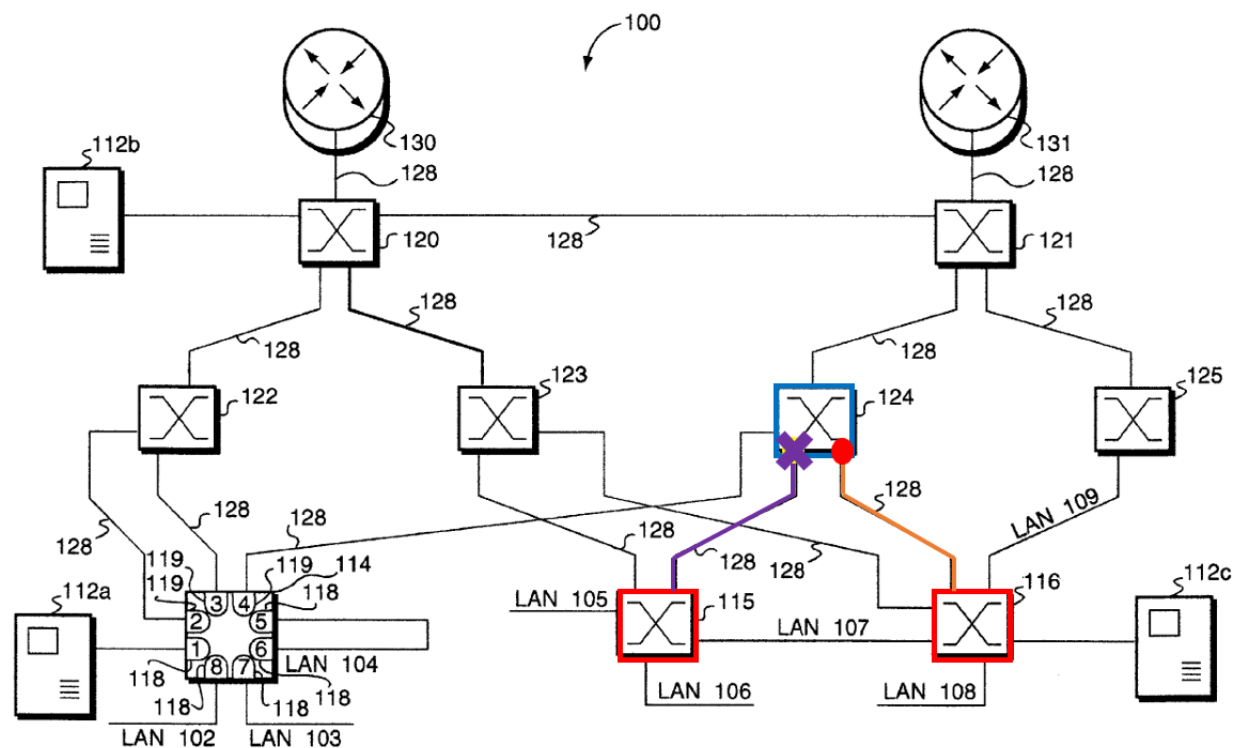


FIG. 1

EX1005, Figure 1.

99. A failure in a port is a failure associated with the link connected to the port, because the port is one endpoint of the link it is connected to, and a failure on the port would result in a failure to the associated link. For example, Gai, in connection with Figure 3D below, shows “a flow diagram of a rapid reconfiguration process 340 following a *link failure*... In response to the detection of a *failure at port number three* (the root port), indicated at block 342, rapid

reconfiguration entity 234 at switch 214 *selects a backup port* to become the new root port, as shown at box 344.” EX1005, 12:13-19; *see also* EX1005, 12:32-35 (“Rapid reconfiguration entity 234, at block 346, then directs the spanning tree state machine engine 236 to immediately transition the selected back-up port (e.g., port number four) to the forwarding state.”).

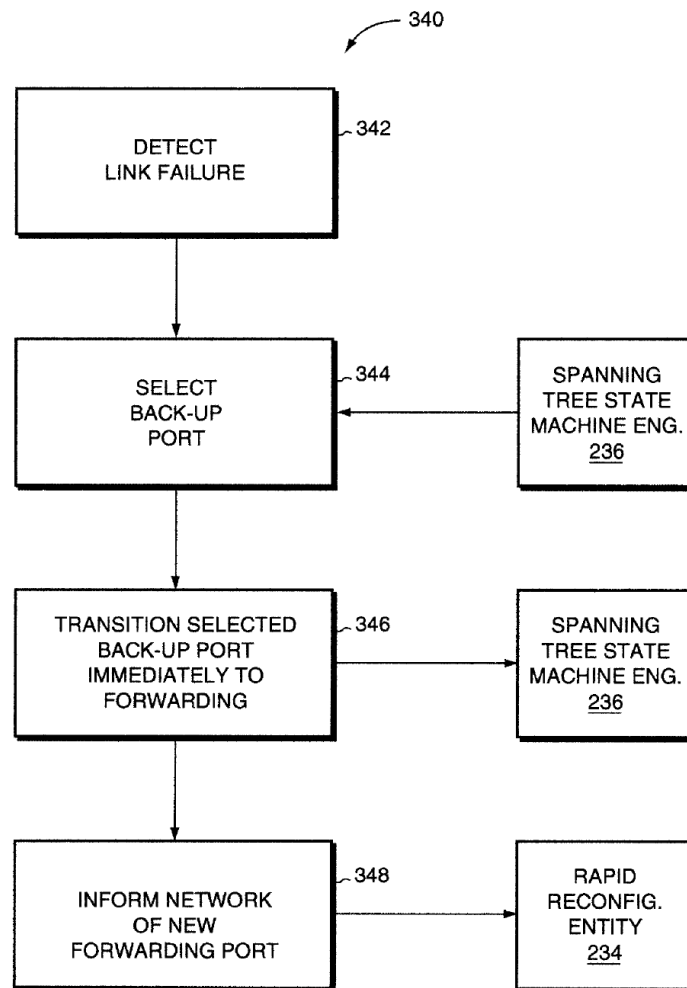


FIG. 3D

EX1005, Figure 3D.

- vi. [1e/12e] sending dummy traffic from the [edge switch/first bridge] over the second link [downstream] to [the one or more access switches/the second bridge], so as to cause each of [the one or more access switches/the second bridge] to modify its respective database responsive to the second port being in the active state.

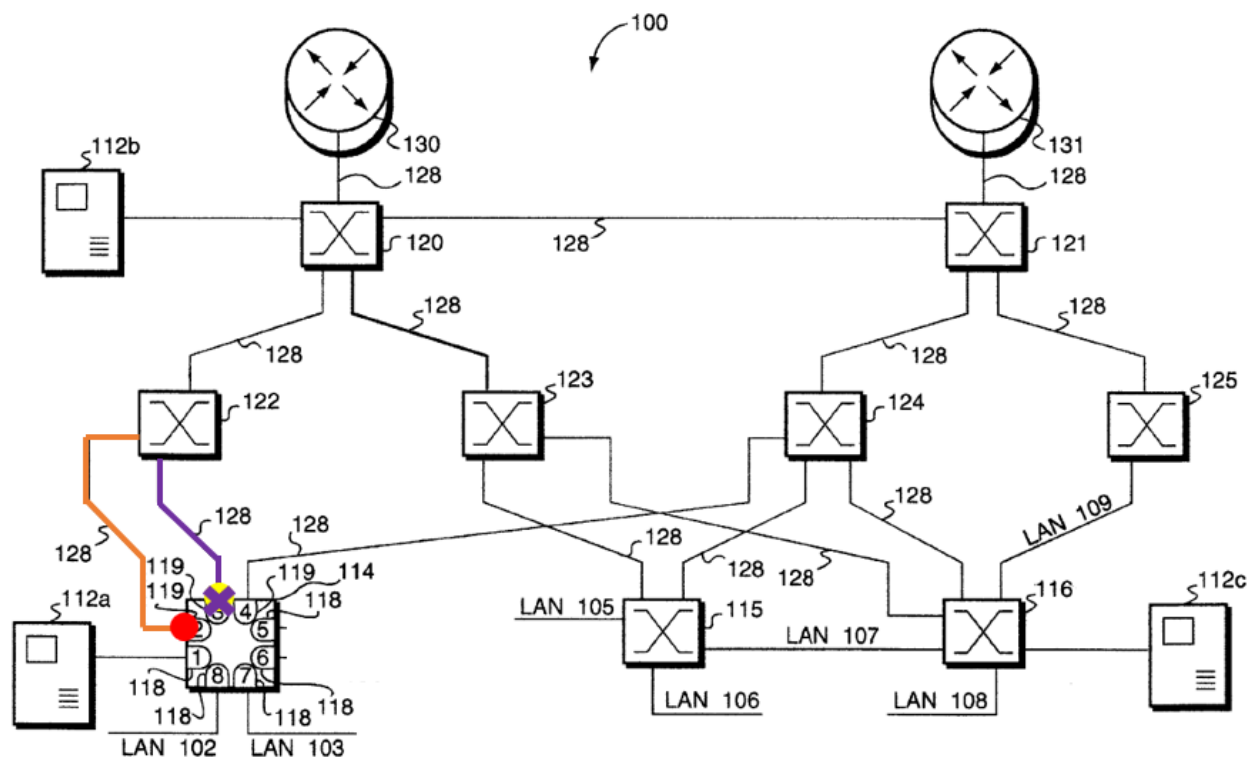
100. Gai discloses and/or renders obvious limitations [1e/12e]. As an initial matter, the '986 Patent admits that Gai discloses sending dummy traffic from a switch through a new active port, such that other switches in the network learn of the new active port. EX1001, 3:29-35 ("The switch [in Gai] then transmits dummy multicast messages toward the root (upstream) through the newly-active trunk port, each containing the source address of a network entity that is coupled to the switch itself or is downstream of the switch. Upon receiving these messages, other devices in the network learn to use the new active port."). This admission relates to an upstream transmission of dummy messages. However, as more fully explained below, Gai teaches (or at a minimum renders obvious) downstream transmission of dummy messages as well.

101. *First*, Gai discloses sending dummy traffic from the edge switch/first bridge over the second link downstream to the one or more access switches/second bridge. As shown below, consistent with the '986 Patent admission, Gai discloses one specific example of a failure associated with port 3

(marked with **purple X**) of switch 114 that transmits packets towards the root (i.e., upstream) through the newly activated port 2 (**red dot**):

With reference to switch 114, *for example*, assume that port number three, which is coupled to backbone switch 122, *is the root port* for switch 114 and thus in the forwarding state. ... *If the link 128 coupled to the port number three at switch 114 fails*, either one of these two other ports (i.e., port numbers two or four) will immediately transition to the forwarding state and begin receiving and sending messages.

EX1005, 11:41-52; *see also*, EX1005, 12:62-66 (“Switch 114, via rapid reconfiguration entity 234, preferably informs the other devices of its new forwarding port, as indicated at block 348.”).

**FIG. 1**

EX1005, Figure 1.

102. In this example, switch 114 happens to be at the lowermost level of the spanning tree with regard to switches, e.g., there are no other switches downstream of switch 114. The particular port that failed (port 3) also happens to be a port transmitting packets directed towards the root switch 121. EX1005, 14:16-17 (“For example, assume that backbone switch 121 (FIG. 1) is the root...”).

103. However, a POSITA would have understood that a failure can happen at a port or a link anywhere in the network. For example, a failure can occur at a port other than the “root port,” i.e., a port that is used to transmit packets upstream.

Such a port is called a “designated” port, i.e., a port that is used to transmit packets downstream. *See e.g.*, EX1012, 2:56-59 (“A ... *port may fail for various reasons*, including a software defect, an equipment failure in the call control infrastructure or a failure in the transmission link.”); EX1013, 16:59-60 (“[The] bridge status memory file is updated *whenever any ports in any of said plurality of bridge servers fails*.”); EX1006, 17:61-63 (“[A] root port for communication with a root router and [a] *designated port to communicate with at least one router in a lower leaf of a spanning tree*.”); EX1011, 5:6-9 (“[I]mprovements to the spanning tree algorithm can be used to facilitate *rapid detection of a designated port* or designated bridge *failure*.”); EX1014, ¶63 (“*If any of the ports fails*, the SM in the sub-switch 1104 reconfigures the mappings to reroute the packets correctly.”).

104. For example, as shown below, if a failure happened at the port of **switch 124** that is connected to **first redundant link 128**, then a back-up port (e.g., the port indicated with a red dot below) would transition from a “blocked” state to a “forwarding” state, therefore coupling **switch 124** to **access switch 116** through **second redundant link 128**. *See e.g.*, EX1005, 44-47 (“Upon detection of a failure at the active forwarding port, the state of one of the back-up ports immediately transitions from blocked to forwarding, thereby becoming the new active port for the device.”).

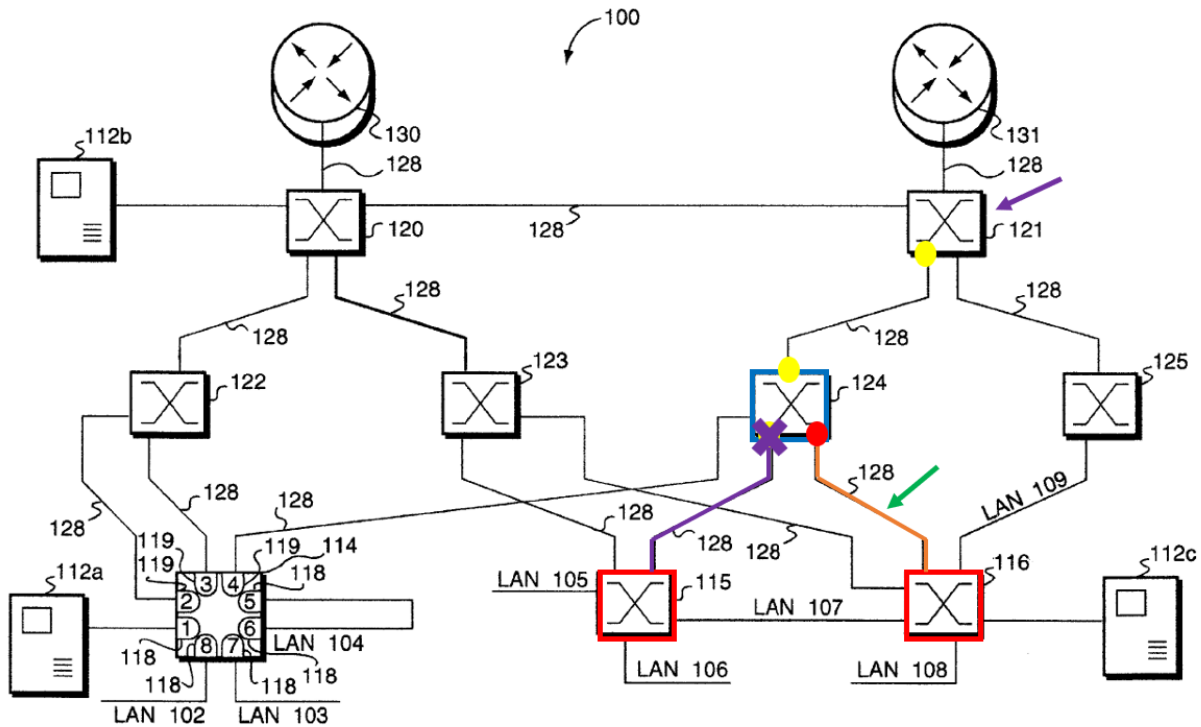


FIG. 1

EX1005, Figure 1.

105. Accordingly, when *edge switch 124* activates the new forwarding port, *edge switch 124* would start transmitting dummy messages, e.g., a dummy message to *access switches 116 and 115* (in that order) through the port indicated by the red dot, i.e., through the new forwarding port, over *link 128*. Specifically, Gai discloses “[u]pon transition to the *new forwarding port*, the device begins transmitting ‘dummy’ multicast messages *through the new port*.” EX1005, 5:54-56.

106. Figure 4 of Gai, which is reproduced below, “is a highly schematic illustration of a *dummy multicast message 400* preferably utilized by rapid

reconfiguration entity 234.” EX1005, 12:66-13:1. “Dummy multicast message 400 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. ... Rapid reconfiguration entity 234 preferably loads DA field 410 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, 13:1-13.

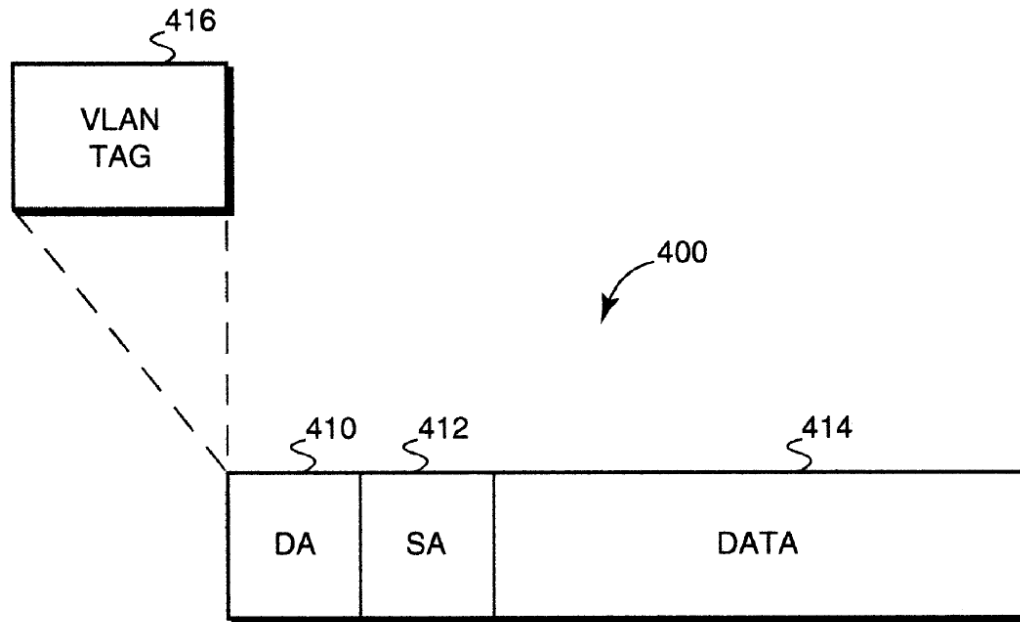


FIG. 4

EX1005, Figure 4.

107. That Gai discloses downstream transmission of dummy messages is also evidenced by the claims of U.S. Patent No. 7,848,264 (EX1006). EX1006 is a

continuation application of Gai (EX1005) and shares identical Specifications with Gai. The claims of EX1006 include transmitting dummy messages from a newly active designated port, i.e., a port transmitting packets downstream, to inform devices logically located downstream in the spanning tree about the new designated port. EX1006, 18:8-11 (“[T]ransmitting, by the back-up *designated port*, one or more messages, *the messages to tell devices downstream in the spanning tree that the router has a new designated port.*”); see also, EX1006, claim 1 (17:56-18:11).

