

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CISCO SYSTEMS INC.,

Petitioner

IPR2023-00537
U.S. Patent No. 7,916,630

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

Ex.1003

TABLE OF CONTENTS

I.	INTRODUCTION	4
II.	QUALIFICATIONS AND PROFESSIONAL EXPERIENCE	5
III.	LEVEL OF ORDINARY SKILL IN THE ART	9
IV.	RELEVANT LEGAL STANDARDS	10
V.	OVERVIEW OF THE '630 PATENT	11
VI.	CLAIM CONSTRUCTION	16
VII.	IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE ...	16
VIII.	SILVA RENDERS OBVIOUS CLAIMS 1-12, 14, AND 15	17
A.	Summary of Silva.....	17
B.	Detailed Analysis of Claims.....	21
1.	Claim 1	21
2.	Claim 2.....	40
3.	Claim 3.....	44
4.	Claim 4.....	50
5.	Claim 5.....	54
6.	Claim 6.....	55
7.	Claim 7.....	59
8.	Claim 8.....	67
9.	Claim 9.....	67
10.	Claim 10.....	69

11.	Claim 11.....	70
12.	Claim 12.....	75
13.	Claim 14.....	76
14.	Claim 15.....	79
IX.	CONCLUSION	79

I, Nader F. Mir, Ph.D., do hereby declare as follows:

I. INTRODUCTION

1. I am making this declaration at the request of Cisco Systems, Inc. in the matter of the *Inter Partes* Review of U.S. Patent No. 7,916,630 (“the ’630 patent”) to Berndt et al.

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

3. I have been asked to provide my opinions regarding whether the subject matter of claims 1-12, 14, and 15 (“the Challenged Claims”) of the ’630 patent would have been obvious to a person having ordinary skill in the art (“POSITA”) at the time of the alleged invention, in light of the prior art. It is my opinion that the Challenged Claims would have been obvious to a POSITA.

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

- a. the ’630 patent, Ex. 1001;
- b. the prosecution history of the ’630 patent (“’630 File History”), Ex. 1002; and
- c. Silva et al., “DIP: Distributed Diagnosis Protocol,” *Microprocessing and Microprogramming* 38 (1993), Ex. 1005.

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

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

II. QUALIFICATIONS AND PROFESSIONAL EXPERIENCE

7. My complete qualifications and professional experience are described in my *Curriculum Vitae*, a copy of which can be found in Exhibit 1004. The following is a brief summary of my relevant qualifications and professional experience.

8. I am a tenured professor in the Electrical Engineering Department of San Jose State University, California, where I also held the position of Associate Chair from 2006 through 2008. I have been a faculty member at San Jose State University since 2001.

9. In 1995, I received a Ph.D. in electrical engineering from Washington University, St. Louis. I received an M.S. degree in electrical engineering in 1990, also from Washington University. In 1985, I received a B.S. in electrical engineering from Polytechnic University.

10. I have over 30 years of experience with computer networks and communication systems, including both industry and academic experience. Before my current position at San Jose State University, I was an Assistant Professor in the Department of Electrical and Computer Engineering at the University of Kentucky. I joined San Jose State University in 2001 as an Associate Professor. Since 2005, I have been a Full Professor in the Electrical Engineering Department at San Jose State University, where I perform research and teach in the areas of computer networks, networking devices, switches, routers, virtualization, virtual switches, cloud computing, software-defined networking in clouds, and cloud data centers.

11. I hold several technical editorial positions for various journals, including IEEE Communication Magazine. As a Technical Editor of IEEE Communications Standards Magazine for Cloud and Edge Computing, I am responsible for accepting or rejecting scientific articles submitted to the journal in the areas of computer networking focusing on cloud systems. From 2005 through 2020, I held the technical editorial position for IEEE Communications Magazine. I am a senior member of the IEEE and have served as a member of the technical program committees and the steering committees for a number of major IEEE communications and networking conferences.

12. For more than 30 years, I have conducted research in the areas of computer and communication networks; networking devices; protocols including (but not limited to) packet-switched networks, cloud and edge computing, software-defined networking in cloud systems, integrated voice, video, data center networks, TCP/IP, client/server operations, content delivery networks, multimedia and streaming, client/server, networking traffic flow analysis, wireless/mobile networks, networking devices, network security, and network virtualization, among others.

13. I have received a number of university awards as well as national and international awards. In particular, I have received a number of research grants from private, state, and governmental funding agencies for conducting research in the fields of computers and communication networks. I am also the recipient of several university teaching recognition awards from San Jose State University; several research excellence awards; and the school's published book award for the year.

14. Before my positions in academia, I worked in various positions in the telecommunication and computer network industries. From 1985 to 1988, I was a Telecomm R&D Engineer at the Telecommunication Research & Development Center (TRDC) in Surrey, England. In particular, I headed engineering projects in the area of telecommunication engineering. I also participated in the design of a

high-speed switch for digital Private Branch Exchange (PBX) and SS7 networks.

In 1991, I worked as an Electrical Engineering Consultant for Bussmann Company, where I instructed engineers on electrical engineering topics and prepared them for the National Principles and Practice of Engineering (PE) exam.

From 1989 to 1995, I worked as a Computer/Communications Chip-Design Engineer at the Computer & Communications Research Center in St. Louis, Missouri. In that position, I designed a VLSI gigabit switch control plane chip for a high-speed computer network using CMOS technology. From 1995 to 1996, I was a Research Scientist Engineer at Advanced Telecommunications Institute, Stevens Institute of Technology in Hoboken, New Jersey. There I headed analyzing communications and high-speed computer networking projects.

15. Since obtaining my Ph.D., I have served in various consulting positions in the area of computer networking technology and communication engineering, including for CITS Inc. in San Jose, California (2012-2013), Morrison & Foerster LLP in San Francisco, California (2015-2016), and Butler and Snow in Jackson, Mississippi (2017-2018).

16. I have authored or co-authored over 100 articles in peer-reviewed journals, conference proceedings, texts, and monographs in the aforementioned areas of computer networking, communications, and related fields. I have

published two textbooks, one of which is entitled “Computer Communication Networks.” This textbook has been adopted by numerous universities worldwide.

17. I am a named inventor on one U.S. patent, which relates to a packet-switching network and method of routing packets.

18. I have served as an expert witness and technical consultant in litigation and *inter partes* review matters concerning computer networks, wireless networks at both the protocol and hardware levels, and telecommunications.

III. LEVEL OF ORDINARY SKILL IN THE ART

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

20. A POSITA in the field of the '630 patent, as of its priority date of September 16, 2004, would have been someone knowledgeable and familiar with monitoring system conditions of a network with distributed components. Such a POSITA would have a bachelor's degree in computer science, computer engineering, electrical engineering, or equivalent training, and approximately two years of experience working in the field of communication networks. Additional work experience can substitute for specific educational background, and vice versa.

21. For purposes of this Declaration, in general, and unless otherwise noted, my statements and opinions, such as those regarding my own experience and what a POSITA would have understood or known generally (and specifically related to the references I consulted herein), reflect the knowledge that existed in the relevant field as of the priority date of the '630 patent.

IV. RELEVANT LEGAL STANDARDS

22. I am not an attorney. In preparing and expressing my opinions and considering the subject matter of the '630 patent, I am relying on certain basic legal principles that Cisco's counsel has explained to me.

23. I understand that prior art to the '630 patent includes printed publications in the relevant art that predate the priority date of the '630 patent. For purposes of this Declaration, I am applying September 16, 2004, as the priority date of the '630 patent.

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

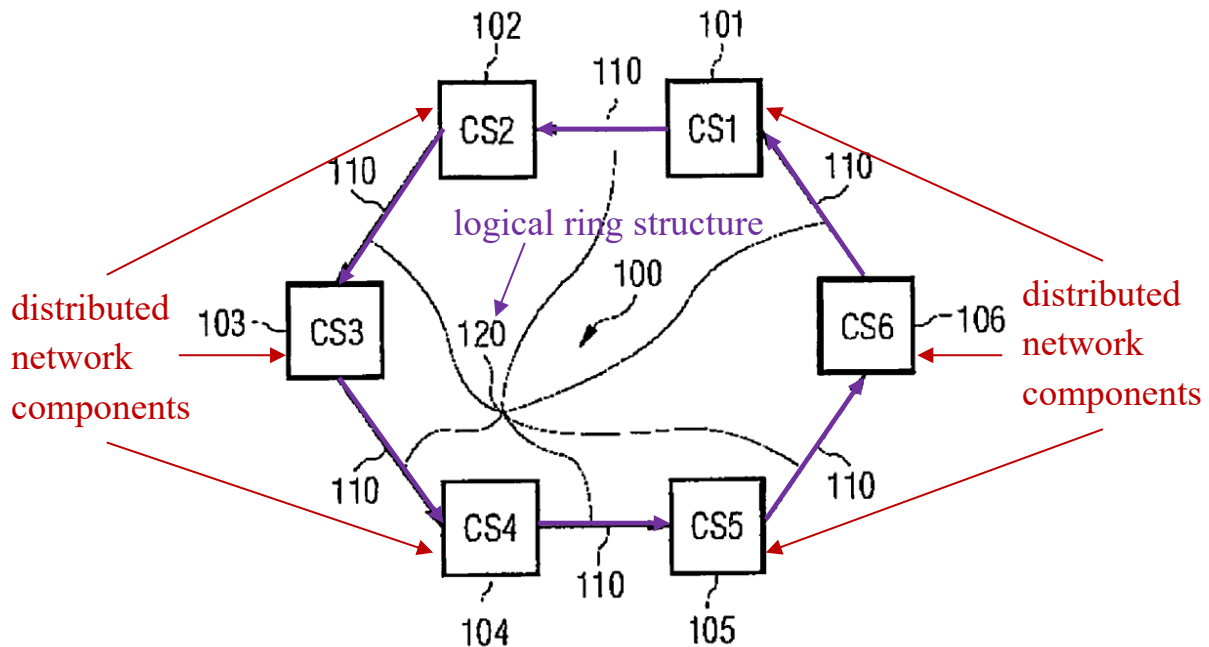
25. I have been further informed by Cisco’s counsel that there are several recognized rationales for combining references or modifying a reference to show obviousness. These rationales include: (a) combining prior art elements according to known methods to yield predictable results; (b) simple substitution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

26. Also, I have been informed and understand that obviousness does not require physical combination/bodily incorporation, but rather consideration of what the combined teachings would have suggested to a POSITA at the time of the alleged invention.

V. OVERVIEW OF THE ’630 PATENT

27. The ’630 patent relates generally to monitoring a “system condition of a system with distributed components.” Ex.1001, 1:14-17. According to the ’630 patent, distributed network components may be “organized in a logical ring structure.” Ex.1001, 2:36-39. This “logical” ring structure is “a conceptual model,

in a virtual ring structure,” as opposed to a “physical concrete structuring of the system components into a ring structure.” Ex.1001, 2:36-45. Figure 1 of the '630 patent shows distributed network components, such as communication servers (101-106), that are arranged in a logical ring structure (120):



Ex.1001 at Fig. 1 (annotated)

28. According to the '630 patent, “there is as a rule a requirement for all the components of the distributed system to have a knowledge of the status or condition of every other component in the system (monitoring).” Ex.1001, 1:18-23. To reduce the volume of system monitoring messages, “a component only monitors its respective neighboring component in the logical ring structure, whereby a condition of the respective neighboring component can be determined.” Ex.1001, 3:8-11.

Figure 1 “shows a distributed system with (distributed) components organized in a logical ring structure, in which each component only monitors its respective neighbor in the ring, according to an exemplary embodiment.” Ex.1001, 4:48-51. The neighboring component “can be a predecessor or also a successor in the logical ring structure.” Ex.1001, 4:30-32.

29. Each network component may monitor its predecessor or successor according to a “predefinable condition,” Ex.1001, 3:12-16, such as “a functional incapacity, in particular an offline condition, or a functional capacity, in particular an online condition.” Ex.1001, 3:38-41. “The monitoring of the respective neighboring component and/or the determination of the condition of a component can preferably be carried out using a method based on a leasing method.” Ex.1001, 3:43-46. Specifically, “‘Alive’ information, particularly an ‘Alive’ message, can be sent or is sent from the neighboring component to the component” monitoring it. Ex.1001, 3:47-50. “On the basis of such ‘Alive’ information, which may be periodic but is not necessarily so, it is then possible to determine a functional incapacity of the neighboring component if the neighboring component does not send (no longer sends) ‘Alive’ information. This can clearly mean that – on the other hand – as long as ‘Alive’ information is being sent, an ‘online’ condition can be assumed for the respective component.” Ex.1001, 3:54-61.

30. If a predefinable condition is determined to exist, the network component that detected the condition may inform the other components using a method based on an “Inform All” method. Ex.1001, 3:62-65. “Here, in other words with regard to the method based on an ‘Inform All’ method applied, an acknowledgment process is performed for each other component in the system, in which the respective other component, if it has received the information about the predefinable condition of a neighboring component, confirms the receipt of the information, in particular by using ‘Acknowledgment’ information, specifically an ‘Acknowledgment’ message.” Ex.1001, 3:66-4:6. If another component does not acknowledge the receipt of the information, “the predefinable condition is also determined for [the] another component.” Ex.1001, 4:14-18.

31. Figure 2, shows an embodiment of the invention where successor component 102 periodically sends an “Alive” message 110 to the preceding component 103 in the logical ring. Ex.1001, 5:30-33. “If in the case of one communication server (103 in FIG. 2) this ‘Alive’ message 110 from its successor (102 in FIG. 2) is absent (the ‘lease’ expires), the successor (102 in FIG. 2) will be categorized as ‘offline’.” Ex.1001, 5:34-37. “The communication server 103 monitoring this ‘offline’ communication server 102 then informs ever other communication server 101, 104 to 106 in the system 100 about this by corresponding

VI. CLAIM CONSTRUCTION

34. It is my understanding that in order to properly evaluate the '630 patent, the terms of the claims must first be interpreted. It is my understanding that for the purposes of this *inter partes* review, the claims are to be construed under the so-called *Phillips* standard, under which claim terms are given their ordinary and customary meaning as would have been understood by one of ordinary skill in the art in light of the specification and prosecution history, unless the inventor has set forth a special meaning for a term. I have also been informed that claim terms only need to be construed to the extent necessary to resolve the obviousness inquiry.

35. I have reviewed the entirety of the '630 Patent as well as its prosecution history. It is my opinion that for purposes of applying the prior art presented herein to evaluate the patentability of the claims no term requires express construction. Accordingly, I have analyzed the claims consistent with their ordinary and customary meaning as would have been understood by a POSITA.

VII. IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE

36. The discussion in this Declaration provides a detailed analysis of how the asserted prior art references teach each limitation of the Challenged Claims.