108. Also, as Gai explains, the “object of the [Gai] invention to provide a method and apparatus for reducing the time necessary to reconfigure the network following a change...”. EX1005, 5:16-19. Therefore, sending dummy traffic from a new active port downstream so “that other devices within the network [i.e., the downstream devices] learn to utilize the new forwarding port, rather than the failed port, when directing messages to these entities,” (EX1005, 5:65-6:1) would have been entirely consistent with reducing the time necessary to reconfigure the network.

109. *Second*, Gai discloses and/or renders obvious “so as to cause each of the one or more *access switches/second bridge* to modify its respective database responsive to the second port being in the active state.” For example, Gai discloses:

Upon receipt [of the dummy messages,] the other devices examine the contents of these messages and note the port on which they were received, which may differ from the port on which messages from these entities were previously received (i.e., before the failure and subsequent replacement of the device's active port). It is through this process that other devices within the network learn to utilize the new forwarding port, rather than the failed port, when directing messages to these entities.”

EX1005, 5:54-6:5.

Nonetheless, the transition at access switch 114 from initial root port number three to back-up root port number four may cause entities to appear to “move” relative to other devices. Thus, the identity of the new root port (e.g., port number four) must be propagated to the other devices, such as access switches 115-116 and backbone switches 120-125 to prevent messages from being lost or misdirected.

EX1005, 12:56-62.

110. Gai discloses that the switch that receives the dummy message from the new port enters this information in its filtering database such that this new port information is used when the switch needs to send a message to be received from the newly activated port. For example, in connection with the example of a failure at port 3 of switch 114, Gai discloses:

Since the multicast message 400 having the source address of server 112a was received by backbone switch 124 at *a new source port, this*

new location information is entered by backbone switch 124 into its filtering database, replacing the previous information that was stored therein. Thereafter, if backbone switch 124 receives a message intended for server 112a, *it will use this new destination port* and the message will be received at port number four of switch 114, which is now capable of receiving and forwarding messages. ... Backbone switch 124 distributes the multicast message 400 through all of its forwarding ports *so that other devices, as necessary, may learn of the new forwarding port....*

EX1005, 13:34-49; *see also*, EX1005, 13:34-49, 12:66-13:33.

As set forth above, upon receipt of these multicast messages 400, the other devices within network 100 update their corresponding filtering databases with the new destination ports, as necessary.

EX1005, 13:67-14:3.

111. Accordingly, a POSITA would have understood that when *access switch 116* receives a dummy message from the newly activated port it would update its filtering database with the new port information and also would distribute the dummy message to *access switch 115*, such that switch 115 also learns of the new forwarding port, e.g., by updating its corresponding database.

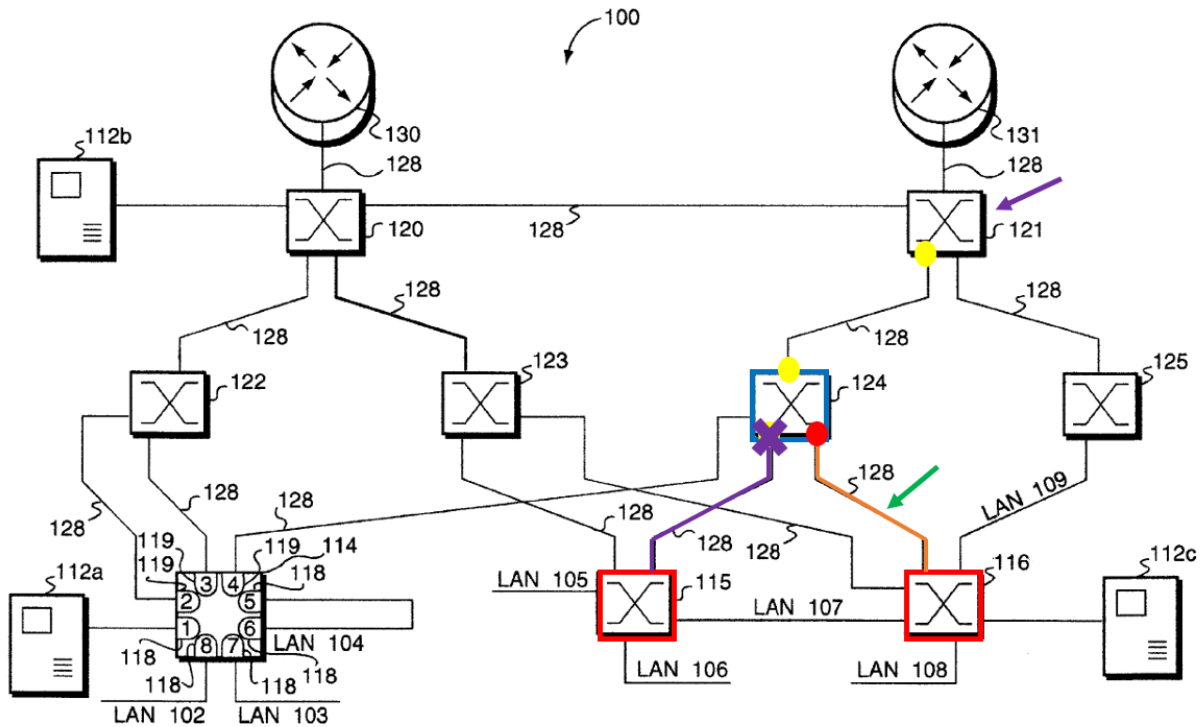


FIG. 1

EX1005, Figure 1.

112. Indeed, Figure 2, which is a “block diagram of a switch 214 ... which corresponds to [an] access switch” shows filtering database 244. A POSITA would have understood that when *access switches 115* and *116* would forward messages intended for entities located upstream, they would use the new port information for upstream messages, such that the messages reach their intended recipients.

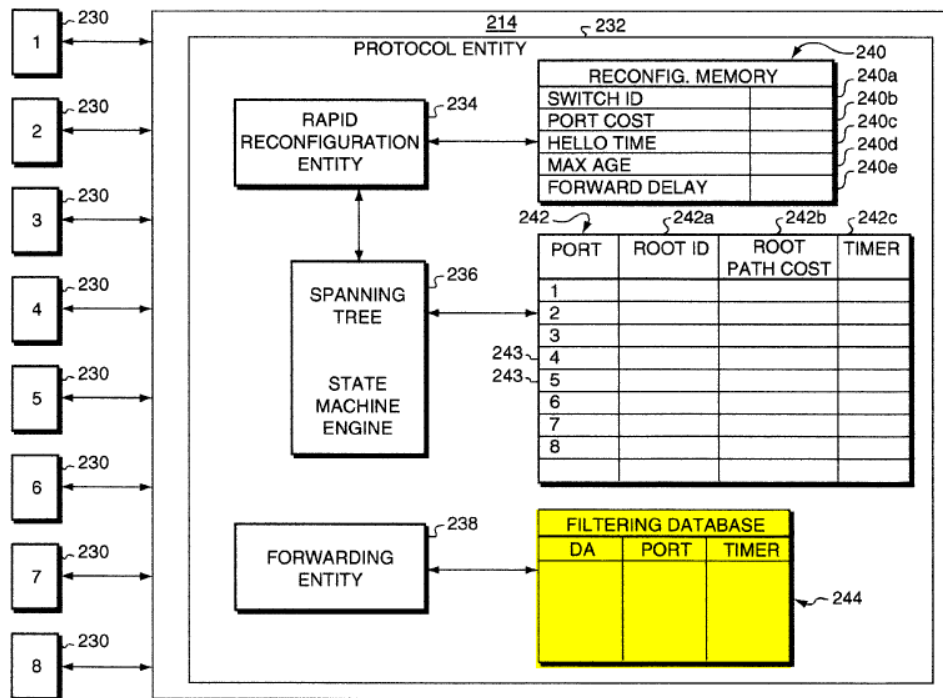


FIG. 2

EX1005, Figure 2.

2. Independent Claim 13

i. [13pre] A **network switching device**, comprising:

113. I understand that the preamble may not be limiting. Nevertheless, Gai discloses and/or renders obvious the preamble. For example, Figure 1 of Gai discloses various examples of a **network switching device**, such as **switch 124**.

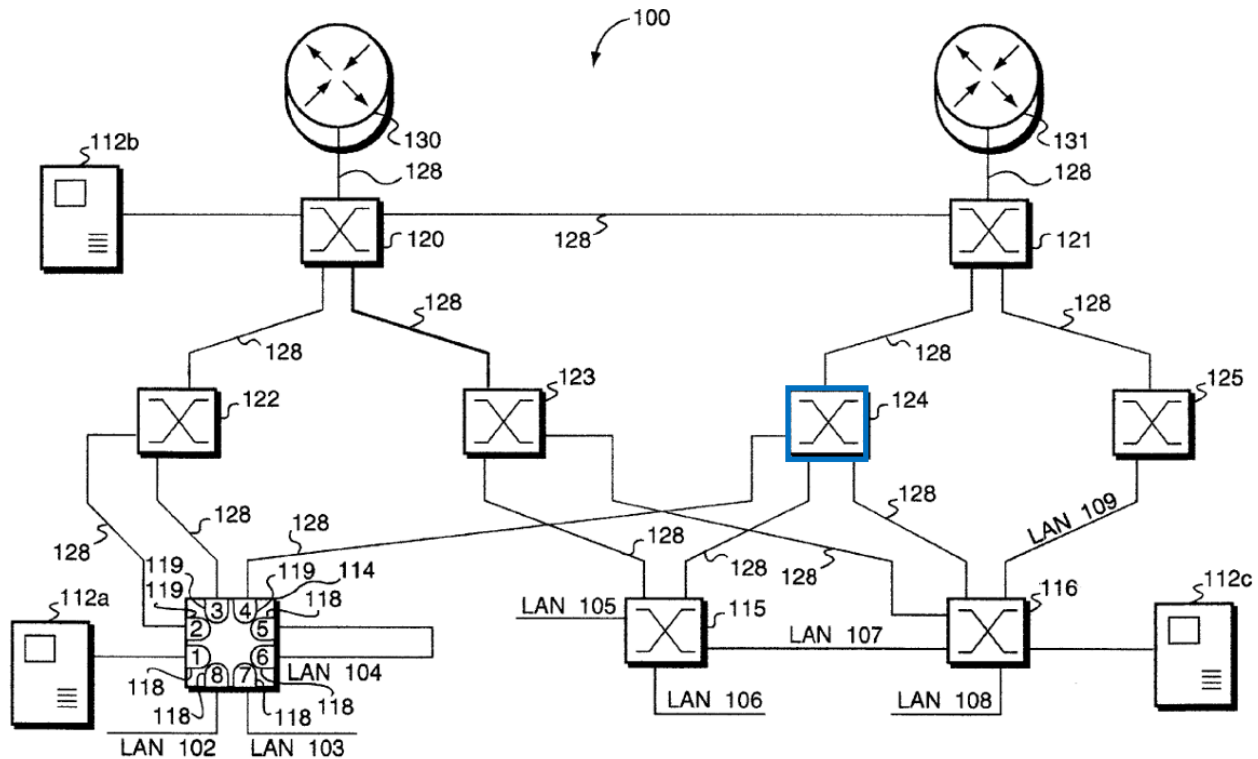


FIG. 1

EX1005, Figure 1.

- ii. [13a] 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;

114. Gai discloses and/or renders obvious this limitation. For example, in the exemplary network of Figure 1 of Gai, **switch 124** has a network port (identified with the **yellow dot**) coupled to core switch 121 over link 128 identified with a **green** arrow, and meets the claimed “trunk link.” I note that the term “trunk link” does not appear in the Specification of the ’986 Patent. *See also*, EX1005, 1:43-46 (“Ports used to couple switches to each other are generally referred to as a

trunk ports, whereas ports used to couple a switch to LANs or end stations are generally referred to as local ports.”).

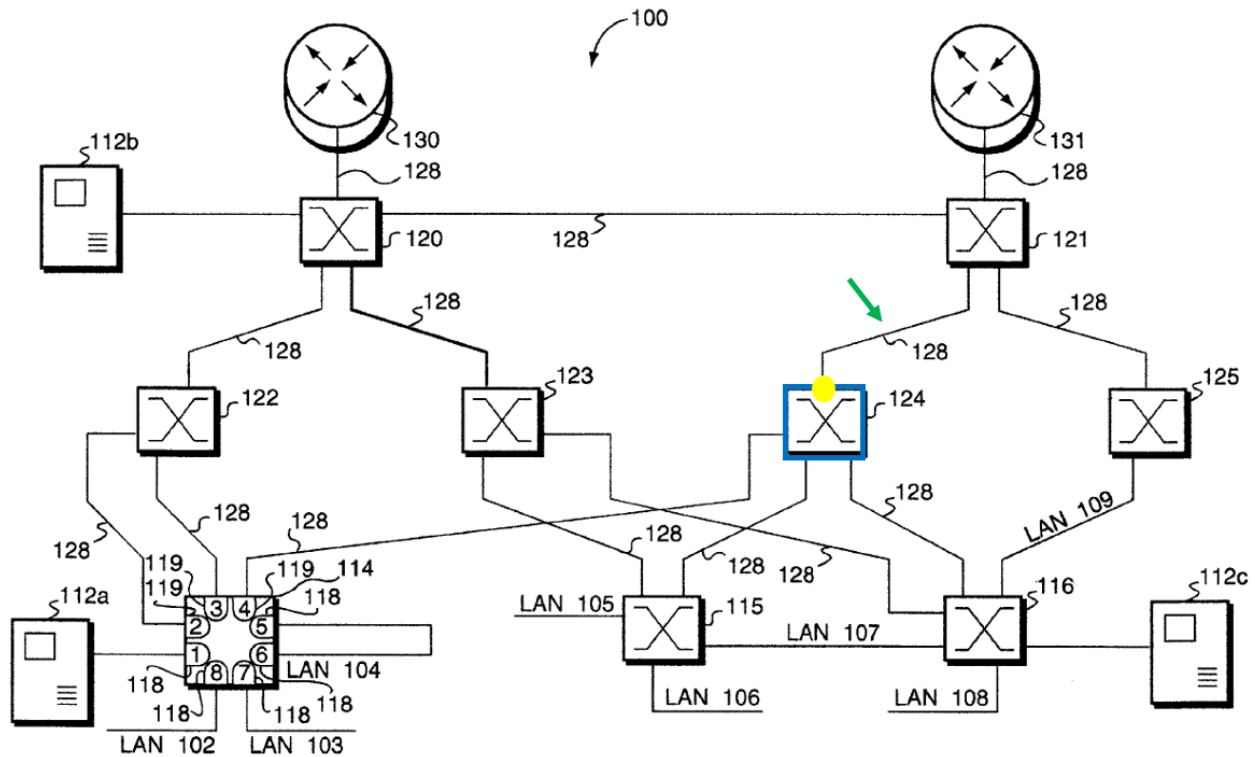


FIG. 1

EX1005, Figure 1.

115. Moreover, a POSITA would have also understood that the network port of **switch 124** exchanges communication traffic with core switch 121 because it is directly connected—through the trunk link—to the core switch 121. For example, all frames received by switch 124 with a destination address for router 131 would be forwarded to switch 121 through the network port identified with the **yellow dot**.

- iii. **[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**

116. Gai discloses and/or renders obvious this limitation for the reasons I discussed in Section VIII.A.1.ii. Additionally, the '986 Patent explains that access ports are the ports that are connected to the downstream links to the access switches. *See* EX1001, 4:1-2 (“[A]ccess ports (i.e., the ports that are connected to the downstream links to the access switches)...”). Gai’s “first” and “second” ports discussed in connection with limitations [1a/12a] are “access” ports because they are connected to the downstream links to the access switches, such as access switch 115 and 116.

117. A POSITA would have also understood that the access switches in Gai serve network users. For example, a POSITA would have understood that users would be connected to the LANs shown in Figure 1 of Gai, such as LANs 106, 107, and 108. *See also*, EX1005, 5:2-7.

- iv. **[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,**

118. Gai discloses and/or renders obvious limitation [13c].

119. *First*, Gai discloses a “protocol processor.” For example, Gai’s network switching device “includes *a protocol entity 232*. The protocol entity 232

preferably includes a rapid reconfiguration entity 234, at least one spanning tree state machine engine 236 and a forwarding entity 238.” EX1005, 8:12-15. Gai’s protocol entity 232 is a protocol processor because it includes “processing elements” that implement the STP [spanning tree **protocol**]. EX1005, 8:46-50 (“[E]ntities 234 and 238 and state machine engine 236 preferably comprise *programmable processing elements* containing software programs pertaining to the methods described herein and executable by the processing elements.”); 7:53-61 (“[N]etwork 100 includes redundant communication paths interconnecting the access and backbone switches 114-116 and 120-125. ... [T]hese switches execute the spanning tree algorithm.”). Indeed, Gai discloses that its devices implement the *spanning tree algorithm and protocol*. Specifically, Gai discloses “a plurality of devices executing the *spanning tree algorithm*” (EX1005, 5:37-38) where “switches exchange special messages called bridge protocol data unit (BPDU) frames that allow them to calculate a spanning tree or active topology...”. EX1005, 2:33-35.

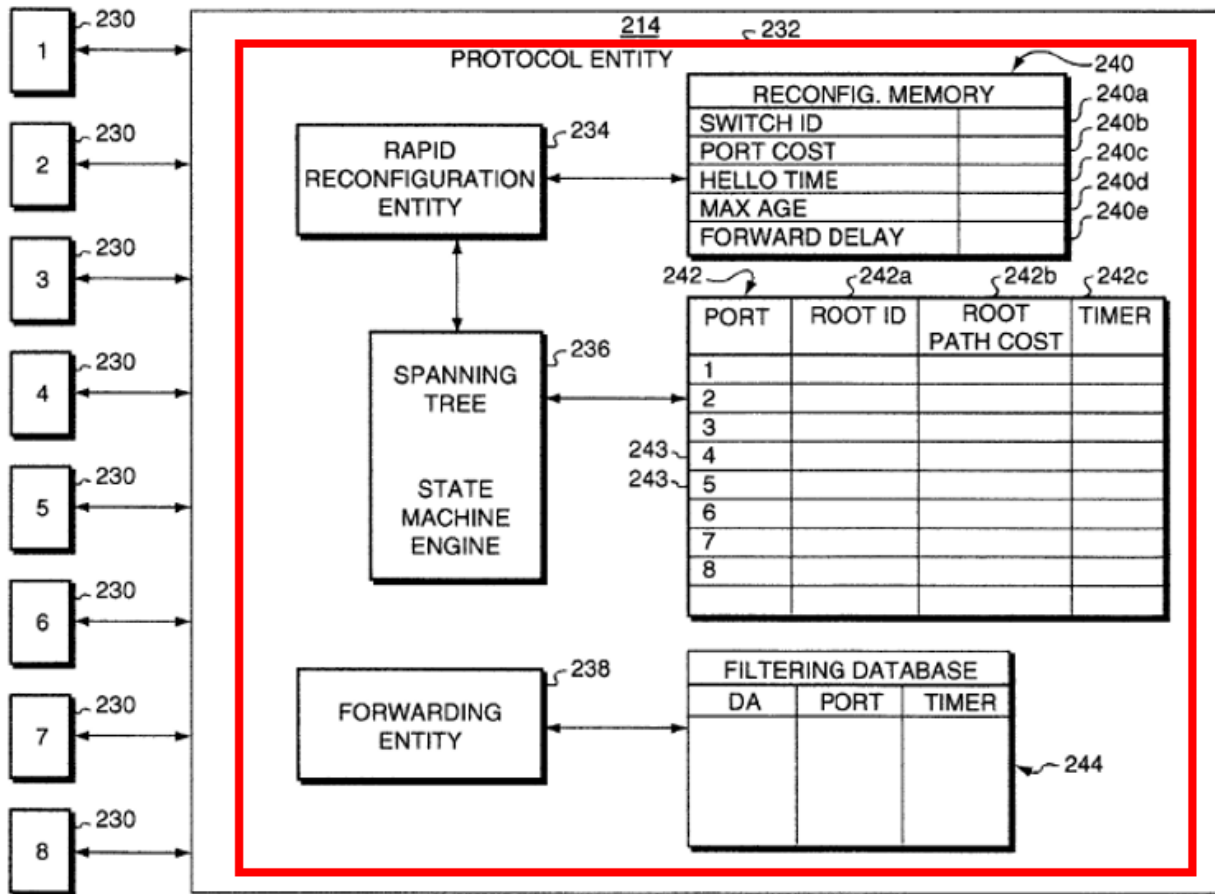


FIG. 2

EX1005, Figure 2.