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

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

VIII. SILVA RENDERS OBVIOUS CLAIMS 1-12, 14, AND 15

A. Summary of Silva

39. Silva relates to a “DIP - a Diagnosis Protocol to check the availability of processors in a distributed system.” Ex.1005, 171. Like the ’630 patent, “network nodes are organized on a virtual ring and all the sites are numbered from 0 up to N-1, along the path of the virtual ring.” Ex.1005, 172. The virtual ring “does not reflect the physical organization of the network” but follows the logical “order of the Status Table, which is unique to all the processors in the network.” Ex.1005, 172. Figure 1 of Silva shows distributed nodes that are arranged in a virtual ring:

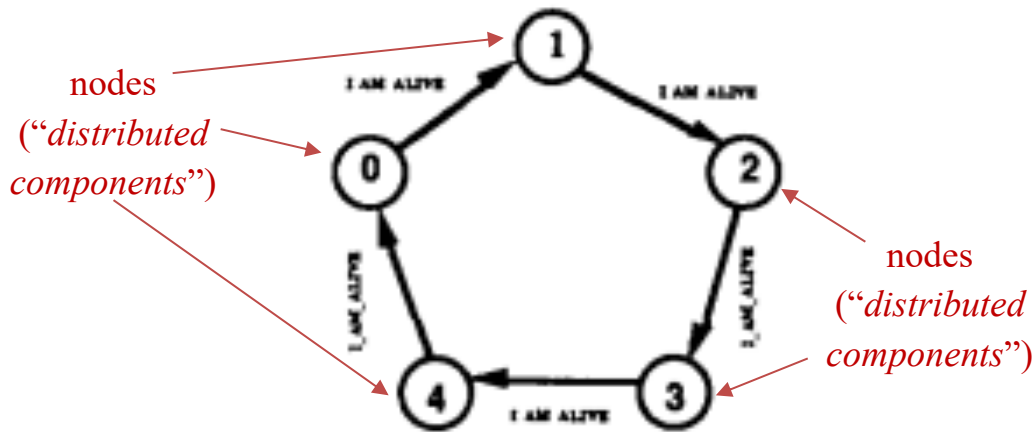


Figure 1 : Virtual Ring

Ex.1005 at Fig. 1 (annotated)

40. According to Silva, “[e]ach node maintains a table with the status of the other nodes, called the Network Status Table (NST.)” Ex.1005, 172. Like the ’630 patent, “to reduce the overhead during normal operation,” Silva teaches a Distributed Diagnosis Protocol that requires that each node monitor its respective neighbor in the ring. Ex.1005, 172. In particular, “[e]ach processor P_i sends periodically an I_AM_ALIVE message to the next neighbor $P_{((i+1)\text{mod}N)}$ while it is checking for the messages sent by the previous processor $P_{((i-1)\text{mod}N)}$.” Ex.1005, 172. “Each processor has two independent processes to perform that job. 1) The *Heartbeat* process is periodically awakened by a timer interrupt in order to send a message to the next neighbour ... 2) The detection of failures is done by the Watchdog process, which is listening to the messages sent by the previous neighbor.” Ex.1005, 172-173.

41. In particular, a node detects failures of its previous neighbor by listening for its I_AM_ALIVE messages. “Every time it receives an I_AM_ALIVE message it triggers a timer T_W with $((\Omega+1).\alpha + \partial_{\max} - \partial_{\min})$, where Ω is the number of omission failures or lost messages that can be tolerated. If the timer expires, this means that the previous node has crashed and the Node Down Protocol is started for that node.”

Ex.1005, 173.

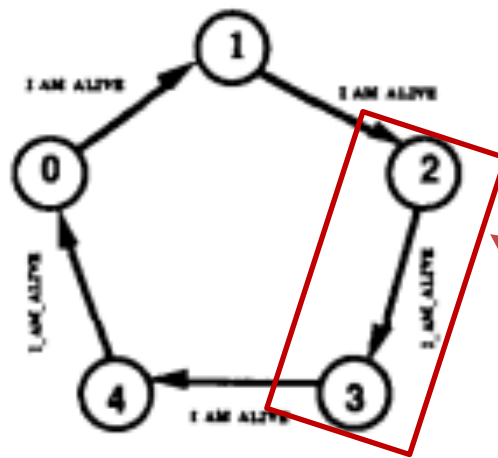


Figure 1 : Virtual Ring

Example:

Node P_3 checks for Node P_2 's I_AM_ALIVE message. A failure of Node P_2 is detected if its I_AM_ALIVE message is not received by Node P_3 before timer T_W expires.

Ex.1005 at Fig. 1 (annotated)

42. Like the '630 patent, if a node detects a failure, “the node should notify all the others immediately by a broadcast message to the network, announcing the Node_Down event of its neighbour.” Ex.1005, 173.

43. “Take as an example the network of figure 2. One can notice that node #2 has crashed and its tester node (node #3) will detect that when its timer T_W

expires. The status of node #2 is marked as DOWN in its Network Status Table, and it will send a broadcast message to the network, which includes the number of the node and the NST(2).IN: NODE_DOWN (2,1)." Ex.1005, 173.

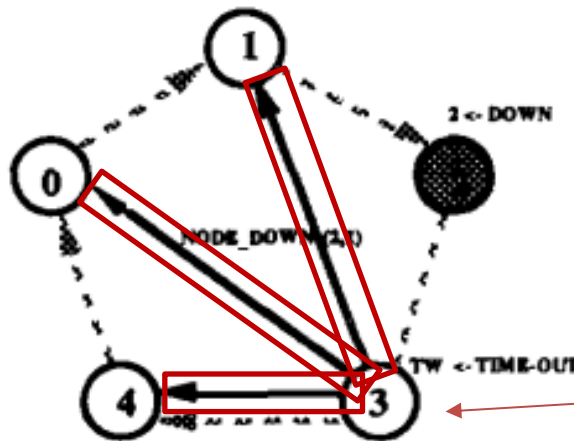


Figure 2 : Detection of a Failure

Example:
Node P₃ detects Node P₂ has crashed, and broadcasts a NODE_DOWN (2,1) message to the other nodes in the network.

Ex.1005 at Fig. 2 (annotated)

44. "[W]hen a node receives a NODE_DOWN broadcast it replies with a ND_Ack message to node #3." Ex.1005, 173. In certain cases, "some nodes may not answer" the NODE_DOWN broadcast message with a ND_Ack message, such as in the case of a network partition. Ex.1005, 173. Nodes that do not answer the NODE_DOWN broadcast with an acknowledgement message "should also be classified as DOWN." Ex.1005, 173.

B. Detailed Analysis of Claims

1. Claim 1

- a. [1.0] *A method for monitoring a system condition of a network with distributed components organized in a logical ring structure, comprising:*

45. To the extent that the preamble is limiting, Silva renders this limitation obvious.

46. First, Silva teaches a distributed diagnosis protocol for detecting node failures or crashes (“*system condition*”) in a network:

The DIP protocol should be able to perform three functions: (1) **To detect node crashes.** (2) To notify all the network nodes of the crash of a processor (**Node Down event**). (3) and also to notify all the nodes of the restart of a crashed node (Node Up event).

Ex.1005, 172.

This paper presents the DIP – a Diagnosis Protocol **to check the availability of processors** in a distributed system. Each node maintains a Network Status Table, **and failure detection is achieved** by alive messages. The overhead of such messages is minimal and the main virtue of the protocol is the lower diagnosis latency when a failure is detected. Since the unavailability of a node can be the result of a processor crash or a network partition, the DIP protocol is also able to cope with partitions. The protocol can diagnose fail-stop and omission failures.

Moreover, it is robust against out-of-order messages, duplicates and false alarms.

Ex.1005, Abstract.

Since failures are the exception rather than the rule, the first imperative is to reduce the overhead during normal operation. The protocol must assure a minimal number of messages during steady-state operation. Another important goal is to reduce the diagnosis latency to a minimal value, i.e., **the protocol must provide timely failure detection**. This is a paramount figure of merit to allow an effective recovery/reconfiguration of distributed applications. **When a node failure occurs all the other nodes must be acquainted with that as quickly as possible**. As failures and restarts are asynchronous events, the protocol should be robust against their occurrence during its execution. Other important issue is the handling of network partitions. The other desired goals are efficiency, simplicity and correctness of the protocol.