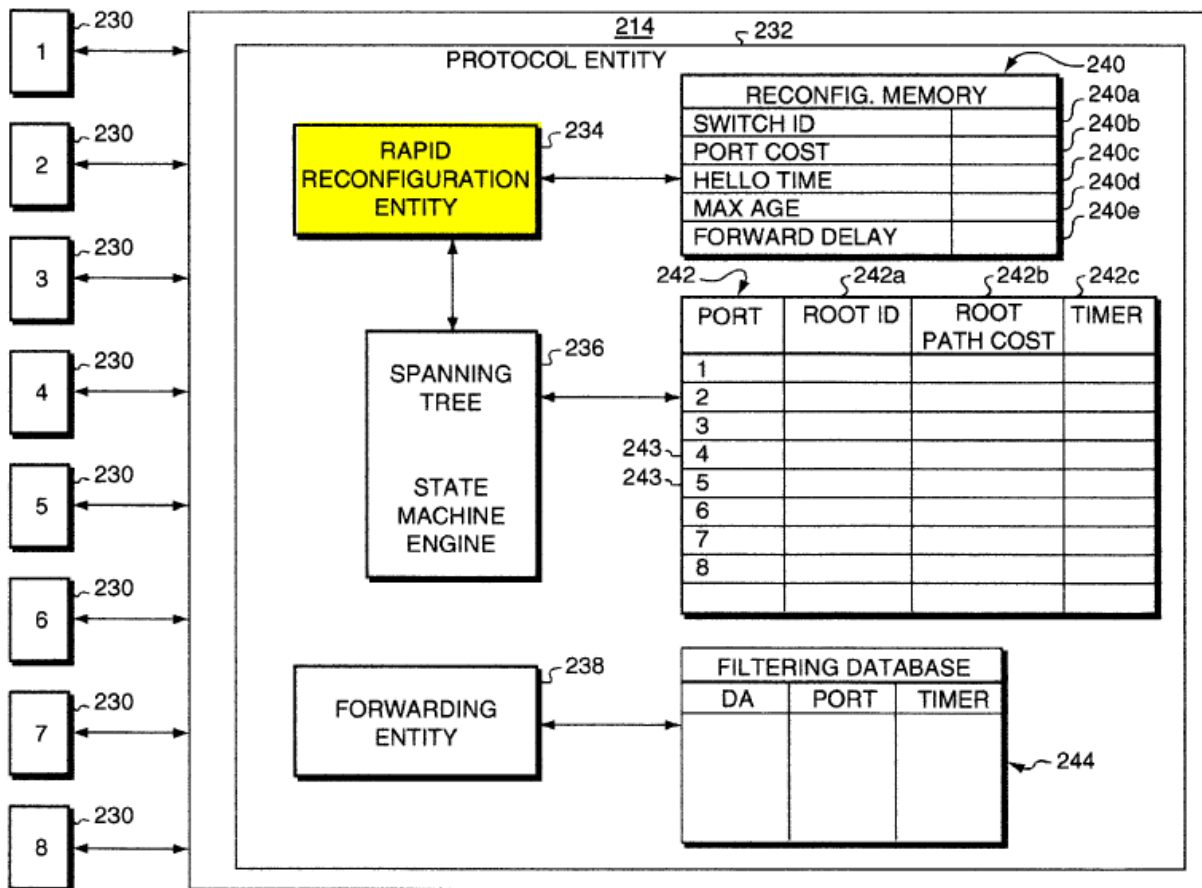
120. *Second*, Gai discloses and/or renders obvious “adapted to place the first access port in an active state, while placing the second access port in a blocking state,” for the reasons I discussed in Section VIII.A.1.iii.

- v. **[13d] 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,**

121. Gai discloses and/or renders obvious this limitation for the reasons I discussed in Section VIII.A.1.iv.

- vi. **[13e] 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**

122. Gai discloses and/or renders obvious this limitation for the reasons I discussed in Section VIII.A.1.v. In addition, Gai discloses that it is the protocol processor that is being further adapted to place the second access port in the active state and the first port in the blocking state responsive to a failure associated with the first link. Specifically, Gai discloses “[r]apid reconfiguration entity 234 [*within protocol entity 232*] then directs the spanning tree state machine engine 236 to immediately transition the selected back-up port ... to the forwarding state.” EX1005, 12:32-35.



EX1005, Figure 2.

- vii. [13f] [the protocol processor being further adapted, responsive to a failure associated with the first link] 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.

123. Gai discloses and/or renders obvious this limitation for the reasons I discussed in Section VIII.A.1.vi. In addition, Gai discloses that “switch 114, via rapid reconfiguration entity 234 [within protocol entity 232], preferably informs

the other devices of its new forwarding port by transmitting dummy multicast packets through the new port...” EX1005, 12:62-65.

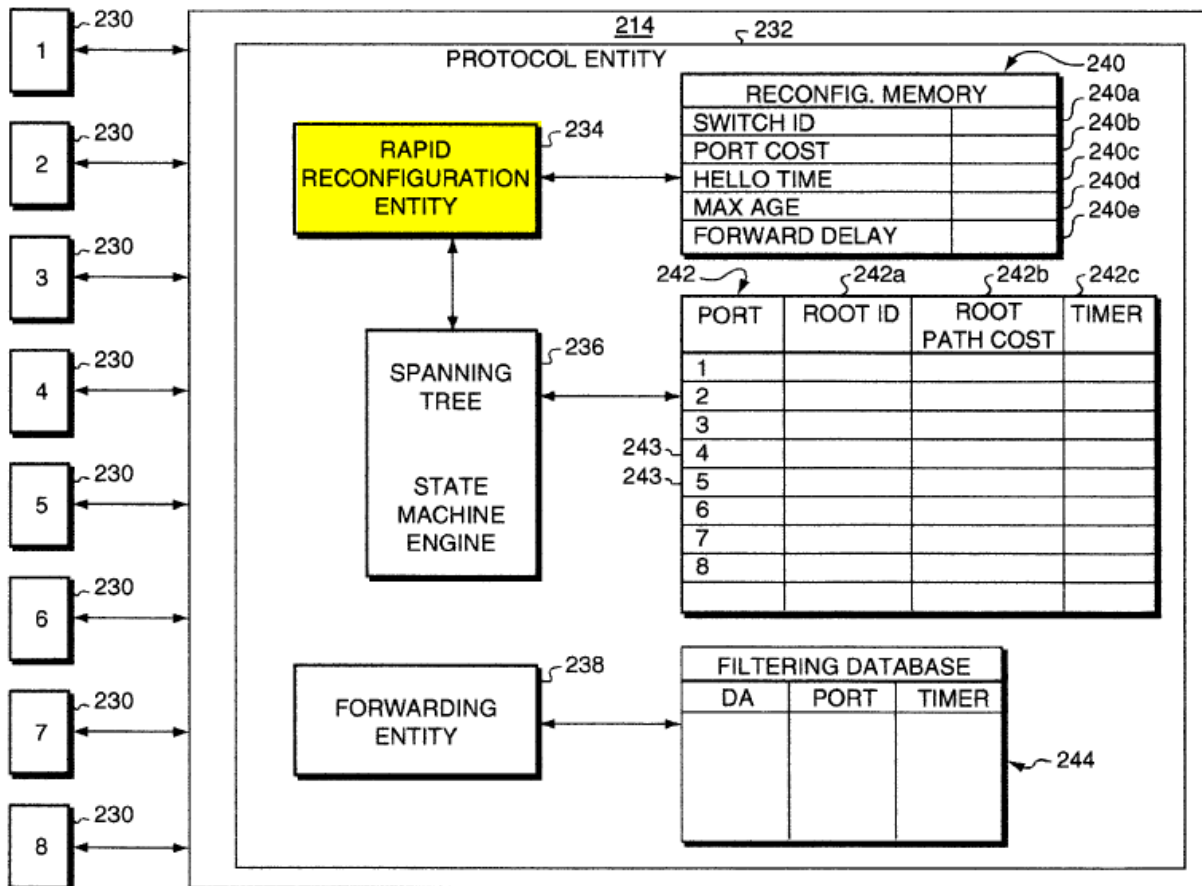


FIG. 2

EX1005, Figure 2.

3. Independent Claim 24

- i. [24pre] A bridge device for use in a communication network having a spanning tree root, the device comprising:

124. Gai discloses and/or renders obvious this limitation for the reasons I discussed in Sections VIII.A.2.i, VIII.A.1.i, VIII.A.1.ii, and VIII.A.1.vi.

- ii. [24a] a designated port, adapted to be coupled over an upstream link in the network so as to communicate with the root;

125. Gai discloses and/or renders obvious this limitation for the reasons I discussed in Section VIII.A.2.ii. Additionally, *switch 124* has a designated port (identified by the yellow dot) that connects it to the root bridge 121.

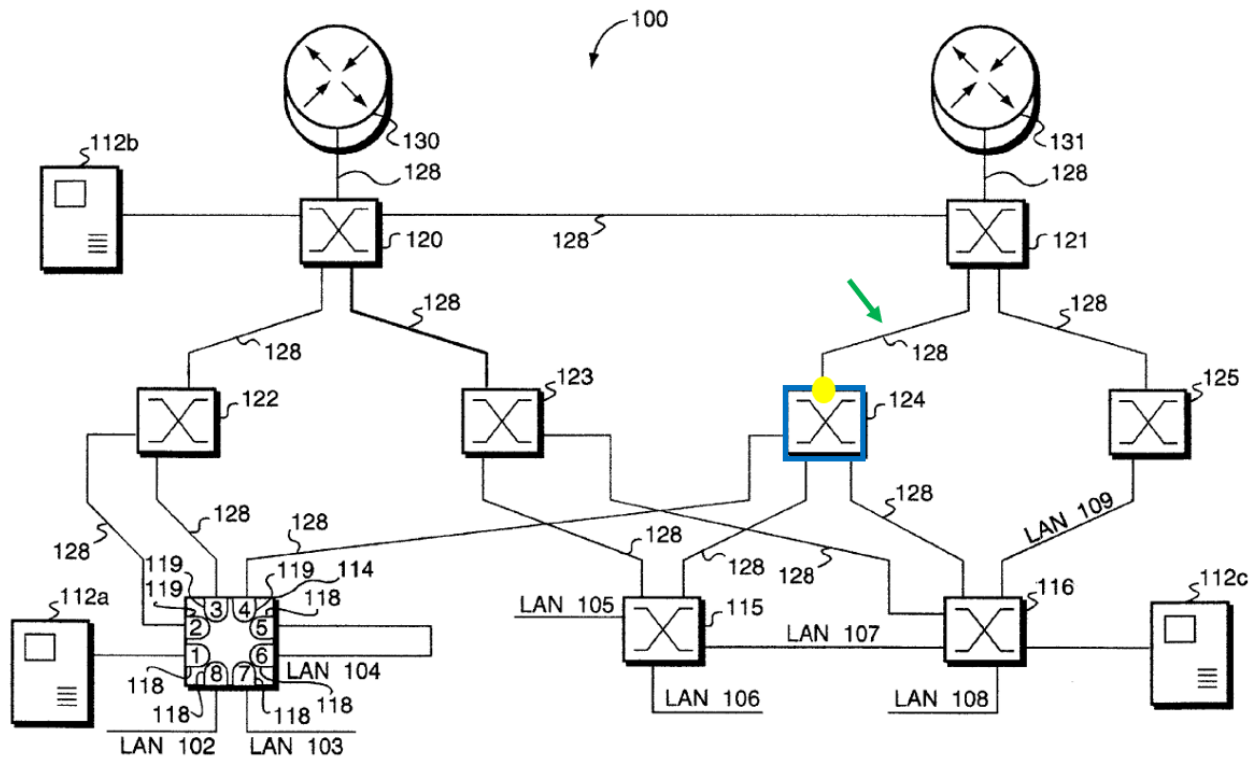


FIG. 1

EX1005, Figure 1.

- iii. **[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**

126. Gai discloses and/or renders obvious this limitation for the reasons I discussed in Sections VIII.A.1.ii and VIII.A.1.vi.

- iv. **[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,**

127. Gai discloses and/or renders obvious this limitation for the reasons I discussed in Sections VIII.A.2.iv and VIII.A.1.vi.

- v. **[24d] 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,**

128. Gai discloses and/or renders obvious this limitation for the reasons I discussed in Sections VIII.A.2.v and VIII.A.1.vi.

- vi. **[24e] 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.**

129. Gai discloses and/or renders obvious this limitation for the reasons I discussed in Sections VIII.A.2.vi, VIII.A.2.vii, and VIII.A.1.vi.

4. Dependent Claims 2 and 14

i. [2pre/14pre] A method/device according to claim 1/13,

130. Gai renders obvious claims 1 and 13. *See* Sections VIII.A.1 and VIII.A.2.

ii. [2a/14a] wherein [placing the second port in the blocking state comprises configuring]/[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.

131. Gai discloses and/or renders obvious this limitation. For example, Gai implements the spanning tree algorithm, which severs redundant paths and, therefore, ensures that there are no loops in the network. EX1005, 5:36-39; 2:30-41 (“To avoid the formation of loops, devices, such as switches or bridges, execute a Spanning tree algorithm...”). Accordingly, a POSITA would have understood that no loop is created in the network among any of the network switches, including the edge switch and the one or more access switches.

5. Dependent Claim 4

i. [4pre] A method according to claim 1, and comprising,

132. Gai renders obvious claim 1. *See* Section VIII.A.1.

- ii. **[4a] 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.**

133. Gai discloses and/or renders obvious this limitation. For example, Gai discloses that the switches in the network, e.g., including the edge switch, discard old information in their databases after a port transitions to an active state. EX1005, 4:30-34. (“As ports transition between the blocked and forwarding states, entities may appear to move from one port to another. To prevent switches from distributing messages based upon incorrect information, *switches* quickly age-out and *discard the ‘old’ information in their filtering databases.*”).

134. Gai’s protocol entity includes a “filtering database 244” that stores address information corresponding to the network entities. EX1005, 8:32-34 (“The forwarding entity 238 is coupled to a filtering database 244 configured to store address information corresponding to the entities of network 100.”). Specifically, “filtering database 244” includes columns with information such as, a destination port cell, and rows which correspond to particular network entities. EX1005, 8:34-39 (“Specifically, filtering database 244 has a plurality of columns or cells, including a destination address cell, *a destination port cell* and a corresponding timer cell. Each row or entry in the filtering database 244 preferably corresponds to a particular entity.”); EX1005, 13:34-38 (“Since the multicast message 400

having the source address of server 112a was received by backbone switch 124 at a new source port, this new location information is entered by backbone switch 124 into its filtering database, *replacing the previous information that was stored therein.*”).

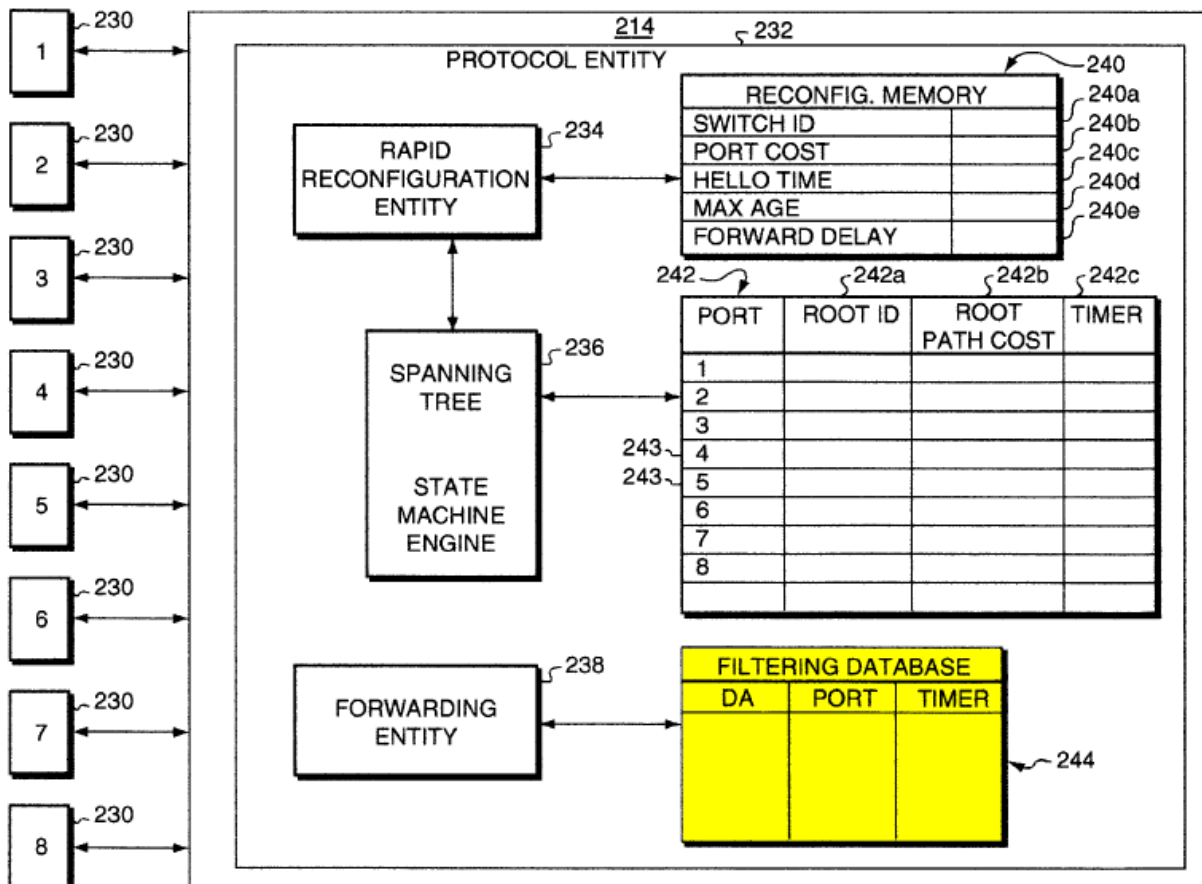


FIG. 2

EX1005, Figure 2.