Ex.1005, 172.

47. Second, Silva teaches that the distributed diagnosis protocol detects failures of remote nodes (“*distributed components*”) that communicate with each other over a network:

DIP: **Distributed** Diagnosis Protocol.

Ex.1005, Title.

One of the requirements of a fault-tolerant distributed system is the failure detection capability of remote sites. Recovery and reconfiguration are other desirable properties. Since the paper of Preparata, Metze and Chien [1] there were further theoretical developments [2] in the area of distributed diagnosis. As a result, several PMC-based diagnosis algorithms have been suggested [3-9]. Recently, Bianchini et al. [10,11] have conducted the first implementations of system diagnosis theory in a real distributed network environment. Nevertheless, all those PMC-based algorithms have serious drawbacks: 1) when the number of node failures exceeds the connectivity of the system digraph, those algorithms simply yield false diagnostic results. 2) The PMC-model is only valid for permanent processor failures and it assumes that a node cannot fail and restart in the period between two tests. 3) Most of the algorithms require a high number of diagnostic messages and, as a result, the execution time may exceed reasonable values.

Ex.1005, 171.

The system consists of a set of N stations interconnected by a LAN with broadcast capability. Each station has access to stable storage which can be used to store information that must survive to failures. In perfect conditions each processor can communicate with any other processor in the network. We assume that the

relation “can communicate” between nodes is symmetric and transitive. Two kinds of failures are considered: processor failures and communication link failures. Processor failures are assumed to be clean. When a node fails it simply stops running and does not send any arbitrary or malicious messages. Thus the universe of failures includes stop and omission failures and no assumption is made on the number of allowable processor failures.

Ex.1005, 171-172.

48. Figure 1 depicts the remote network nodes communicating across a network by sending I_AM_ALIVE messages:

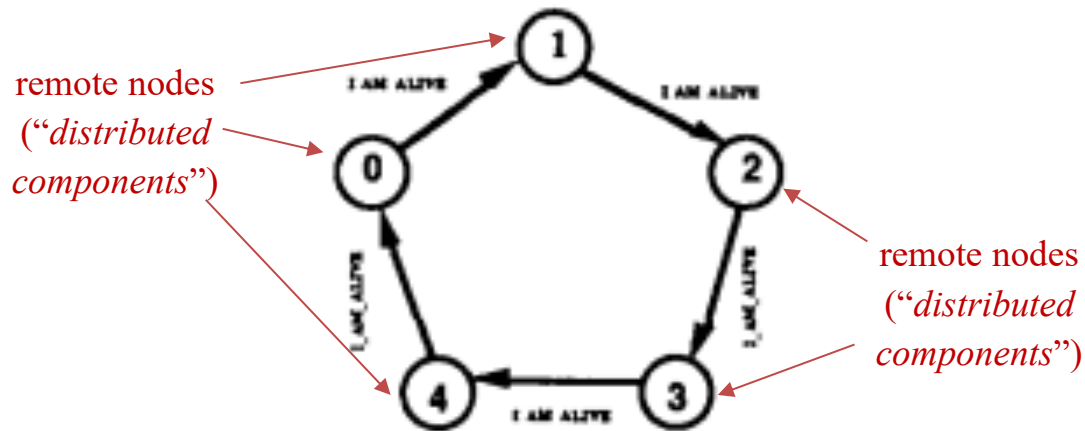


Figure 1 : Virtual Ring

Ex.1005, Fig. 1 (annotated)

49. **Third**, Silva teaches that the remote network nodes are arranged in a virtual ring (“logical ring”) structure:

DIP: Distributed Diagnosis Protocol.

Ex.1005, Title.

As a fail-silent processor only stops when an internal fault occurs the failure can be simply detected by background I_AM_ALIVE messages. **The network nodes are organized on a virtual ring and all the sites are numbered from 0 up to N-1, along the path of the virtual ring. It should be noted that the ring does not reflect the physical organization of the network. The order of the nodes in the ring follows the order of the Status Table,** which is unique to all the processors in the network.

Ex.1005, 172.

The algorithm presented by Walter [22] is similar to **the DIP – it also organizes the network as a virtual ring of processors.** But, one may notice that it is not robust against lost messages, duplicates, out of order messages and false alarms, since it assumes a reliable network. The critical message path of the diagnosis is 3, while the critical path of the DIP is just 1, which means that the diagnosis latency is at least 3 times higher than the latency of DIP. During steady-state operation the RSC protocol always requires the same number of messages: P.N.

Ex.1005, 177.

50. Figure 1 depicts the remote nodes arranged in a virtual ring structure:

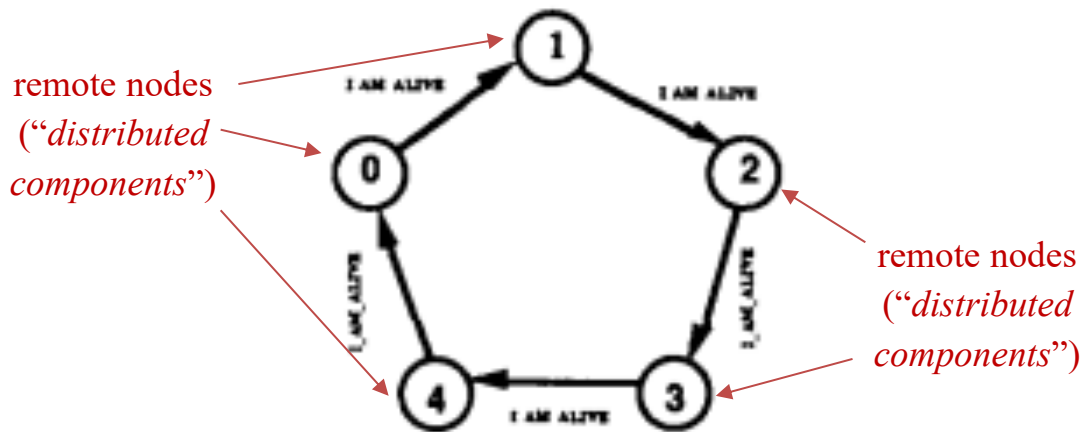


Figure 1 : Virtual Ring

Ex.1005, Fig. 1 (annotated)

51. A POSITA would have understood that Silva’s “virtual ring” structure is a “*logical ring*” structure because it reflects the logical organization of nodes in the ring, rather than the physical organization of the network.

52. Further, a POSITA would have understood Silva’s “virtual ring” structure to be a “*logical ring*” structure because one example of a logical ring structure provided by the ’630 Patent is a “virtual ring” structure:

In this situation, a “**logical**” ring structure (of system components) means that without restricting the generality no physical concrete structuring of the system components into a ring structure takes place as a rule but that these are organized or structured in this manner conceptually, in

other words as a conceptual mode, in **a virtual ring structure.**

Ex.1001, 2:40-45.

53. Therefore, Silva's disclosure of a distributed diagnosis protocol for detecting failures of remote network nodes arranged in a virtual ring, renders obvious a "*method for monitoring a system condition of a network with distributed components organized in a logical ring structure.*"

- b.** [1.1] *each component in the system monitoring only a single respective neighboring component among said distributed components that is a predecessor or successor of said each component in the logical ring structure to determine a current condition of the respective neighboring component; and*

54. Silva renders this limitation obvious.

55. First, Silva teaches that each node in the virtual ring monitors only its neighboring predecessor node ("*a single respective neighboring component*") in the virtual ring for an I_AM_ALIVE message.

Each processor P_i sends periodically an I_AM_ALIVE message to the next neighbour $P_{((i+1)\bmod N)}$ while it is checking for the messages sent by the previous processor $P_{((i-1)\bmod N)}$.

Ex.1005, 172.

As a fail-silent processor only stops when an internal fault occurs the failure can be simply detected by

background I_AM_ALIVE messages. The network nodes are organized on a virtual ring and all the sites are numbered from 0 up to N-1, along the path of the virtual ring. It should be noted that the ring does not reflect the physical organization of the network. The order of the nodes in the ring follows the order of the Status Table, which is unique to all the processors in the network.

Ex.1005, 172.

56. A POSITA would understand that the expression, $P_{((i-1)\bmod N)}$, refers to the network node immediately preceding network node P_i . In particular, the variable “i” indicates the identity of a particular node, the variable “N” indicates the total number of nodes in the virtual ring, and “mod” is the mathematical operation, modulo, that indicates the remainder after dividing (i-1) with N. Thus, for example, in a virtual ring with 5 nodes, network node P_2 monitors P_1 because $P_{((2-1)\bmod 5)} = P_1$.

57. Figure 1 depicts how network node P_i checks for an I_AM_ALIVE message sent from only its neighboring processor node $P_{((i-1)\bmod N)}$. For example, in Figure 1, network node P_2 checks for an I_AM_ALIVE message sent from only network node P_1 .

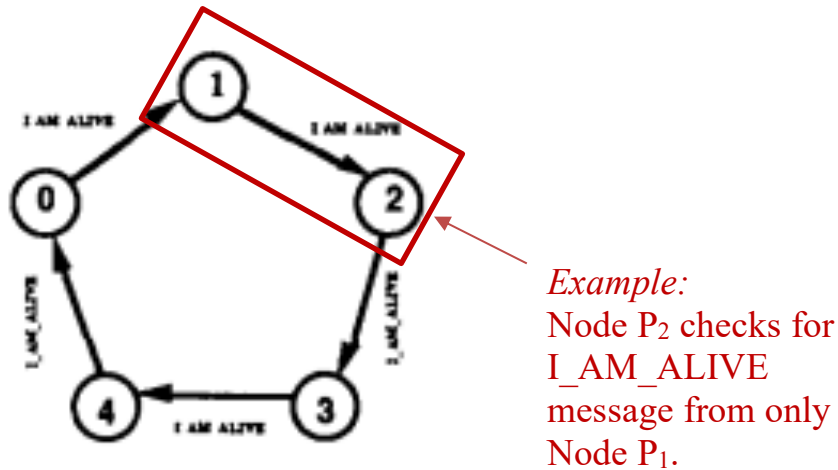


Figure 1 : Virtual Ring

Ex.1005 at Fig. 1 (annotated)

Each processor has two independent processes to perform that job.

- 1) The Heartbeat process is periodically awakened by a timer interrupt in order to send a message to the next neighbour. This timer is presented by T_H and is set to α .
- 2) The detection of failures is done by the Watchdog process, which is **listening to the messages sent by the previous neighbour**. Every time it receives an I_AM_ALIVE message it triggers a timer T_W with $((\Omega+1).\alpha + \partial_{\max} - \partial_{\min})$, where Ω is the number of omission failures or lost messages that can be tolerated. If the timer expires, this means that the previous node has crashed and the Node Down Protocol is started for that node.

Ex.1005, 172-173.

58. Second, Silva teaches that each node in the virtual ring determines whether its neighboring predecessor node is down or alive (“*determine a current condition*”) using the Watchdog process that detects the absence or presence of an I_AM_ALIVE message from its neighboring predecessor node. A node is down when an I_AM_ALIVE message is not received before a timer, T_W , expires. If an I_AM_ALIVE message is received from the neighboring predecessor node before the timer expires, the timer is reset and the Node Down Protocol is not started because the node is alive.

Each processor P_i sends periodically an I_AM_ALIVE message to the next neighbour $P_{((i+1)\bmod N)}$ while it is **checking for the messages sent by the previous processor $P_{((i-1)\bmod N)}$** . Each processor has two independent processes to perform that job.

- 1) The Heartbeat process is periodically awakened by a timer interrupt in order to send a message to the next neighbour. This timer is presented by T_H and is set to α .
- 2) **The detection of failures is done by the Watchdog process, which is listening to the messages sent by the previous neighbour. Every time it receives an I_AM_ALIVE message it triggers a timer T_W with $((\Omega+1)\cdot\alpha + \hat{o}_{\max} - \hat{o}_{\min})$, where Ω is the number of omission failures or lost messages that can be tolerated. If the timer expires, this means that the previous**

node has crashed and the Node Down Protocol is started for that node.

Ex.1005, 172-173.

In steady-state operation, the protocol requires the minimum number of diagnostic messages. In a network of N active processors, N messages are sent to the network between each testing interval. **When a node failure occurs, it is detected by its neighbour** in a finite interval called failure detection latency. This latency depends on how much time has elapsed between the time the last I_AM_ALIVE message was sent and the failure. In the worst case, the failure occurs immediately after the last message is sent out, while in the best case the node fails just before sending the message. So, the detection latency falls between $(\partial_{\max})$ and the period of the Watchdog Timer: $((\Omega+1).\alpha + \partial_{\max} - \partial_{\min})$.

Ex.1005, 176.

In order to reduce the diagnosis latency, the node should notify all the others immediately by a broadcast message to the network, announcing the **Node_Down event of its neighbour**.

Ex.1005, 173.

Take as an example the network of figure 2. One can notice that node #2 has crashed and its tester (node #3) will detect that when its timer T_W expires. **The status of node**

#2 is marked as DOWN in its Network Status Table, and it will send a broadcast message to the network, which includes the number of the node and the NST(2).IN **NODE_DOWN (2,1)**.

Ex.1005, 173.

59. Therefore, Silva's disclosure of each node monitoring only its neighboring predecessor node in the virtual ring for the absence or presence of an I_AM_ALIVE message, to detect whether a node is down or alive using a Watchdog process, renders obvious "*each component in the system monitoring only a single respective neighboring component among said distributed components that is a predecessor or successor of said each component in the logical ring structure to determine a current condition of the respective neighboring component.*"

- c. **[1.2] *each component in the system informing all other components of the system about the current condition of the respective neighboring component when the current condition corresponds to at least one predefined condition.***

60. Silva renders this limitation obvious.

61. First, as discussed in [1.1], Silva teaches that each node in the virtual ring determines that a neighboring predecessor node is down when its Watchdog process does not detect an I_AM_ALIVE message before a timer, T_w , expires ("at

least one predefined condition”). Silva refers to such condition as a “Node Down” condition:

Each processor P_i sends periodically an I_AM_ALIVE message to the next neighbour $P_{((i+1)\bmod N)}$ while it is checking for the messages sent by the previous processor $P_{((i-1)\bmod N)}$. Each processor has two independent processes to perform that job.

- 1) The Heartbeat process is periodically awakened by a timer interrupt in order to send a message to the next neighbour. This timer is presented by T_H and is set to α .
- 2) The detection of failures is done by the Watchdog process, which is listening to the messages sent by the previous neighbour. Every time it receives an I_AM_ALIVE message it triggers a timer T_W with $((\Omega+1).\alpha + \hat{o}_{\max} - \hat{o}_{\min})$, where Ω is the number of omission failures or lost messages that can be tolerated. **If the timer expires, this means that the previous node has crashed and the Node Down Protocol is started** for that node.

Ex.1005, 172-173.

In order to reduce the diagnosis latency, the node should notify all the others immediately by a broadcast message to the network, announcing the **Node_Down event of its neighbour**.

Ex.1005, 173.

Take as an example the network of figure 2. One can notice that node #2 has crashed and its tester (node #3) will detect that when its timer T_W expires. **The status of node #2 is marked as DOWN** in its Network Status Table, and it will send a broadcast message to the network, which includes the number of the node and the NST(2).IN **NODE_DOWN** (2,1).