135. Gai also discloses that “[s]witch 114, via rapid reconfiguration entity 234, preferably informs the other devices of its new forwarding port by transmitting dummy multicast packets *through the new port...*” EX1005, 12:62-

65. A POSITA would have understood that the contents, e.g., the port information, of the edge switch filtering database would be updated to identify the new port that should be used for transmitting messages to a particular entity, such as the dummy message(s) and/or communication messages would be sent from the appropriate port to the “other devices.”

136. As such, it would have been obvious to update the edge switch filtering database to reflect the change in the active port, such that when rapid reconfiguration entity 234 generates the dummy message(s) or when forwarding entity 238 forwards communication messages the switch would use the new port. For example, in the filtering database of switch 124, the destination port cell information in the row that corresponds to entities associated to switch 116 would be updated to reflect the port identified with the red dot.

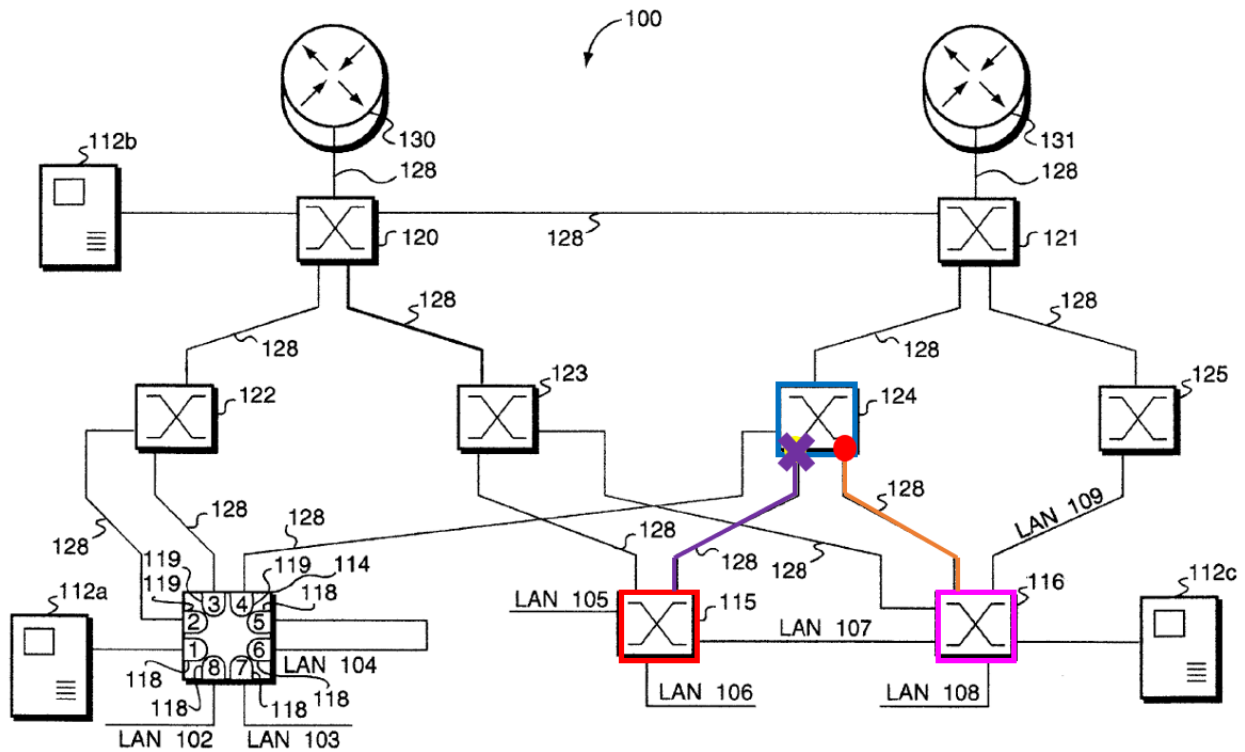


FIG. 1

EX1005, Figure 1.

6. Dependent Claims 5 and 17

i. [5pre/17pre] A method/device according to claim 4/16,

137. Gai renders obvious claims 4 and 16. See Sections VIII.A.5 and

VIII.A.12.

- ii. **[5a/17a] wherein [updating the database comprises changing]/[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.**

138. Gai discloses and/or renders obvious this limitation. For example, Gai discloses that “[*u*]pon *transition* to the new forwarding port, the device begins transmitting ‘dummy’ multicast messages through the new port.” EX1005, 5:54-56. Gai also discloses that “[s]witch 114, via rapid reconfiguration entity 234, preferably informs the other devices of its new forwarding port by transmitting dummy multicast packets *through the new port...*” EX1005, 12:62-65.

139. That is, Gai discloses that, *upon transition to the new forward port*, the dummy messages are transmitted through the new forward port. Accordingly, a POSITA would have understood that the dummy messages are transmitted before any communication traffic is received by the switch. As I discussed in Section VIII.A.5.ii, the filtering database is updated before transmitting the dummy messages. Accordingly, a POSITA would have understood that the database entries would have been updated before receiving any communication traffic from the access switches.

7. Dependent Claims 6 and 18

i. [6pre/18pre] A method/device according to claim 1/13,

140. Gai renders obvious claims 1 and 13. *See* Sections VIII.A.1 and VIII.A.2.

ii. [6a/18a] 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.

141. Gai discloses this limitation. For example, Gai discloses:

Upon transition to the new forwarding port, the device begins transmitting ‘dummy’ multicast messages through the new port. These 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; 11:16 (“As messages (i.e., data frames)...”).

Next, rapid reconfiguration entity 234 (FIG. 2) *loads SA field 412 with the address of an entity directly coupled to switch 114*. For example, in a first multicast message 400, entity 234 loads the address of server 112a into the source address field 412. ... Multicast message 400, *having the address of server 112a loaded in SA field 412*, is then forwarded through the new root port (i.e., port number four) of switch 114.

EX1005, 13:14-23; *see also*, 13:50-14:3.

142. That is, upon transitioning to the new forwarding port, switch 124 would transmit a dummy message through the new port with the source address of server 112a, which is a device coupled downstream of switch 124. As shown in Figure 1 of Gai below, server 112a is accessible to access switches 115 and 116 through switch 124.

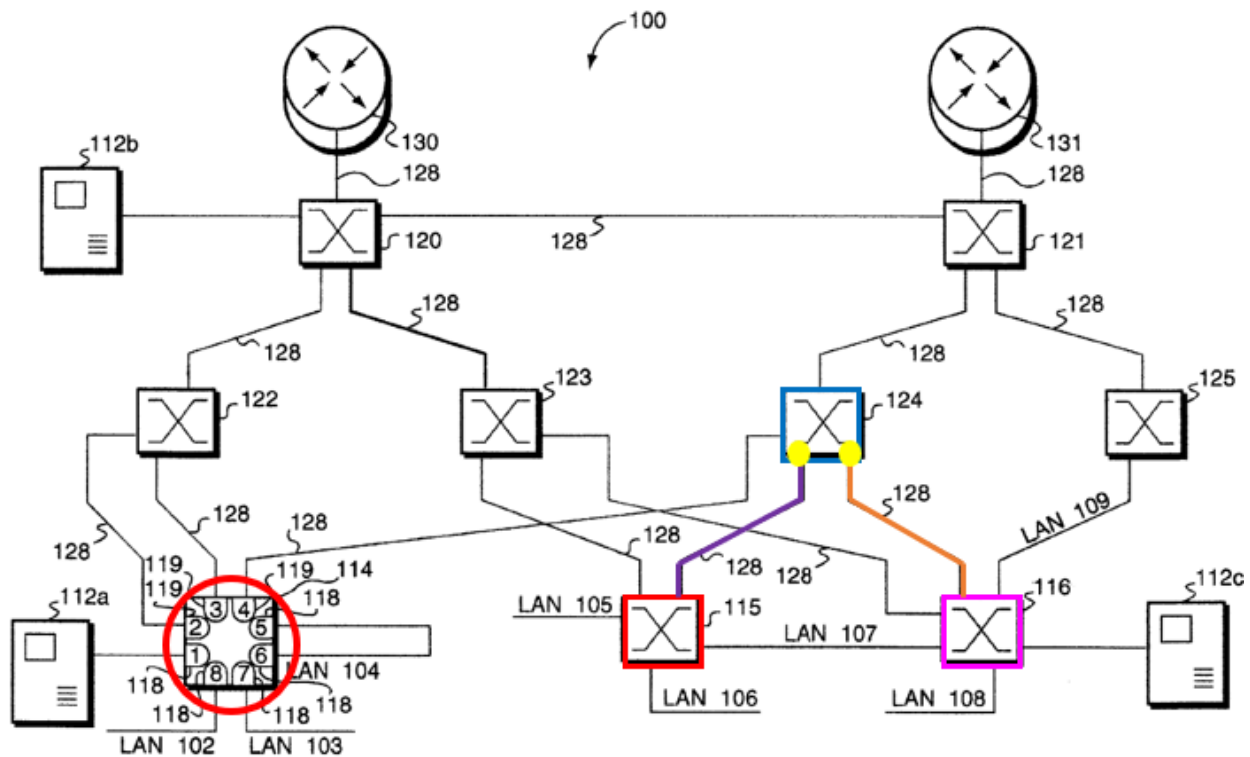


FIG. 1

EX1005, Figure 1.

8. Dependent Claim 7

i. [7pre] A method according to claim 6,

143. Gai renders obvious claim 6. *See* Section VIII.A.7.

ii. [7a] wherein sending the data frames comprises copying the source addresses for the data frames from the database of the edge switch.

144. Gai renders obvious this limitation. For example, for each dummy message sent from a switch, Gai discloses loading the address of a corresponding entity into the source address field of the dummy message. EX1005, 13:14-18 (“[R]apid reconfiguration entity 234 (FIG. 2) loads SA field 412 with the address of an entity directly coupled to switch 114. For example, in a first multicast message 400, *entity 234 loads the address of server 112a into the source address field 412.*”); 13:50-55 (“Rapid reconfiguration entity 234 similarly generates one or more separate multicast messages 400 *for each remaining entity* directly coupled to access switch 114 (i.e., entities or end stations on LANs 102-104). In each message 400, entity 234 *loads SA field 414 with the address of the corresponding entity.*”)

145. Gai also discloses that the “filtering database 244 [is] configured to store **address information** corresponding to the entities of network 100. Specifically, filtering database 244 has a plurality of columns or cells, including a

destination address cell...” and that “[e]ach row or entry in the filtering database 244 preferably corresponds to a particular entity.” EX1005, 8:31-39.

146. Accordingly, a POSITA would have found it obvious to copy into the [source address] SA field of the dummy message the information from the “destination address cell,” which corresponds to the “address information corresponding to” each entity of network 100. For example, one of the ways that a POSITA would have used to copy the destination address into the SA field of the dummy message would be performing a look up function at the filtering database to retrieve the address information. *See also*, EX1005, 11:19-21 (“The forwarding entity 238 first examines the destination address of the message and performs a look up function at the filtering database 244.”).

9. Dependent Claims 9 and 21

i. [9pre/21pre] A method/device according to claim 6/18,

147. Gai renders obvious claims 6 and 18. *See* Section VIII.A.7.

ii. [9a/21a] [wherein sending the data frames comprises incorporating]/[wherein the protocol processor is adapted to incorporate] a Virtual Local Area Network (VLAN) tag in a header of the data frames.

148. Gai discloses this limitation. For example, Figure 4 of Gai “is a block diagram of a dummy multicast message in accordance with a preferred embodiment of the present invention.” EX1005, 6:65-67.

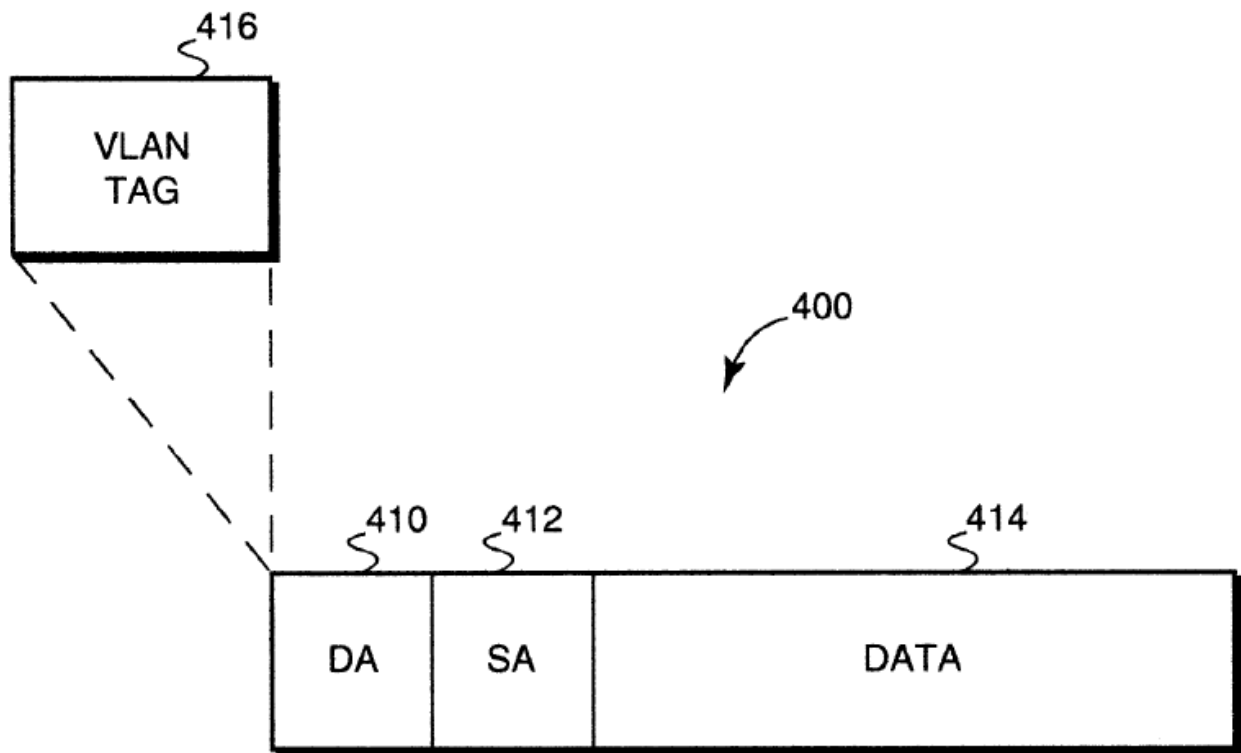


FIG. 4

EX1005, Figure 4.

149. Gai discloses incorporating, e.g., appending to or inserting in, a virtual local area network (VLAN) to message 400. EX1005, 17:32-39 (“Referring again to FIG. 4, in formulating dummy multicast messages following the detection of a change, access switches 114-116 *may append a VLAN tag 416 to message 400.* The VLAN tag 416 is preferably loaded with the VLAN designation corresponding to the entity whose address is loaded into SA field 412. Although VLAN tag 416

may be appended to the message 400, as shown, it is preferably inserted after the source address field 412.”).

150. Gai explains that access switches may append the VLAN tag to message 400. However, a POSITA would have understood that any network switch, e.g., including backbone switches, would also append a VLAN tag to new/unlabeled frames/messages, because “an objective of [the] VLAN arrangement is to allow all ports and entities of the network having the same VLAN designation to interchange messages by associating a VLAN designation with each message. Thus, those entities having the same VLAN designation function as if they are all part of the same LAN.” EX1005, 16:17-22. Accordingly, when a switch sends a frame in a VLAN arrangement, the VLAN tag would be appended to the frame, such that the frame reaches the correct VLAN destination. *See also*, EX1005, 15:52-16:27, 17:42-56.

10. Dependent Claims 10 and 22

i. [10pre/22pre] A method/device according to claim 6/18,

151. Gai renders obvious claims 6 and 18. *See* Section VIII.A.7.

- ii. **[10a/22a] [wherein sending the data frames comprises multicasting]/[wherein the protocol processor is adapted to multicast] the data frames to all of the one or more access switches.**

152. Gai discloses this limitation. For example, Gai discloses multicasting the dummy frames by sending dummy *multicast* packets with a *multicast address* that results in the dummy packet being *received by all the devices in the network*. Specifically, Gai discloses that “[s]witch 114, via rapid reconfiguration entity 234, preferably informs the other devices of its new forwarding port by transmitting *dummy multicast packets* through the new port.” EX1005, 12:62-65; *see also*, EX1005, 5:54-56 (“Upon transition to the new forwarding port, the device begins transmitting ‘*dummy*’ *multicast messages* through the new port.”). Gai also discloses that “[r]apid reconfiguration entity 234 preferably loads DA field 410 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.

153. Gai also discloses that “[b]ackbone switch 124 distributes the *multicast message 400 through all of its forwarding ports* so that other devices, as necessary, may learn of the new forwarding port (i.e., port number four) at access switch 114.” EX1005, 13:45-49; *see also*, EX1005, Abstract (“Following the transition to a new active port, *dummy multicast messages are transmitted*, each

containing the source address of an entity directly coupled to the affected device or downstream thereof. By examining the dummy multicast messages, *other devices* in the network learn to use [] the new forwarding port of the affected device.”).

11. Dependent Claims 11 and 23

- i. [11pre/23pre] A method/device according to claim 1/13,

154. Gai renders obvious claims 1 and 13. *See* Sections VIII.A.1 and VIII.A.2.

- ii. [11a/23a] wherein [sending the dummy traffic comprises sending data frames downstream]/[the dummy traffic comprises data frames sent downstream] from the edge switch toward the access switches.

155. Gai renders obvious this limitation for the reasons I discussed in Sections VIII.A.1.ii and VIII.A.1.vi.

12. Dependent Claims 16 and 19

- i. [16pre/19pre] A device according to claim 13/18, and

156. Gai renders obvious claims 13 and 18. *See* Sections VIII.A.2 and VIII.A.7.

- ii. [16a/19a] comprising a memory,

157. Gai discloses this limitation. For example, Gai discloses that “[s]witches typically learn which destination port to use in order to reach a particular entity by noting on which source port the last message originating from

that entity was received. This information is then stored by each switch in *a block of memory* referred to as a filtering database.” EX1005, 1:50-54.

158. Gai also discloses “that memory 240 and tables 242 and 244 may be implemented as *content addressable memory (CAM) devices* and that entities 234 and 238 and state machine engine 236 may comprise registers and combinational logic configured and arranged to produce sequential logic circuits.” EX1005, 8:40-45; *see also*, EX1005, 8:32-39.

- iii. **[16b/9b] 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,**

159. Gai discloses and/or renders obvious this limitation. For example, Gai discloses that switches inspect received messages and store inspected information into a filtering database. Specifically, in Gai “[s]witches typically learn which destination port to use in order to reach a particular entity by noting on which source port the last message originating from that entity was received. *This information is then stored by each switch in a block of memory referred to as a filtering database.*” EX1005, 1:50-54.

160. Gai also discloses that the information in the filtering database may be discarded, e.g., when a failure happens at an active port and a blocked port transitions to be the new active port. EX1005, 4:30-34 (“As ports transition

between the blocked and forwarding states, entities may appear to move from one port to another. To prevent switches from distributing messages based upon incorrect information, switches quickly age-out and ***discard the ‘old’ information in their filtering databases.***”). Accordingly, Gai discloses that the switch databases are built prior to a transition to a blocked port, which can be, for example, as a result of a failure. *See e.g.*, EX1005, 5:44-46 (“Upon detection of a failure at the active forwarding port, the state of one of the back-up ports immediately transitions from blocked to forwarding...”).

161. Gai further explains:

As messages (i.e., data frames) are ... received at the ports 230, they are passed to forwarding entity 238... The forwarding entity 238 first examines the destination address of the message and performs a look up function at the filtering database 244. ***Assuming an entry is found*** and a corresponding destination port (e.g., port number four) is identified, the message is switched out onto this port, assuming this port is in the forwarding state. If no entry is found in the filtering database 244, the message is flooded out all ports 230 in the forwarding state, except the port 230 on which the message was received. The forwarding entity 238 next examines the source address of the message and performs another look-up at filtering database 244. ***If no entry is found for the source address, a new entry is formed*** and the source port (e.g., port number eight) on which the message

was received is entered as the destination port in the corresponding port column. ...

Should a change occur in the network 100, such as a failure disabling the link coupled to a root port of an access switch, the affected access switch will be able to rapidly reconfigure the network 100 without the significant interruption or message flooding experienced through conventional operation of the spanning tree algorithm.

EX1005, 11:16-41.

By quickly *discarding the contents of their filtering databases 244*, the possibility of data messages being misdirected or lost is significantly reduced. Furthermore, the transmission of dummy multicast messages 400, as described above, quickly informs devices of newly activated ports, thereby reducing the flooding that otherwise occurs when the contents of filtering databases 244 are discarded.

EX1005, 15:18-25.

- iv. **[16c] 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.**

162. Gai discloses and/or renders obvious this limitation for the reasons I discussed in Section VIII.A.5.ii.

- v. **[19c] to copy the source addresses for the data frames from the edge switch database.**

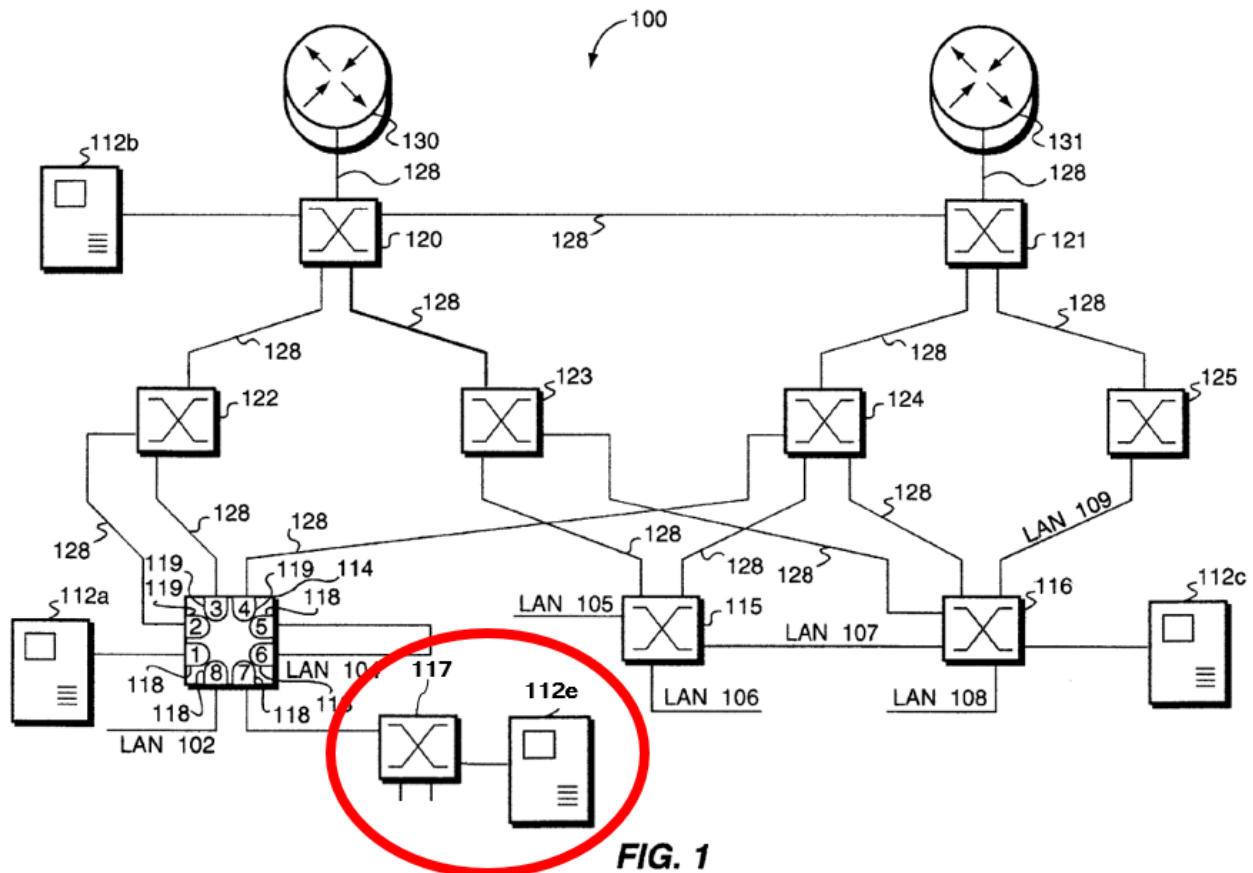
163. Gai discloses and/or renders obvious this limitation for the reasons I discussed in Section VIII.A.8.ii.

B. Ground 1B: Gai (Upstream Embodiment) Renders Obvious Claims 1-2, 4-7, 9-14, 16-19, and 21-24

164. As I explained in Section V.A, Applicant of the '986 Patent admitted that Gai discloses sending dummy packets *upstream* and distinguished the alleged '986 Patent invention as “sending *dummy traffic...downstream*” (EX1001, 12:4-5), which is “in contrast to the methods described in [Gai]...”. EX1001, 4:39-40. But not all claims require sending dummy packets downstream, and the Examiner appears to have overlooked this fact. In Ground 1B, I explain how Gai’s teachings regarding sending dummy packets upstream (Gai’s upstream embodiment) render obvious the challenged claims. The analysis I provided for Ground 1A also applies to this ground, to the extent it is not specifically directed to and limited to Gai’s downstream embodiment.

165. I explain the upstream embodiment discussed in Gai in connection with a modified Figure 1 of Gai that is provided below. *First*, Gai expressly teaches that embodiments of his invention are not limited to the exact network of Figure 1, but that they also apply to other networks that may have more complex topologies than the one depicted in Figure 1. EX1005, 7:41-44 (“It should be understood that the network 100 of FIG. 1 is meant for illustrative purposes only and that the present invention will operate with other network designs having possibly far more complex topologies.”). *Second*, Gai expressly discloses a

variation of the topology depicted in Figure 1. Specifically, Gai discloses that port seven of switch 114 can be coupled to another switch that has entities directly coupled to it. EX1005, 13:60-63 (“[I]f port number seven at switch 114 were *coupled to another access switch*, entity 234 would transmit multicast messages 400 carrying the source address of *entities directly coupled to this additional access switch* as well.”). Such network is illustrated below, where the topology of the network in Figure 1 of Gai has been modified to add switch 117 directly connected to port seven of switch 114 and to also add server 112e directly connected to a port of switch 117.



EX1005, Figure 1 (modified to add switch 117 and server 112e).

166. In this exemplary network, **switch 114** is both an access switch, as well as an **edge switch**. **Switch 114** is a core switch, because it interconnects access switches 117, 120, 115, 116, and 125 into a WAN, e.g., by direct connection between core switch 114 and access switch 117. Additionally, **switch 114** is directly connected to an access switch, such as **switch 117**. See EX1001, 1:33-38 (“Core 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. Core switches that are directly connected to access switches are referred to herein as edge switches.”).

1. Independent Claim 1

i. [1pre] A method for network communication, comprising:

167. I understand that the preamble may not be limiting. Nevertheless, Gai discloses and/or renders obvious the preamble. See Section VIII.A.1.i.

- ii. [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**;

[12a] 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**;

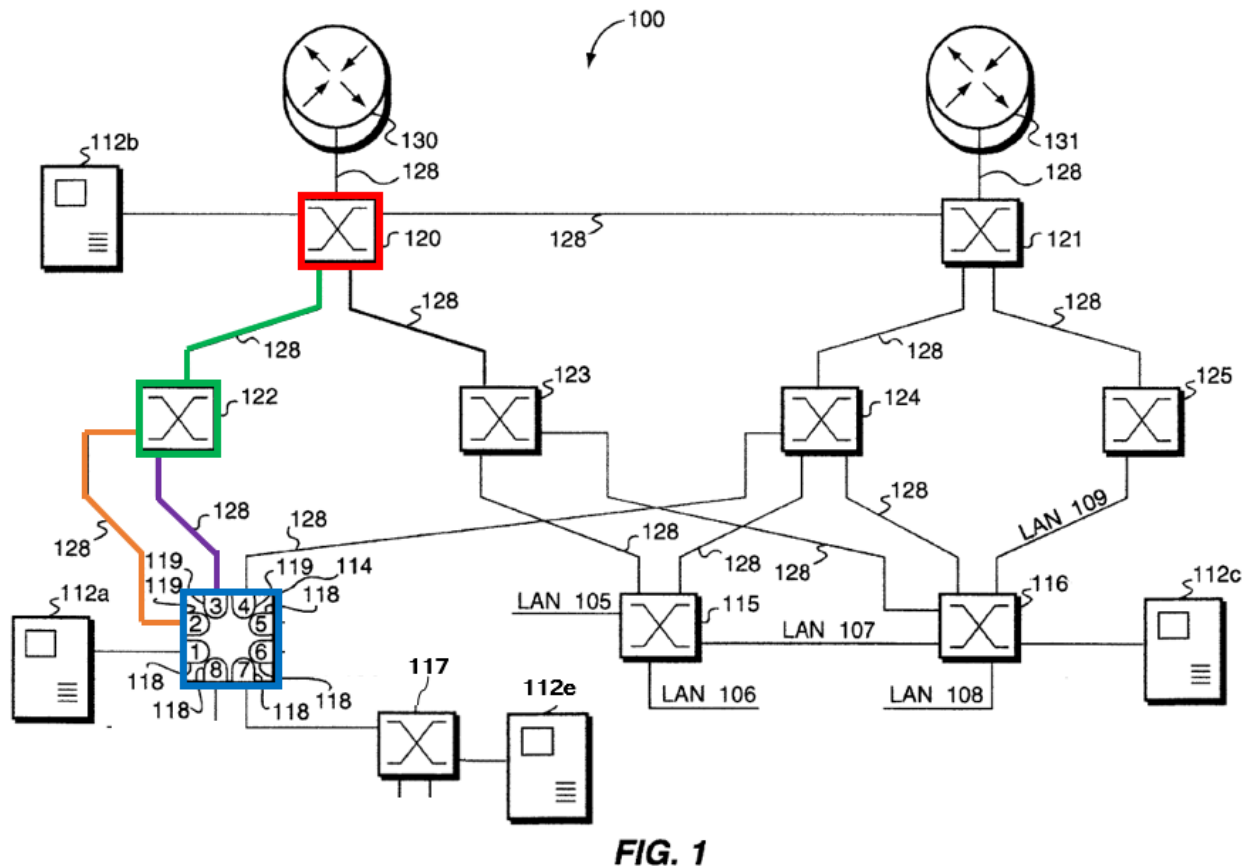
168. Gai discloses and/or renders obvious limitations [1a/12a]. As the '986 patent explains, a “switch” is one particular type of a “bridge.” *See, e.g.*, EX1001, 1:27 (“The most common type of MAC bridge is a switch...”); 4:50-54 (“Although preferred embodiments are described herein with reference to certain switches, and particularly to edge switches and access switches used in a WAN, the principles of the present invention are generally applicable to bridges of other types and to other network configurations.”).

169. As the '986 Patent also admits, Gai discloses an **edge switch** and one or more **access switches**. *See* EX1001, 3:16-24.

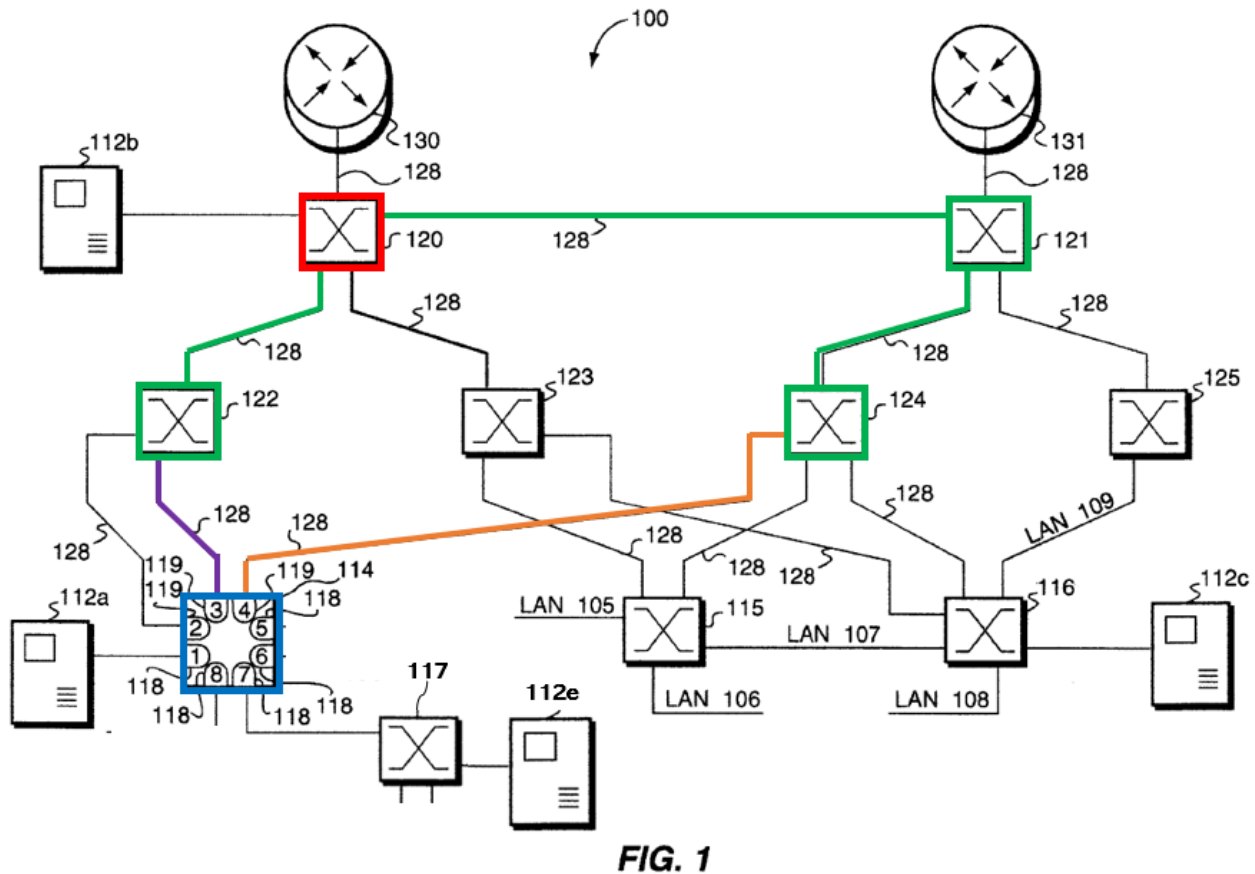
170. Indeed, as I explain below, Gai’s **switch 114** satisfies the claim terms **edge switch/first bridge** and Gai’s **access switch 120** satisfies the claim terms **one or more access switches/second bridge**.

171. **Switch 114**, as shown in the two examples below, satisfies the claim term **edge switch** because **edge switches** are “[c]ore switches that are directly

connected to **access switches**.” EX1001, 1:36-38. *First*, **switch 114** is directly connected to **access switch 117**. *Second*, **switch 114** is a core switch because, as shown below, it “interconnect[s] the **access switch [117]** into a Wide Area Network (WAN), by direct connection between the core switch and access switches and/or by connection to other core switches [such as switch 124].” EX1001, 1:33-36; *see also* EX1005, 1:31-34.



EX1005, Figure 1 (modified).



EX1005, Figure 1 (modified).

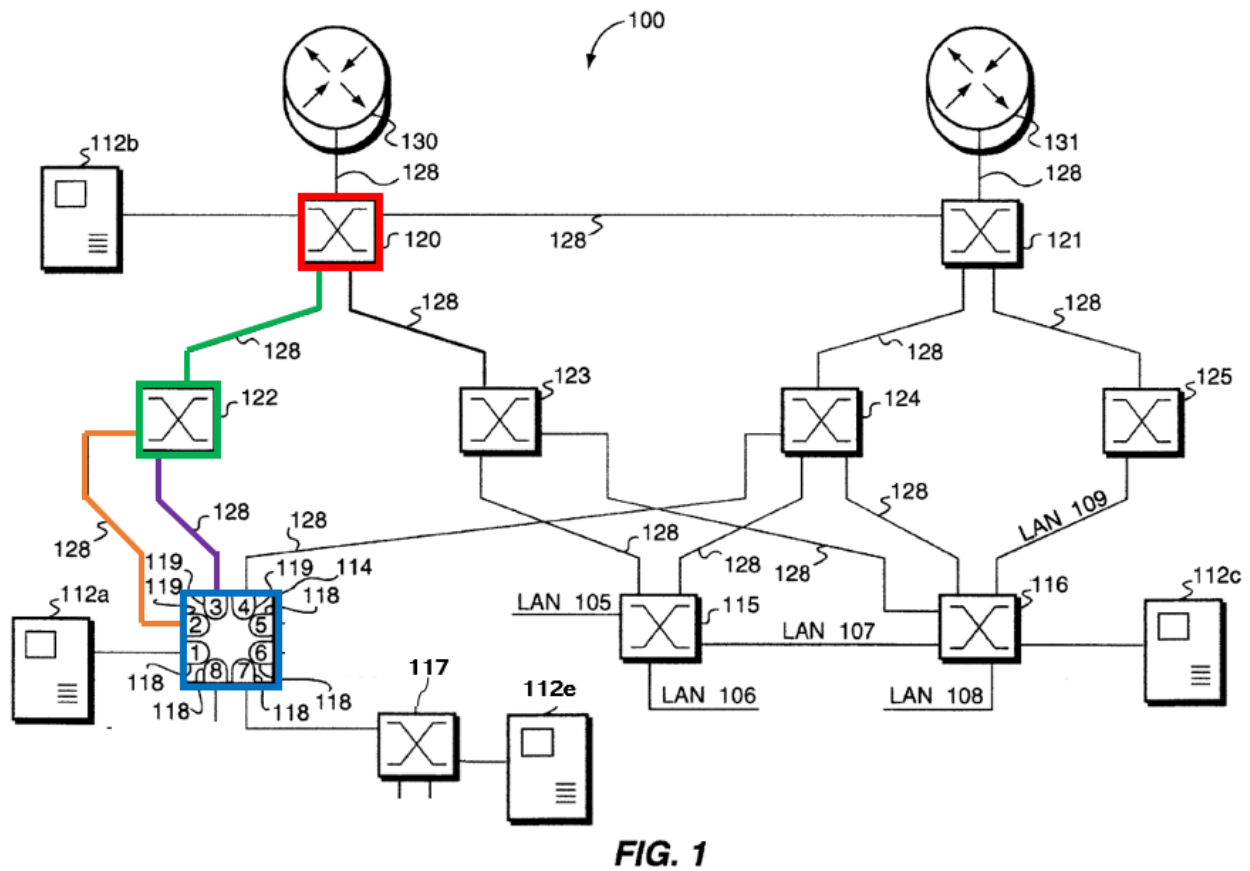
172. Gai also discloses coupling the **edge switch** to the one or more **access switches** via **at least first** and **second** redundant links, respectively connected to a first port, e.g., port 3 of switch 114, and a second port, e.g., either port 2 or port 4 of switch 114, ports of the **edge switch**.