Ex.1005, 173.

62. Second, a POSITA would understand that Silva's Node Down condition is "*predefined*" because its requirements are known prior to detecting a node is actually down. In particular, a Node Down condition requires that no I_AM_ALIVE message is received before Watchdog Timer, T_W , expires.

Each processor P_i sends periodically an I_AM_ALIVE message to the next neighbour $P_{((i+1)\bmod N)}$ while it is checking for the messages sent by the previous processor $P_{((i-1)\bmod N)}$. Each processor has two independent processes to perform that job.

- 1) The Heartbeat process is periodically awakened by a timer interrupt in order to send a message to the next neighbour. This timer is presented by T_H and is set to α .
- 2) **The detection of failures is done by the Watchdog process**, which is listening to the messages sent by the

previous neighbour. Every time it receives an I_AM_ALIVE message it triggers a timer T_W with $((\Omega+1).\alpha + \partial_{\max} - \partial_{\min})$, where Ω is the number of omission failures or lost messages that can be tolerated. **If the timer expires, this means that the previous node has crashed and the Node Down Protocol is started for that node.**

Ex.1005, 172-173.

Take as an example the network of figure 2. One can notice that node #2 has crashed and its tester (node #3) will detect that **when its timer T_W expires. The status of node #2 is marked as DOWN** in its Network Status Table, and it will send a broadcast message to the network, which includes the number of the node and the NST(2).IN **NODE_DOWN (2,1).**

Ex.1005, 173.

63. Figure 2 depicts node #3 has detected that node #2 has crashed because node #3 did not receive an I_AM_ALIVE message from node #2 before node #3's Watchdog Timer, T_W , expired:

previous neighbour. Every time it receives an I_AM_ALIVE message it triggers a timer T_W with $((\Omega+1).\alpha + \partial_{\max} - \partial_{\min})$, where Ω is the number of omission failures or lost messages that can be tolerated. **If the timer expires, this means that the previous node has crashed and the Node Down Protocol is started for that node.**

Ex.1005, 173.

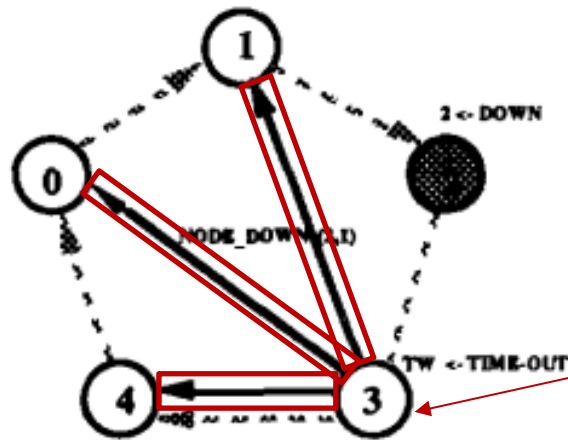
In order to reduce the diagnosis latency, the node should **notify all the others immediately** by a broadcast message to the network, **announcing the Node_Down event of its neighbour.**

Ex.1005, 173.

Take as an example the network of figure 2. One can notice that node #2 has crashed and its tester (node #3) will detect that when its timer T_W expires. **The status of node #2 is marked as DOWN in its Network Status Table, and it will send a broadcast message to the network,** which includes the number of the node and the NST(2).IN: **NODE_DOWN (2,1).**

Ex.1005, 173.

65. Figure 2 depicts node P_3 notifying all other nodes (P_0 , P_1 , and P_4) about the failure of neighboring predecessor node P_2 , by sending the other nodes a **NODE_DOWN (2,1)** message:



Example:
Node P₃ notifies all other nodes in the virtual ring about the Node Down condition of Node P₂ by broadcasting a NODE_DOWN(2,1) message.

Figure 2 : Detection of a Failure

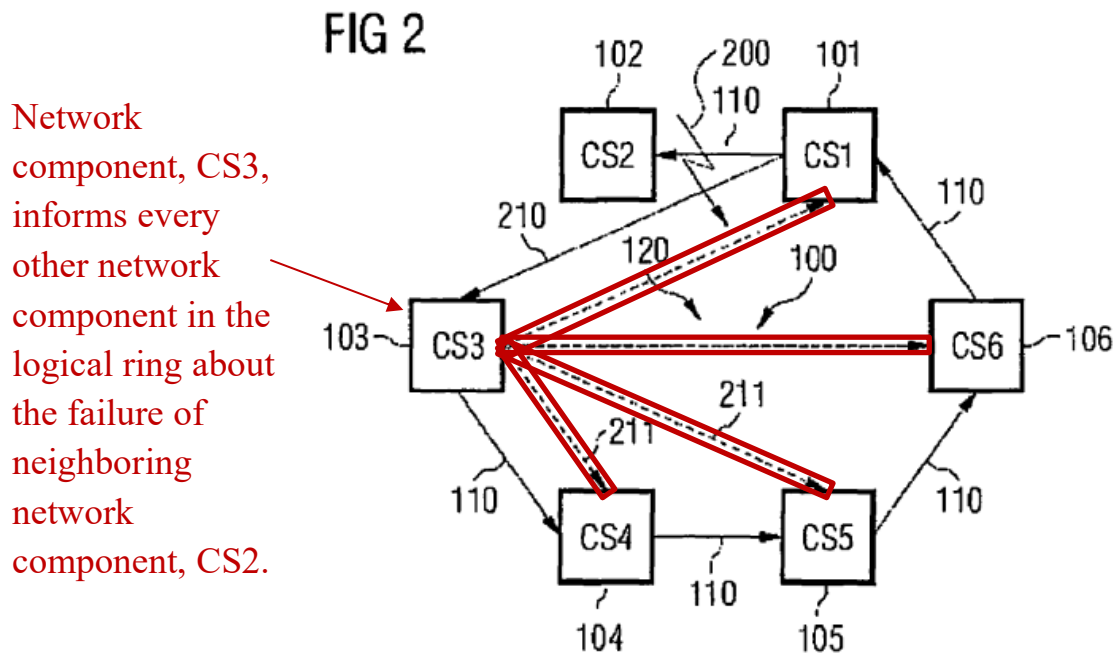
Ex.1005 at Fig. 2 (annotated)

The detection of failures is done by the Watchdog process, which is listening to the messages sent by the previous neighbour. Every time it receives an I_AM_ALIVE message it triggers a timer T_W with $((\Omega+1).\alpha + \partial_{\max} - \partial_{\min})$, where Ω is the number of omission failures or lost messages that can be tolerated. **If the timer expires, this means that the previous node has crashed and the Node Down Protocol is started for that node.**

In order to reduce the diagnosis latency, **the node should notify all the others immediately by a broadcast message to the network, announcing the Node_Down event of its neighbour.**

Ex.1005, 173.

66. The broadcast message (e.g. NODE_DOWN (2,1)) sent to each node in Silva is much like the information message (211) sent to each network component described in the '630 patent to notify all other nodes that a failure is detected. Figure 2 depicts an example of a network component (i.e., CS3) notifying all other network components (i.e., CS1, CS4, CS5, CS6) in the virtual ring that it has detected a failure of its neighboring networking component (i.e., CS2).



Ex.1001 at Fig. 2 (annotated)

The communication server 103 monitoring this 'offline' communication server 102 then informs every other communication server 101, 104 to 106 in the system 100 about this by corresponding information messages 211, in other words about the failure 200 of its successor 102.

Ex.1001, 5:30-53.