173. As I explained in Section V.A.3, claim 1, and therefore limitation [1a], is not limited to a downstream transmission of dummy frames. However, limitation [12a] specifies that the “**second bridge** [is] located downstream of the **first bridge**,” i.e., away from the root switch. See EX1001, 3:30-31. But, as I

further explained in Section IV.A.2, any switch can be the root switch, e.g., the switch with the lowest switch identifier or a switch that reduces the average distance between end switches and a primary server. EX1005, 2:42-47

(“[E]xecution of the spanning tree algorithm causes the switches to elect a single switch, among all the switches within each network, to be the ‘root’ switch. Each switch has a unique numerical identifier (switch ID) and the root is the switch having the lowest switch ID numeric value...”); 10:29-33 (“The conventional spanning tree algorithm, however, simply selects the switch having the lowest ID to become the root, regardless of where in the network 100 this switch is located. As a result, an access switch is often elected to be the root...”); 3:17-39.

174. Accordingly, in the two examples below, it would have been obvious that *second bridge 120* would be located downstream of *first bridge 114*, if switch 117 was the root switch, because, for example, it was the switch with the lowest switch identifier.



EX1005, Figure 1 (modified).

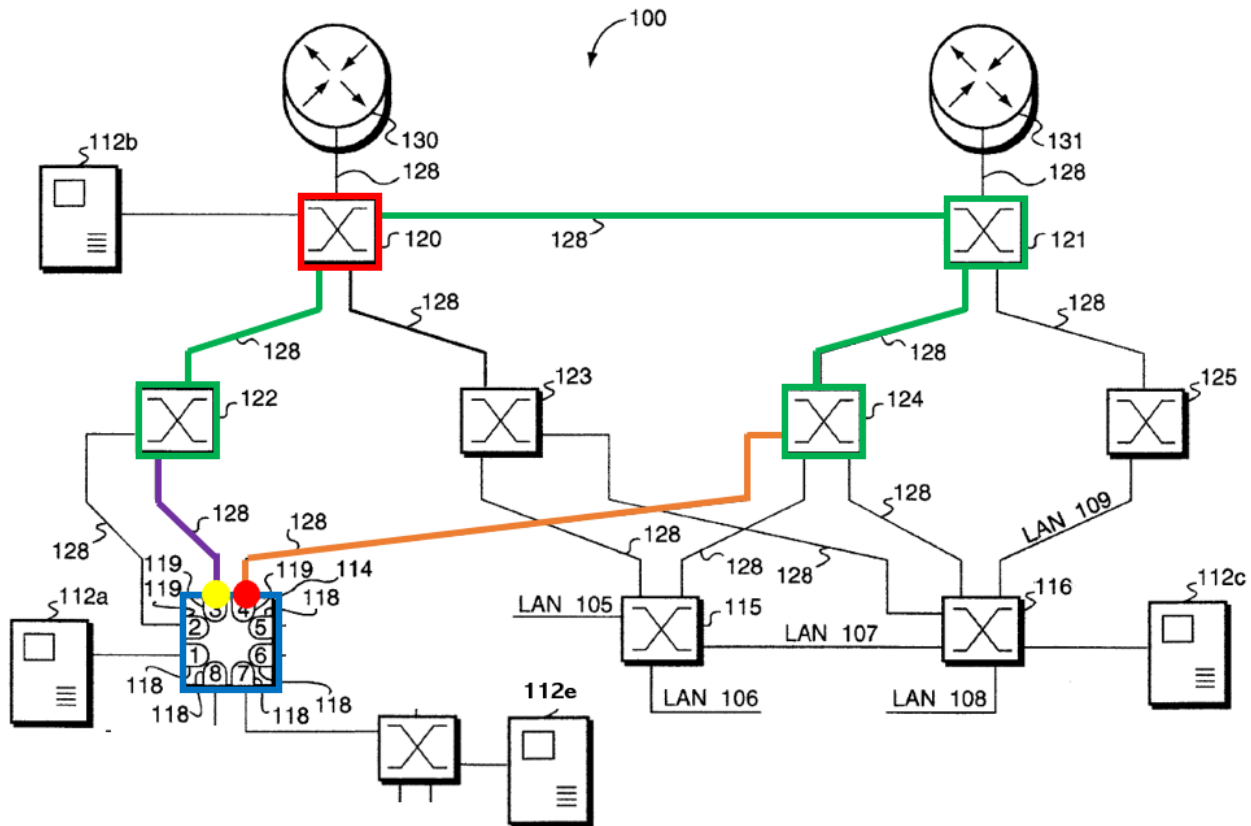


FIG. 1

EX1005, Figure 1 (modified).

177. Specifically, Gai discloses “[w]ith reference to **switch 114**, for example, assume that **port number three**, [is] in the forwarding state... **Port numbers two and four**, are in the blocked state.” EX1005, 11:41-48; *see also*, Section VIII.A.1.iii.

- iv. **[1c/12c] conveying communication traffic over the first link while the first port is in the active state, thereby causing [each of the switches/the bridges] to build a respective database for use in forwarding the traffic;**

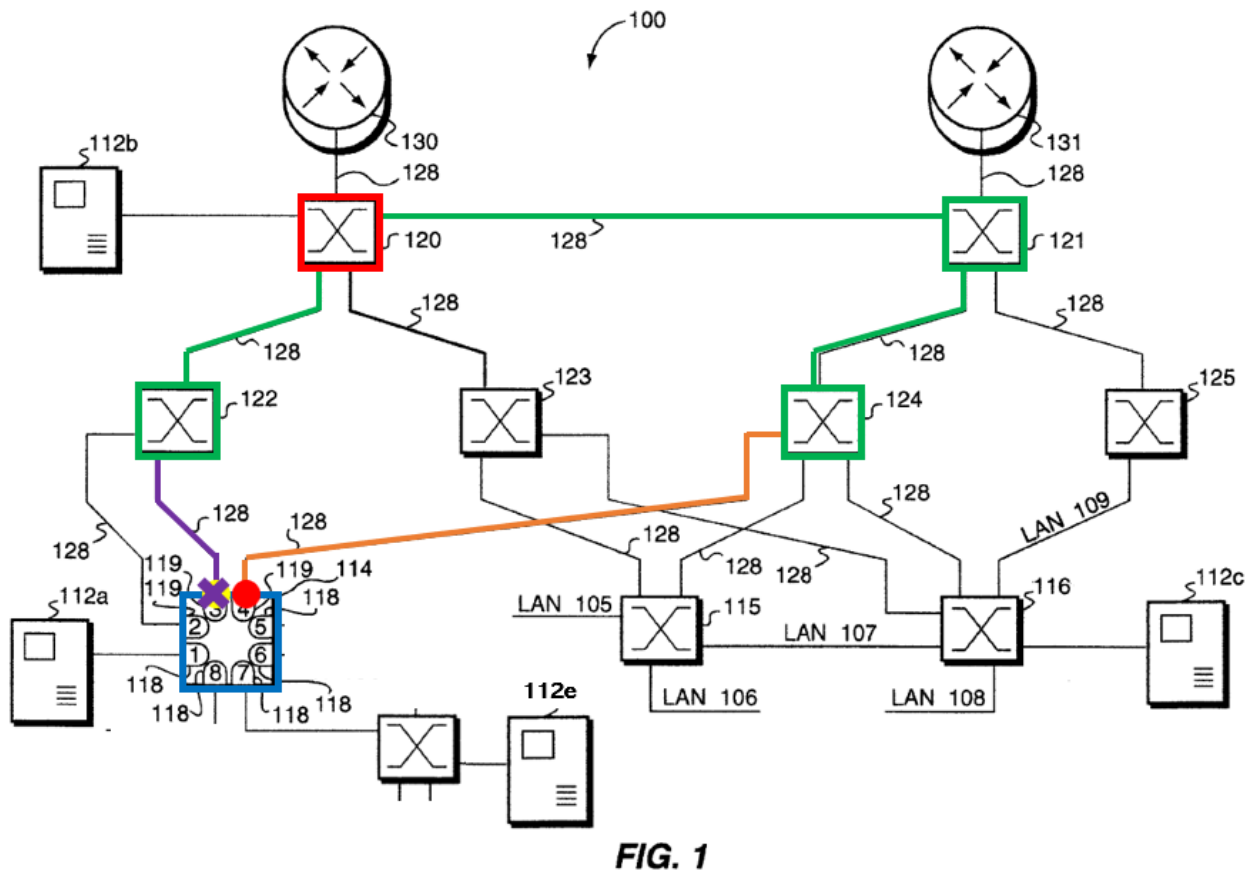
178. Gai discloses and/or renders obvious this limitation. *See* Section VIII.A.1.iv.

v. [1d/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

179. Gai discloses and/or renders obvious this limitation.

180. As an initial matter, the '986 Patent admits that Gai discloses this limitation. 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.”).

181. Indeed, Gai discloses that “[u]pon detection of a failure at the active forwarding port [indicated with a **purple X**], the state of one of the back-up ports [in **red dot**] immediately transitions from blocked to forwarding, thereby becoming the new active port for the device.” EX1005, 5:44-47; *see also*, EX1005, Abstract (“Upon detection of a failure at the active forwarding port, one of the back-up ports immediately transitions from blocked to forwarding, thereby becoming the new active port for the device.”); 5:20-24; 11:36-41.



EX1005, Figure 1 (modified).

182. A failure in a port is a failure associated with the link connected to the port, because the port is one endpoint of the link it is connected to. In addition, a failure on the port would result in a failure to the connected link.

183. Gai, in connection with Figure 3D below, shows “a flow diagram of a rapid reconfiguration process 340 following a *link failure*... In response to the detection of a *failure at port number three* (the root port), indicated at block 342, rapid reconfiguration entity 234 at switch 214 *selects a backup port* to become the new root port, as shown at box 344.” EX1005, 12:13-19; *see also* EX1005, 12:32-

35 (“Rapid reconfiguration entity 234, at block 346, then directs the spanning tree state machine engine 236 *to immediately transition the selected back-up port (e.g., port number four) to the forwarding state.*”).

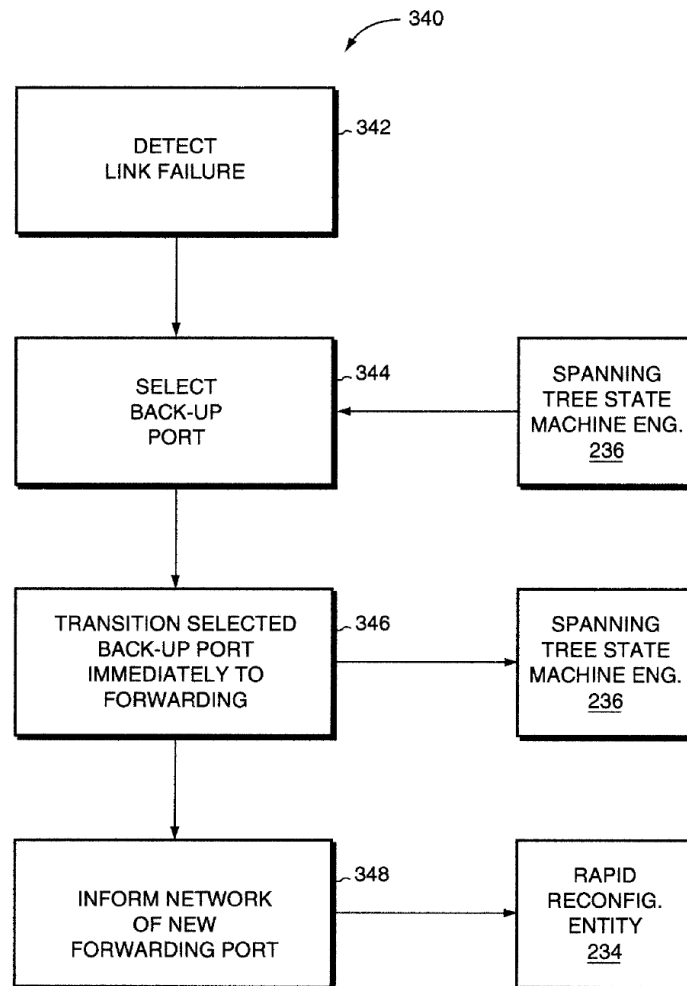


FIG. 3D

EX1005, Figure 3D; *see also*, EX1005, 12:29-31 (“[T]he blocked trunk port having the lowest port number (e.g., *port number two*) may be selected.”).

- vi. [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.

[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 respective database responsive to the second port being in the active state.

184. Gai discloses and/or renders obvious this limitation. As discussed in Section V.A, the '986 Patent admits that Gai discloses sending dummy traffic from a switch such that other switches in the network learn of the new active port. *See* EX1001, 3:29-35 (“The switch then transmits dummy multicast messages toward the root (upstream) through the newly-active trunk port, each containing the source address of a network entity that is coupled to the switch itself or is downstream of the switch.”). While the admission is explicitly directed to an upstream transmission of dummy messages—which satisfies claim 1—as more fully explained below, Gai renders obvious downstream transmission of dummy messages in connection with this embodiment as well.

185. *First*, Gai discloses sending dummy traffic from the **edge switch** over the second link to the one or more **access switches**. Gai discloses:

[T]he transition at access switch 114 from initial root port number three to back-up root port number four [or two] may cause entities to

appear to ‘move’ relative to other devices. Thus, ***the identity of the new root port (e.g., port number four) must be propagated to the other devices***, such as access switches 115-116 and backbone switches 120-125 to prevent messages from being lost or misdirected.”

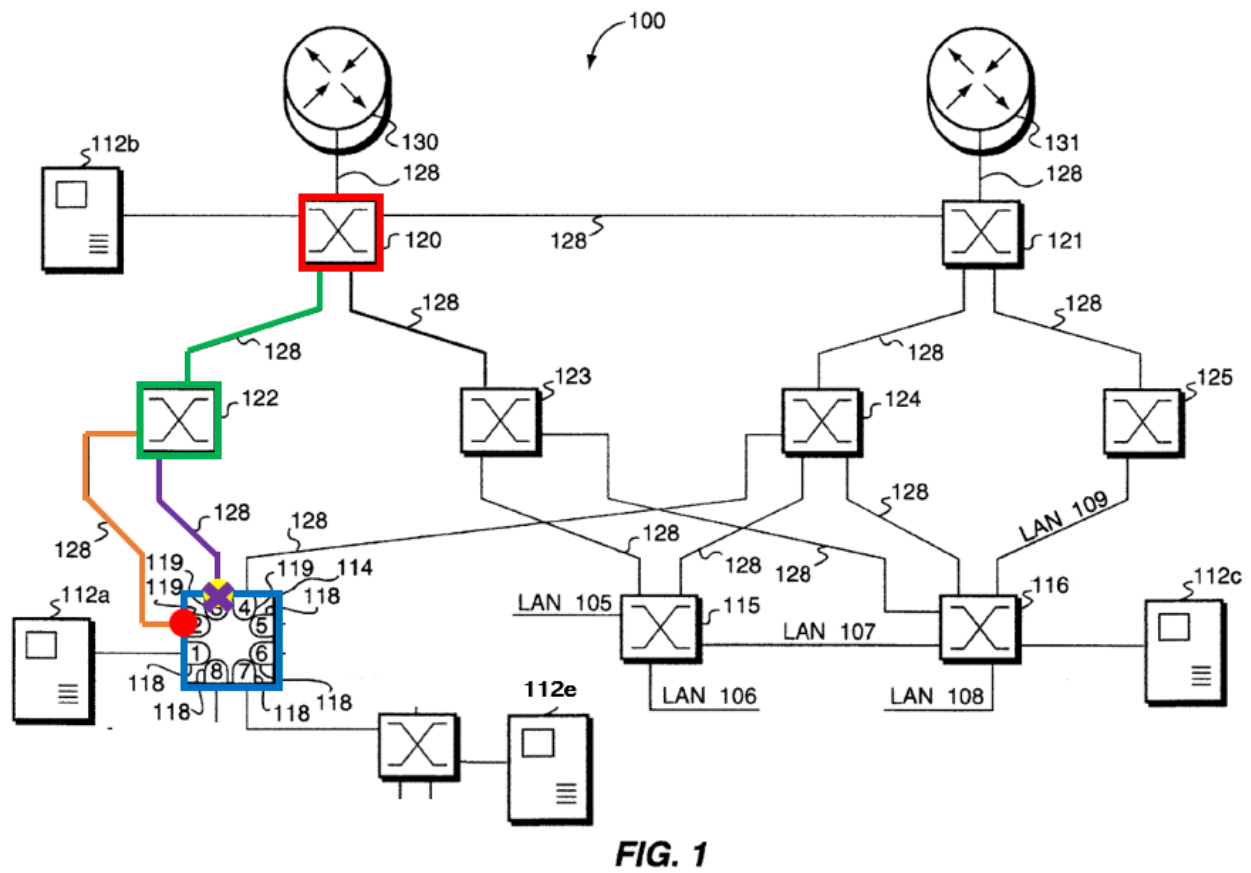
EX1005, 12:56-62; *see also*, EX1005, 12:28-31 (“It should be understood that other methods may be used to select the new root port. For example, the blocked trunk port having the lowest port number (e.g., port number two) may be selected.”).

186. Gai also discloses transmitting dummy messages through the new active port (port 2 or port 4) of switch 114 to inform the other devices of the new active port. EX1005, 12:62-66 (“Switch 114, via rapid reconfiguration entity 234, preferably informs the other devices of its new forwarding port ***by transmitting dummy multicast packets through the new port***, as indicated at block 348.”); 11:48-52 (“If the link 128 coupled to the port number three at switch 114 fails, either one of these two other ports (i.e., port numbers two or four) will immediately transition to the forwarding state and begin receiving and sending messages.”); EX1005, 5:54-56 (“Upon transition to the ***new forwarding port***, the device begins transmitting ‘dummy’ multicast messages ***through the new port***.”); *see also*, EX1005, 12:66-13:49.

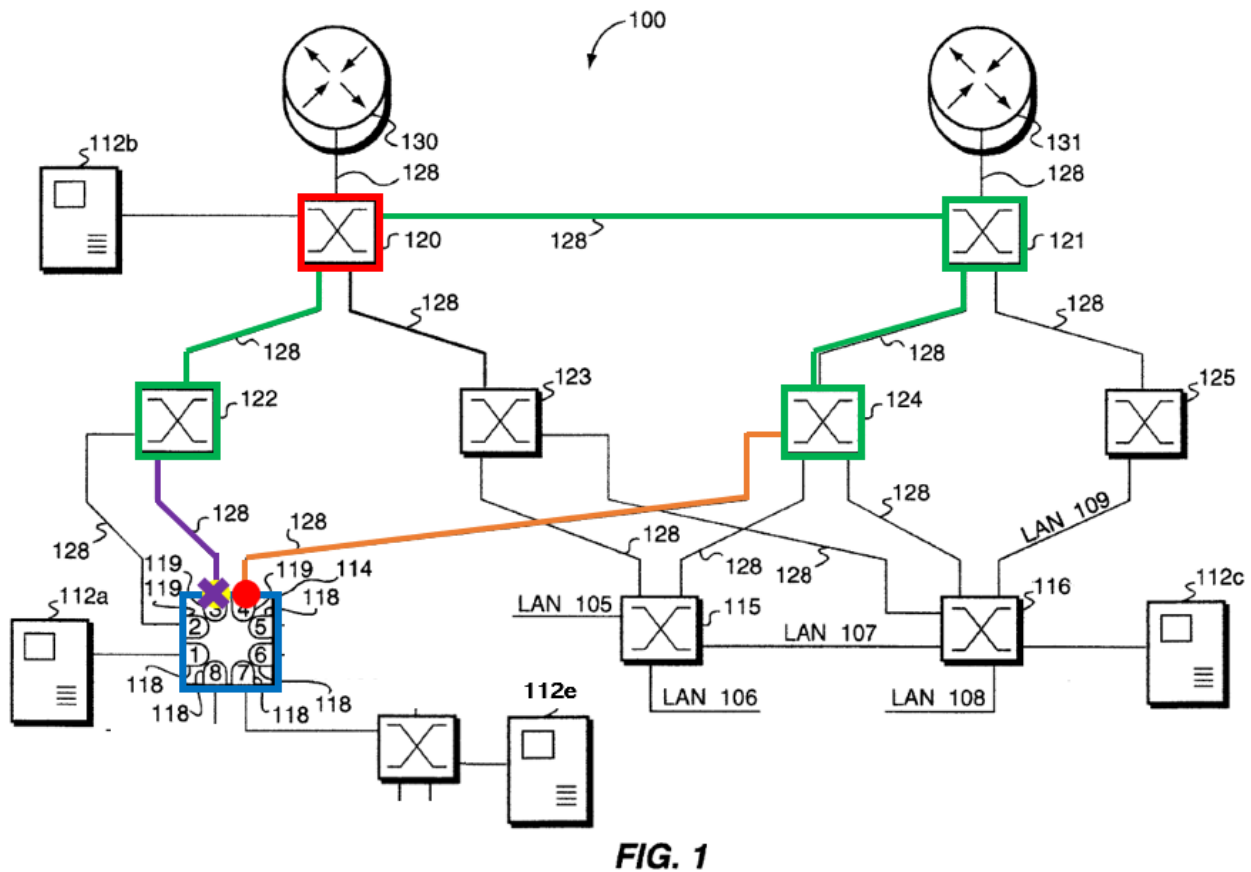
187. Gai further discloses:

Rapid reconfiguration entity 234 similarly generates *one or more separate multicast messages 400* for each remaining entity directly coupled to access switch 114 (i.e., entities or end stations on LANs 102-104). In each message 400, entity 234 loads SA field 414 with the address of the corresponding entity. Each of these messages are similarly forwarded by access switch 114 through the new forwarding port (i.e., port number four). In addition, entity 234 generates and transmits one or more multicast messages 400 for entities reachable through switch 114 on ports other than its root port. For example, if port number seven at switch 114 were coupled to another access switch, entity 234 would transmit multicast messages 400 carrying the source address of entities directly coupled to this additional access switch as well. Thus, at least one multicast message 400 is generated and transmitted for each entity directly coupled to or downstream of access switch 114 (relative to the root). *As set forth above, upon receipt of these multicast messages 400, the other devices within network 100 update their corresponding filtering databases with the new destination ports, as necessary.*

EX1005, 13:50-14:3.



EX1005, Figure 1 (modified).



EX1005, Figure 1 (modified).

188. Figure 4 of Gai, which is reproduced below, “is a highly schematic illustration of a *dummy multicast message* 400 preferably utilized by rapid reconfiguration entity 234.” EX1005, 12:66-13:1.

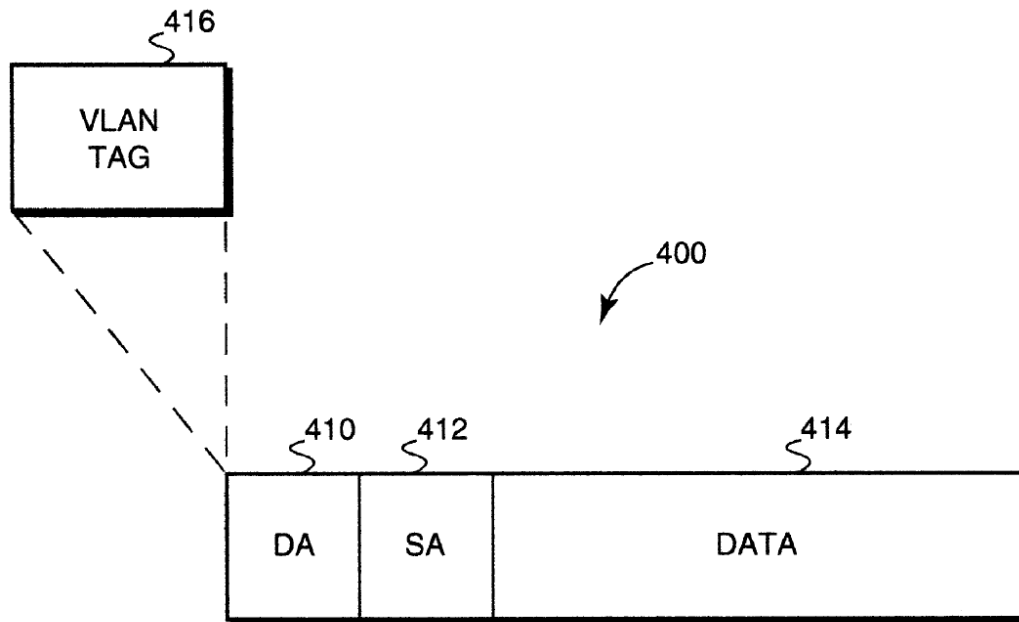


FIG. 4

EX1005, Figure 4.

189. *Second*, Gai discloses “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,” for the reasons I explained in Section VIII.A.1.vi. For example, when access switch 120 receives a dummy message from switch 121 with a source address corresponding to server 112a, it would update its corresponding database, such that the next time switch 120 transmits a frame addressed to server 112a, it would use the appropriate port, e.g., the port to send the frame towards switch 121, because that new port would be the part that is part of the path of spanning tree that couples access switch 120 with server 112a. Gai also discloses that when a packet

is received with a source address on a different receive port, the old entry in the filtering database is replaced with the new entry indicating the new port. EX1005, 13:34-38 (“Since the multicast message 400 having the source address of server 112a was received by backbone switch 124 at a new source port, this new location information is entered by backbone switch 124 into its filtering database, *replacing the previous information that was stored therein.*”).

190. Limitation [12e] adds that the “dummy traffic” is sent downstream (away from the root switch). See EX1001, 3:30-31 (“[T]ransmits dummy multicast messages toward the root (upstream)”). Gai renders obvious this additional limitation, because any switch in the network can be the root switch. For example, the network of Figure 1 assumes that **switch 121** is the root. EX1005, 14:16-17 (“For example, assume that backbone switch 121 (FIG. 1) is the root...”). But, as discussed in Section IV.A.2, any switch in the network can be the root switch. For example, the switch with the lowest switch identifier or a switch that reduces the average distance between end switches and a primary server. EX1005, 2:42-47; 10:29-33; 3:17-39. Accordingly, it would have been obvious that **first bridge 114** would send dummy traffic downstream through the new active ports (indicated by the red dot) over **link 128**, away from the root switch and towards **second bridge 120**, if switch 117 was the root, because, for example, it was the switch with the lowest switch identifier.

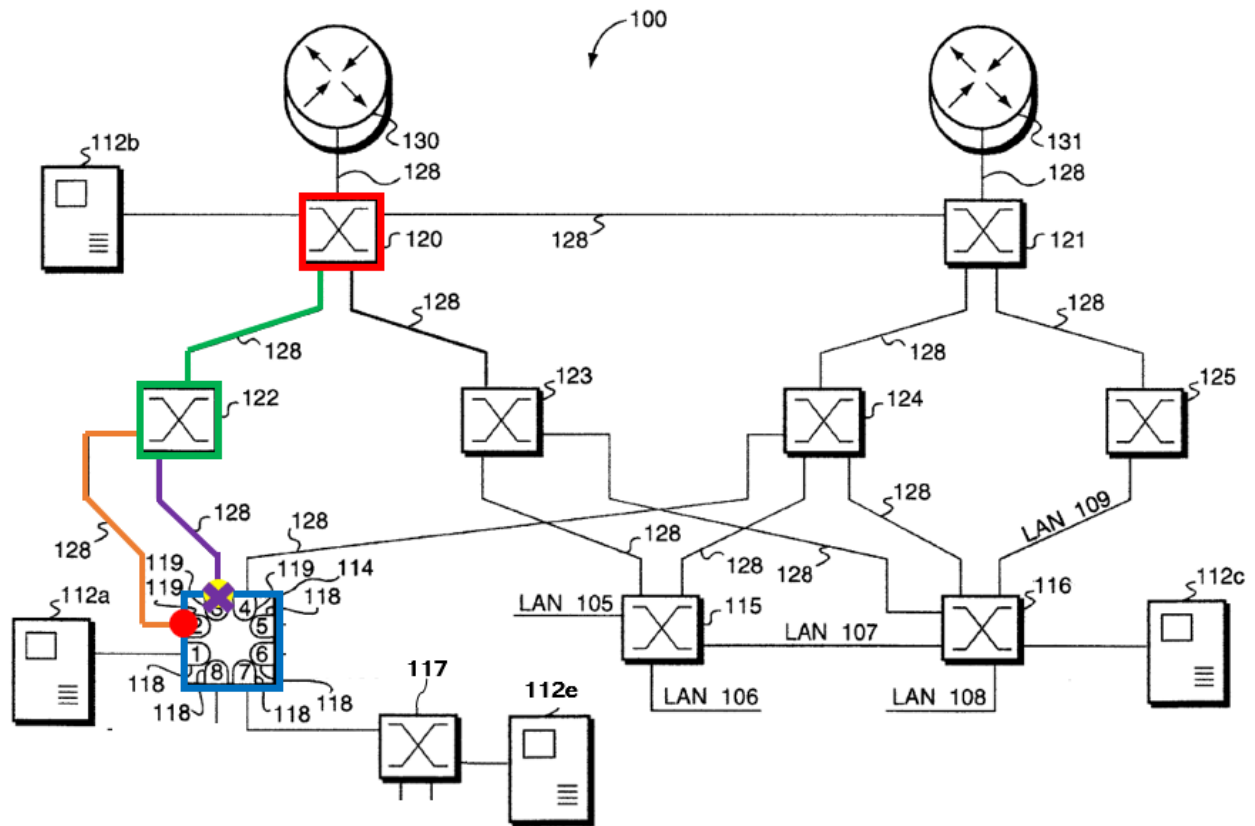


FIG. 1

EX1005, Figure 1 (modified).

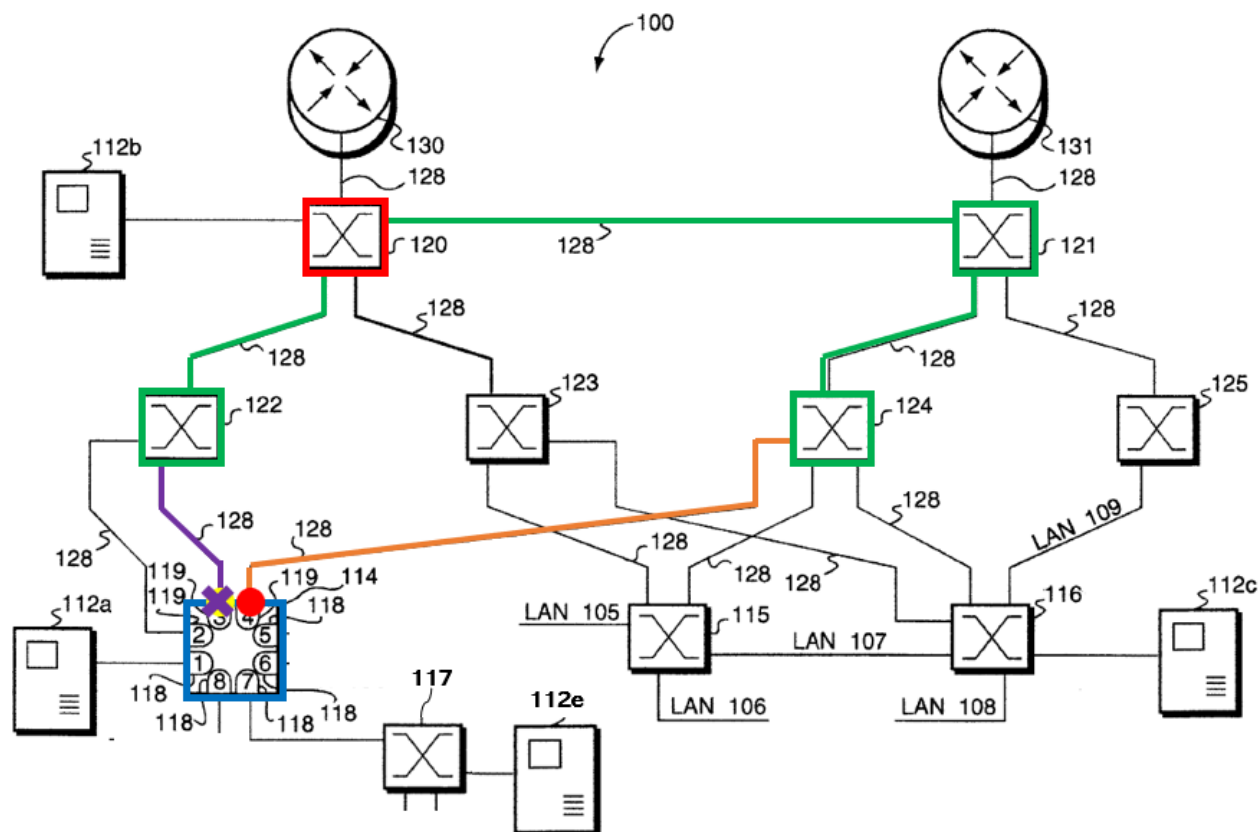


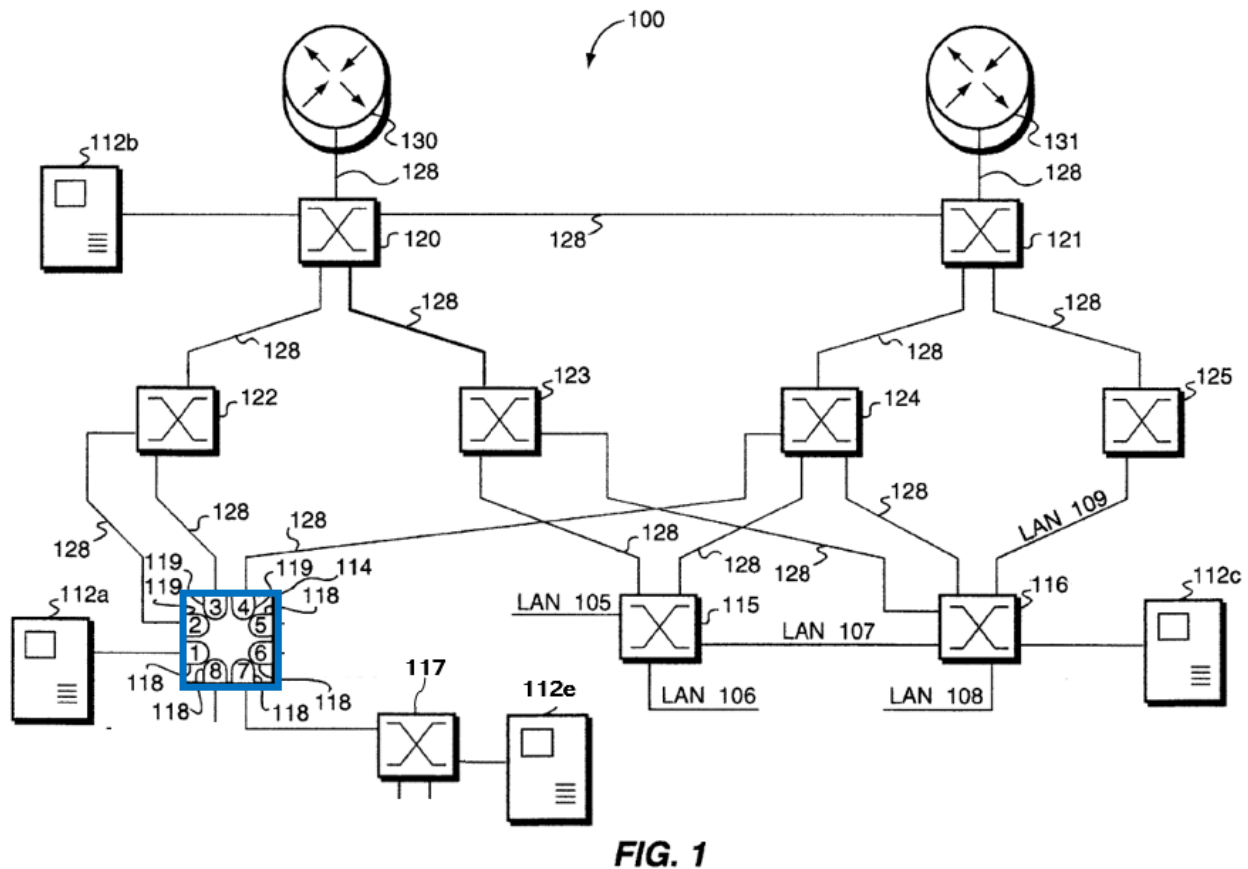
FIG. 1

EX1005, Figure 1 (modified).