67. Therefore, Silva's disclosure of a Watchdog process in each node to detect a Node Down condition of its neighboring predecessor node, and initiating a Node Down Protocol that broadcasts a message to notify all other nodes about the Node Down condition, renders obvious "*each component in the system informing all other components of the system about the current condition of the respective neighboring component when the current condition corresponds to at least one predefined condition.*"

2. Claim 2

[2.0] *The method as claimed in claim 1, wherein the at least one predefined condition is at least one of a functional incapacity corresponding to an offline condition and a functional capacity corresponding to an online condition.*

68. Silva renders this limitation obvious.

69. As discussed in [1.2], Silva teaches a Watchdog process that detects a Node Down condition ("*predefined condition*"). The Node Down condition is a node failure or crash condition ("*a functional incapacity corresponding to an offline condition*") when an I_AM_ALIVE message from its neighboring predecessor node is not received within a predefined period of the Watchdog Timer, T_W :

Each processor P_i sends periodically an I_AM_ALIVE message to the next neighbour $P_{((i+1)\bmod N)}$ while it is checking for the messages sent by the previous processor

$P_{((i-1) \bmod N)}$. Each processor has two independent processes to perform that job.

- 1) The Heartbeat process is periodically awakened by a timer interrupt in order to send a message to the next neighbour. This timer is presented by T_H and is set to α .
- 2) **The detection of failures is done by the Watchdog process, which is listening to the messages sent by the previous neighbour. Every time it receives an I_AM_ALIVE message it triggers a timer T_W with $((\Omega+1)\alpha + \partial_{\max} - \partial_{\min})$, where Ω is the number of omission failures or lost messages that can be tolerated. If the timer expires, this means that the previous node has crashed and the Node Down Protocol is started for that node.**

Ex.1005, 172-173.

In steady-state operation, the protocol requires the minimum number of diagnostic messages. In a network of N active processors, N messages are sent to the network between each testing interval. **When a node failure occurs, it is detected by its neighbour** in a finite interval called failure detection latency. This latency depends on how much time has elapsed between the time the last I_AM_ALIVE message was sent and the failure. In the worst case, the failure occurs immediately after the last message is sent out, while in the best case the node fails

just before sending the message. So, the detection latency falls between $\partial_{\max}$ and the period of the Watchdog Timer: $((\Omega+1).\alpha t + \partial_{\max} - \partial_{\min})$.

Ex.1005, 176.

Take as an example the network of figure 2. **One can notice that node #2 has crashed and its tester (node #3) will detect that when its timer T_W expires. The status of node #2 is marked as DOWN** in its Network Status Table, and it will send a broadcast message to the network, which includes the number of the node and the NST(2).IN NODE_DOWN (2,1).

Ex.1005, 173.

70. A POSITA would have understood that a node “failure” and a node “crash” are each examples of “*a functional incapacity corresponding to an offline condition,*” because a node that has failed or crashed is unavailable to other nodes for communication.

71. Silva’s teaching of detecting a Node Down condition when an I_AM_ALIVE message from a neighboring predecessor node is not received within a predefined period, is much like the example of a “*functional incapacity*” in the ’630 patent specification described as an “*offline condition*” that is determined when a neighboring component no longer sends a periodic “Alive” message.

In a development, provision is made whereby the predefinable condition is **a functional incapacity, in particular an offline condition**, or a functional capacity, in particular an online condition. **The term “alive” has become widely accepted among experts for the functional capacity.**

The monitoring of the respective neighboring component and/or the determination of the condition of a component can preferably be carried out using a method based on a leasing method.

In this situation, in other words with regard to a leasing method, “Alive” information, particularly an “Alive” message, can be sent or is sent from the neighboring component to the component.

It can be advantageous here, particularly in respect of current information about a system condition, to send the “Alive” information periodically.

On the basis of such “Alive” information, which may be periodic but is not necessarily so, **it is then possible to determine a functional incapacity of the neighboring component if the neighboring component does not send (no longer sends) “Alive” information. This can clearly mean—on the other hand—as long as “Alive” information is being sent, an “online” condition can be assumed for the respective component.**”

Ex.1001, 3:38-61.

72. Thus, Silva renders obvious “*wherein the at least one predefined condition is at least one of a functional incapacity corresponding to an offline condition and a functional capacity corresponding to an online condition.*”

3. Claim 3

[3.0] *The method as claimed in claim 2, wherein at least one of said monitoring of the respective neighboring component and determination of the current condition of the respective neighboring component is carried out based on a leasing method.*

73. Silva renders this limitation obvious.

74. **First**, as discussed in [1.1], Silva teaches that each node listens for I_AM_ALIVE messages from only its neighboring predecessor node (“*said monitoring of the respective neighboring component*”) to determine whether or not the predecessor node has failed (“*said determination of the current condition of the respective neighboring component*”).

75. **Second**, as discussed in [1.1] and [1.2], each node sends periodically to its neighboring successor an I_AM_ALIVE message using a Heartbeat process, and the neighboring successor listens for the I_AM_ALIVE message using a Watchdog process.

Each processor P_i sends periodically an I_AM_ALIVE message to the next neighbour $P_{((i+1)\bmod N)}$ while it is checking for the messages sent by the previous processor

$P_{((i-1) \bmod N)}$. Each processor has two independent processes to perform that job.

- 1) **The Heartbeat process** is periodically awakened by a timer interrupt in order to send a message to the next neighbour. This timer is presented by T_H and is set to α .
- 2) The detection of failures is done by the **Watchdog process**, which is listening to the messages sent by the previous neighbour. Every time it receives an I_AM_ALIVE message it triggers a timer T_W with $((\Omega+1) \cdot \alpha + \partial_{\max} - \partial_{\min})$, where Ω is the number of omission failures or lost messages that can be tolerated. If the timer expires, this means that the previous node has crashed and the Node Down Protocol is started for that node.

Ex.1005, 172-173.

76. Every time the neighboring successor node receives an I_AM_ALIVE message, the neighboring successor node's Watchdog Timer, T_W , is triggered. If the Watchdog Timer, T_W , expires then a failure of the neighboring predecessor is detected and it is marked as DOWN ("*current condition*"). If an I_AM_ALIVE message is received before Watchdog Timer, T_W , expires, then no crash or failure is detected and the Watchdog Timer, T_W , is re-triggered (or reset).

Each processor P_i sends periodically an I_AM_ALIVE message to the next neighbour $P_{((i+1) \bmod N)}$ while it is

checking for the messages sent by the previous processor $P_{((i-1) \bmod N)}$. Each processor has two independent processes to perform that job.

- 1) **The Heartbeat process is periodically awakened by a timer interrupt in order to send a message to the next neighbour.** This timer is presented by T_H and is set to α .
- 2) **The detection of failures is done by the Watchdog process, which is listening to the messages sent by the previous neighbour. Every time it receives an I_AM_ALIVE message it triggers a timer T_W with $((\Omega+1).\alpha + \partial_{\max} - \partial_{\min})$, where Ω is the number of omission failures or lost messages that can be tolerated. If the timer expires, this means that the previous node has crashed and the Node Down Protocol is started for that node.**

Ex.1005, 172-173.

Take as an example the network of figure 2. One can notice that node #2 has crashed and its tester (**node #3**) **will detect that when its timer T_W expires.** The status of node #2 is marked as DOWN in its Network Status Table, and **it will send a broadcast message to the network,** which includes the number of the node and the NST(2).IN: **NODE_DOWN (2,1).**

Ex.1005, 173.

As a fail-silent processor only stops when an internal fault occurs the failure can be simply detected by background I_AM_ALIVE messages. The network nodes are organized on a virtual ring and all the sites are numbered from 0 up to N-1, along the path of the virtual ring. It should be noted that the ring does not reflect the physical organization of the network. The order of the nodes in the ring follows the order of the Status Table, which is unique to all the processors in the network.

Ex.1005, 172.