2. Independent Claim 13

i. [13pre] A **network switching device**, comprising:

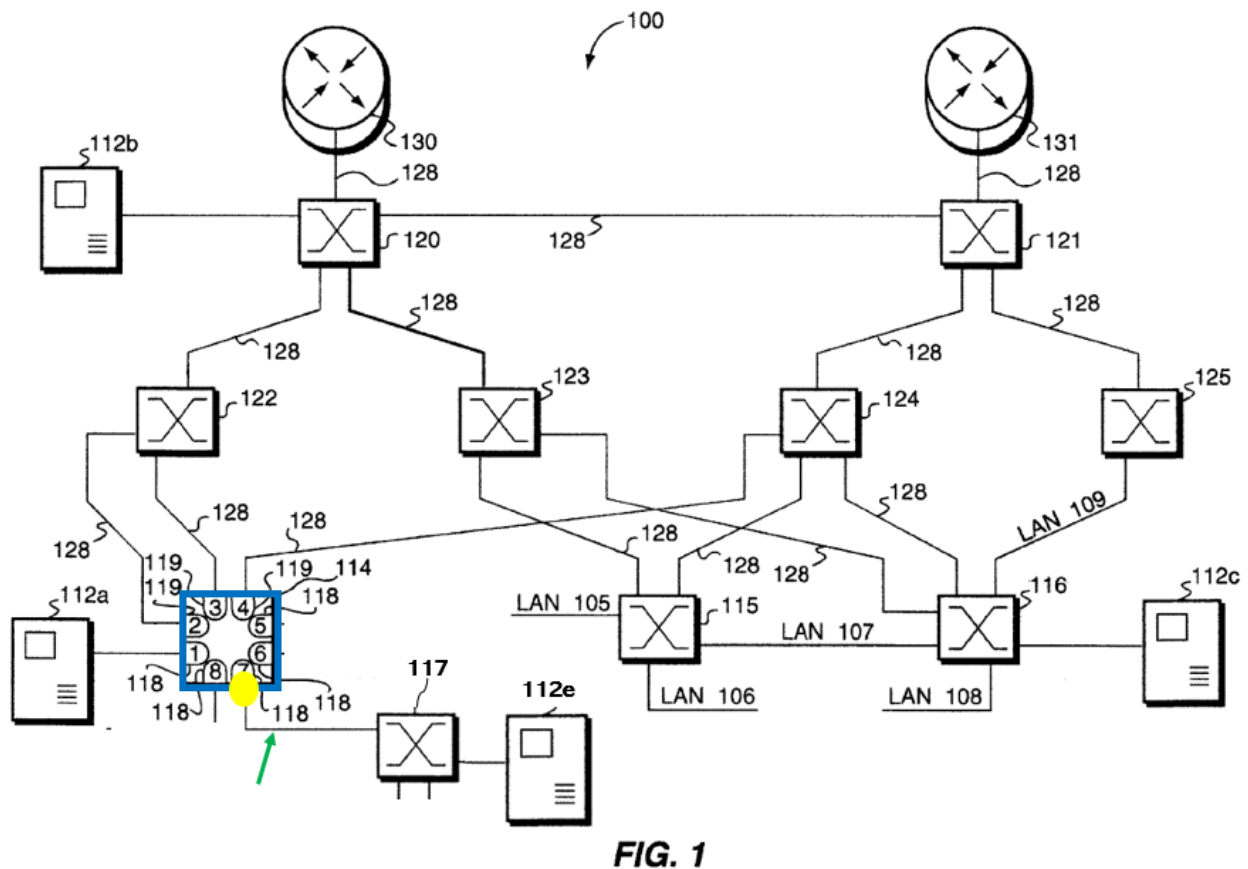
191. I understand that the preamble may not be limiting. Nevertheless, Gai discloses and/or renders obvious the preamble. For example, Figure 1 of Gai discloses various examples of a *network switching device*, e.g., *switch 114*.



EX1005, Figure 1.

- ii. [13a] 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;

192. Gai discloses and/or renders obvious this limitation. For example, in the example below, *switch 114* has network port seven (identified with a yellow dot) coupled to switch 117 over link 128 (identified with a green arrow), and meets the claimed “trunk link.” I note that the term “trunk link” does not appear in the Specification of the ’986 Patent.



EX1005, Figure 1 (modified).

193. Moreover, a POSITA would have also understood that network port seven of *switch 114* exchanges communication traffic with switch 117 because it is directly connected—through the link—to the switch 117. As shown in modified Figure 1 above, switch 117 is an access switch, because it is connected to a network endpoint, e.g., server 112e. *See* EX1001, 1:31-32 (“An access switch has ports that connect to LANs and/or to other network endpoints...”). However, a POSITA would have understood that switch 117 would satisfy a “core” switch, if for example, one of its ports was connected to another switch. *See* EX1001, 1:33-

36 (“Core 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.”).

- iii. **[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**

194. Gai discloses and/or renders obvious this limitation for the reasons I discussed in Sections VIII.B.1.ii and VIII.A.2.iii.

- iv. **[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,**

195. Gai discloses and/or renders obvious this limitation. *See* Section VIII.A.2.iv.

- v. **[13d] 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,**

196. Gai discloses and/or renders obvious this limitation for the reasons I discussed in Section VIII.B.1.iv.

- vi. **[13e] 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**

197. Gai discloses and/or renders obvious this limitation for the reasons discussed in Sections VIII.B.1.v and VIII.A.2.vi.

- vii. **[13f] [the protocol processor being further adapted, responsive to a failure associated with the first link] 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.**

198. Gai discloses and/or renders obvious this limitation for the reasons discussed in Section VIII.B.1.vi and VIII.A.2.vii.

3. Independent Claim 24

- i. **[24pre] A bridge device for use in a communication network having a spanning tree root, the device comprising:**

199. Gai discloses and/or renders obvious this limitation for the reasons discussed in Sections VIII.B.2.i, VIII.B.1.ii, VIII.B.1.vi.

- ii. **[24a] a designated port, adapted to be coupled over an upstream link in the network so as to communicate with the root;**

200. Gai discloses and/or renders obvious this limitation for the reasons discussed in Sections VIII.B.2.ii and VIII.A.3.ii. . Additionally, the link

connected to port 7 of switch 114 would be an upstream link, if switch 117 was the root, because, for example, it was the switch with the lowest switch identifier, as I discussed in Section VIII.B.1.ii.

- iii. **[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**

201. Gai discloses and/or renders obvious this limitation for the reasons discussed in Sections VIII.B.1.ii and VIII.B.1.vi.

- iv. **[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,**

202. Gai discloses and/or renders obvious this limitation for the reasons discussed in Sections VIII.B.2.iv and VIII.B.1.vi.

- v. **[24d] 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,**

203. Gai discloses and/or renders obvious this limitation for the reasons discussed in Sections VIII.B.2.v and VIII.B.1.vi.

- vi. **[24e] 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.**

204. Gai discloses and/or renders obvious this limitation for the reasons discussed in Sections VIII.B.2.vi, VIII.B.2.vii, and VIII.B.1.vi.

4. Dependent Claims 2 and 14

- i. **[2pre/14pre] A method/device according to claim 1/13,**

205. Gai renders obvious claims 1 and 13. *See* Sections VIII.B.1 and VIII.B.2.

- ii. **[2a/14a] wherein [placing the second port in the blocking state comprises configuring]/[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.**

206. Gai discloses and/or renders obvious this limitation. *See* Section VIII.A.4.ii.

5. Dependent Claim 4

- i. **[4pre] A method according to claim 1, and comprising,**

207. Gai renders obvious claim 1. *See* Section VIII.B.1.

- ii. **[4a] 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.**

208. Gai discloses and/or renders obvious this limitation. For example, Gai discloses this limitation for the reasons I discussed in Section VIII.A.5.ii.

209. Alternatively, Gai renders obvious this limitation. Specifically, a POSITA would have found it obvious to update the filtering database of the edge switch to reflect the change in the active ports, such that when rapid reconfiguration entity 234 generates the dummy message(s) or when forwarding entity 238 forwards communication messages they would use the new port. For example, in the database of switches 122 and 124, the destination port cell information in the row that corresponds to switch 114 would be updated to reflect the ports two and four, respectively.

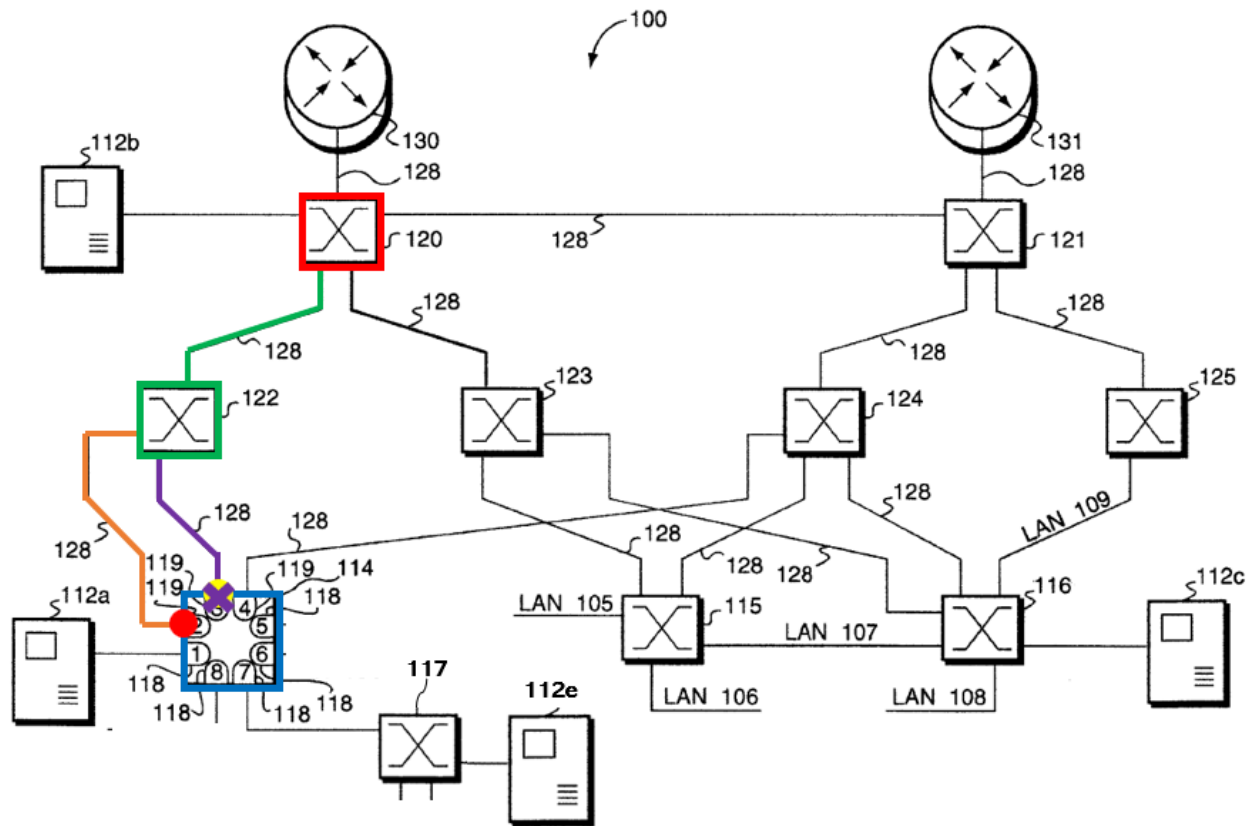


FIG. 1

EX1005, Figure 1 (modified).

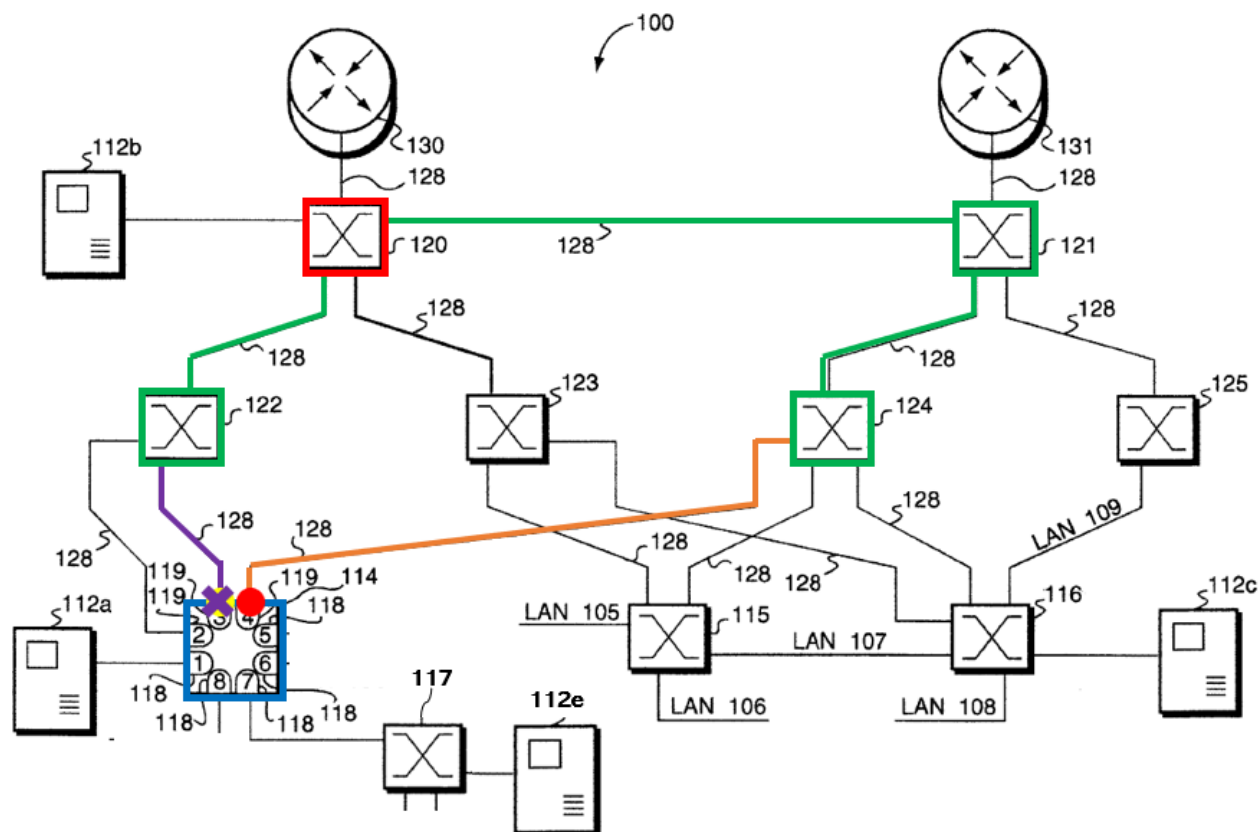


FIG. 1

EX1005, Figure 1 (modified); *see also*, EX1005, 13:34-38 (“Since the multicast message 400 having the source address of server 112a was received by backbone switch 124 at a new source port, this new location information is entered by backbone switch 124 into its filtering database, *replacing the previous information that was stored therein.*”).

6. Dependent Claims 5 and 17

i. [5pre/17pre] A method/device according to claim 4/16,

210. Gai renders obvious claim 4. *See* Section VIII.B.5.

- ii. **[5a/17a] wherein [updating the database comprises changing]/[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.**

211. Gai discloses and/or renders obvious this limitation. *See* Section

VIII.A.6.ii.

7. Dependent Claims 6 and 18

- i. **[6pre/18pre] A method/device according to claim 1/13,**

212. Gai renders obvious claims 1 and 13. *See* Sections VIII.B.1 and

VIII.B.2.

- ii. **[6a/18a] 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.**

213. Gai discloses this limitation. *See* Section VIII.A.7.ii.

8. Dependent Claim 7

- i. **[7pre] A method according to claim 6,**

214. Gai renders obvious claim 6. *See* Section VIII.A.7.

- ii. **[7a] wherein sending the data frames comprises copying the source addresses for the data frames from the database of the edge switch.**

215. Gai renders obvious this limitation. *See* Section VIII.A.8.ii.

9. Dependent Claims 9 and 21

- i. **[9pre/21pre] A method/device according to claim 6/18,**

216. Gai renders obvious claims 6 and 18. *See* Section VIII.B.7.

- ii. **[9a/21a] [wherein sending the data frames comprises incorporating]/[wherein the protocol processor is adapted to incorporate] a Virtual Local Area Network (VLAN) tag in a header of the data frames.**

217. Gai discloses this limitation. *See* Section VIII.A.9.ii.

10. Dependent Claims 10 and 22

- i. **[10pre/22pre] A method/device according to claim 6/18,**

218. Gai renders obvious claims 6 and 18. *See* Section VIII.B.7.

- ii. **[10a/22a] [wherein sending the data frames comprises multicasting]/[wherein the protocol processor is adapted to multicast] the data frames to all of the one or more access switches.**

219. Gai discloses this limitation. *See* Section VIII.A.10.ii.

11. Dependent Claims 11 and 23

- i. **[11pre/23pre] A method/device according to claim 1/13,**

220. Gai renders obvious claims 1 and 13. *See* Sections VIII.B.1 and

VIII.B.2.

- ii. **[11a/23a] wherein [sending the dummy traffic comprises sending data frames downstream]/[the dummy traffic comprises data frames sent downstream] from the edge switch toward the access switches.**

221. Gai renders obvious this limitation for the reasons I discussed in Sections VIII.B.1.ii and VIII.B.1.vi.

12. Dependent Claims 16 and 19

- i. **[16pre/19pre] A device according to claim 13/18, and**

222. Gai renders obvious claims 13 and 18. *See* Sections VIII.B.2 and VIII.B.7.

- ii. **[16a/19a] comprising a memory,**

223. Gai discloses this limitation. *See* Section VIII.B.8.ii.

- iii. **[16b/9b] 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,**

224. Gai discloses this limitation. *See* Section VIII.A.12.iii.

- iv. **[16c] 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.**

225. Gai discloses and/or renders obvious this limitation for the reasons I discussed in Section VIII.A.5.ii.

**v. [19c] to copy the source addresses for the data frames
from the edge switch database.**

226. Gai discloses and/or renders obvious this limitation for the reasons I discussed in Section VIII.B.8.ii.

IX. Conclusion

227. In my opinion, the challenged claims 1-2, 4-7, 9-14, 16-19, and 21-24 of the '986 patent are unpatentable.

X. Availability for Cross-Examination

228. In signing this declaration, I recognize that the declaration will be filed as evidence in a case before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I also recognize that I may be subject to cross examination in the case and that cross examination will take place within the United States. If cross examination is required of me, I will appear for cross examination within the United States during the time allotted for cross examination.

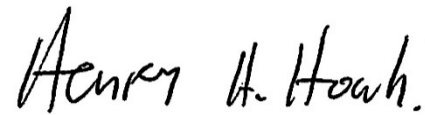
XI. Right to Supplement

229. I reserve the right to supplement my opinions in the future to respond to any arguments that the Patent Owner may raise and to take into account new information as it becomes available to me.

XII. Jurat

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

DATE: January 30, 2024

A handwritten signature in black ink that reads "Henry H. Houh." The signature is written in a cursive style with a horizontal line underneath it.

Dr. Henry Houh