In steady-state operation, the protocol requires the minimum number of diagnostic messages. In a network of N active processors, N messages are sent to the network between each testing interval. **When a node failure occurs, it is detected by its neighbour** in a finite interval called failure detection latency. This latency depends on how much time has elapsed between the time the last I_AM_ALIVE message was sent and the failure. In the worst case, the failure occurs immediately after the last message is sent out, while in the best case the node fails just before sending the message. So, the detection latency falls between $\partial_{\max}$ and the period of the Watchdog Timer: $((\Omega+1).\alpha + \partial_{\max} - \partial_{\min})$.

Ex.1005, 176.

77. Silva's teaching of marking a node as DOWN if a neighboring successor node does not receive an I_AM_ALIVE message before a Watchdog Timer, T_W , expires, is much like the example of a "*leasing method*" described in the '630 patent specification where a network component is categorized as "offline" if its neighboring node does not receive an "Alive" message within a certain period:

The monitoring 110 of a successor server 101 to 106 in the ring structure 120 takes place here, in other words in the case of the HiPath IP telephony system 100 further developed as described herein, **by way of a "leasing" method.**

In this situation, a successor, component 102 for example, periodically sends a so-called "Alive" message 110 to the respective preceding communication server in the ring, component 103 according to the example.

If in the case of one communication server (103 in FIG. 2) this "Alive" message 110 from its successor (102 in FIG. 2) is absent (the "lease" expires), the successor (102 in FIG. 2) will be categorized as "offline".

Ex.1001, 5:26-37.

The monitoring of the respective neighboring component and/or the determination of the condition of a component

can preferably be **carried out using a method based on a leasing method.**

In this situation, in other words with regard to **a leasing method, “Alive” information, particularly an “Alive” message, can be sent or is sent from the neighboring component to the component.**

It can be advantageous here, particularly in respect of current information about a system condition, **to send the “Alive” information periodically.**

On the basis of such “Alive” information, which may be periodic but is not necessarily so, it is then possible to determine a functional incapacity of the neighboring component if the neighboring component does not send (no longer sends) “Alive” information. This can clearly mean—on the other hand—as long as “Alive” information is being sent, an “online” condition can be assumed for the respective component.”

Ex.1001, 3:43-61.

78. Thus, like the leasing method described in the '630 Patent, if an I_AM_ALIVE message is not received or “is absent” from the neighboring predecessor node for a certain period, the “lease” expires and the neighboring node will be categorized as down. *See* Ex.1001, 5:34-37. If an I_AM_ALIVE message is received from the neighboring predecessor node, then the lease does not expire, no crash or failure is detected, and an “online” condition can be assumed.

79. Accordingly, a POSITA would have recognized that the interaction between the Heartbeat process in each node for sending I_AM_ALIVE messages, and the Watchdog process in a neighboring successor node for listening for these messages to determine whether its neighboring predecessor node has crashed or not, teaches a “*leasing method*.”

80. Therefore, Silva’s disclosure of a Heartbeat process in each node for sending I_AM_ALIVE messages periodically to a neighboring successor node, and a Watchdog process in each neighboring successor node to listen for these messages (“*monitoring of the respective neighboring component*”) before a leasing period expires, to determine whether or not a respective predecessor node has failed (“*determination of the current condition of the respective neighboring component*”), renders obvious “*wherein at least one of said monitoring of the respective neighboring component and determination of the current condition of the respective neighboring component is carried out based on a leasing method*.”

4. Claim 4

[4.0] *The method as claimed in claim 3, wherein with regard to the leasing method, an “Alive” message is sent from the respective neighboring component.*

81. Silva renders this limitation obvious.

82. As discussed in [1.1] and [3.0], Silva teaches that each node in the virtual ring monitors only its neighboring predecessor node in the virtual ring for an I_AM_ALIVE message to determine whether it is down or alive.

As a fail-silent processor only stops when an internal fault occurs the failure can be simply detected by background I_AM_ALIVE messages. The network nodes are organized on a virtual ring and all the sites are numbered from 0 up to N-1, along the path of the virtual ring. It should be noted that the ring does not reflect the physical organization of the network. The order of the nodes in the ring follows the order of the Status Table, which is unique to all the processors in the network.

Ex.1005, 172.

Each processor P_i sends periodically an I_AM_ALIVE message to the next neighbour $P_{((i+1) \bmod N)}$ while it is checking for the messages sent by the previous processor $P_{((i-1) \bmod N)}$. Each processor has two independent processes to perform that job.

- 1) The Heartbeat process is periodically awakened by a timer interrupt in order to send a message to the next neighbour.** This timer is presented by T_H and is set to α .
- 2) The detection of failures is done by the Watchdog process, which is listening to the messages sent by the previous neighbour. Every time it receives an**

I_AM_ALIVE message it triggers a timer T_W with $((\Omega+1).\alpha + \partial_{\max} - \partial_{\min})$, where Ω is the number of omission failures or lost messages that can be tolerated. If the timer expires, this means that the previous node has crashed and the Node Down Protocol is started for that node.

Ex.1005, 172-173.

In steady-state operation, the protocol requires the minimum number of diagnostic messages. In a network of N active processors, N messages are sent to the network between each testing interval. **When a node failure occurs, it is detected by its neighbour** in a finite interval called failure detection latency. This latency depends on how much time has elapsed between the time the last I_AM_ALIVE message was sent and the failure. In the worst case, the failure occurs immediately after the last message is sent out, while in the best case the node fails just before sending the message. So, the detection latency falls between $\partial_{\max}$ and the period of the Watchdog Timer: $((\Omega+1).\alpha + \partial_{\max} - \partial_{\min})$.

Ex.1005, 176.

83. Silva's teaching of sending an I_AM_ALIVE message from a respective neighboring component for determining whether the component is down or alive, is much like the example of an "Alive" message in the '630 patent specification describing how an "Alive" message that may be sent periodically

from a network component to its neighbor for determining a functional incapacity of the network component.

In this situation, a successor, component 102 for example, periodically **sends a so-called “Alive” message 110** to the respective preceding communication server in the ring, component 103 according to the example.

If in the case of one communication server (103 in FIG. 2) this “Alive” message 110 from its successor (102 in FIG. 2) is absent (the “lease” expires), the successor (102 in FIG. 2) will be categorized as “offline”.

Ex.1001, 5:30-37.

In a development, provision is made whereby the predefinable condition is a functional incapacity, in particular an offline condition, or a functional capacity, in particular an online condition. **The term “alive” has become widely accepted among experts for the functional capacity.**

The monitoring of the respective neighboring component and/or the determination of the condition of a component can preferably be carried out using a method based on a leasing method.

In this situation, in other words with regard to a leasing method, **“Alive” information, particularly an “Alive” message, can be sent or is sent from the neighboring component to the component.**

It can be advantageous here, particularly in respect of current information about a system condition, to send the “Alive” information periodically.

On the basis of such “Alive” information, which may be periodic but is not necessarily so, it is then possible to determine a functional incapacity of the neighboring component if the neighboring component does not send (no longer sends) “Alive” information. This can clearly mean—on the other hand—as long as “Alive” information is being sent, an “online” condition can be assumed for the respective component.”

Ex.1001, 3:38-61.

84. Therefore, Silva’s disclosure of each node sending its neighboring successor node an I_AM_ALIVE message for detecting whether the node is down or alive, renders obvious “*wherein with regard to the leasing method, an ‘Alive’ message is sent from the respective neighboring component.*”

5. Claim 5

[5.0] *The method as claimed in claim 4, wherein the “Alive” information is sent periodically.*

85. Silva renders this limitation obvious.

86. As described in [4.0], Silva teaches that the Heartbeat process is periodically awakened by a timer interrupt in order to send the “I_AM_ALIVE” message to the next neighbor.

Each processor P_i sends periodically an I_AM_ALIVE message to the next neighbour $P_{((i+1)\bmod N)}$ while it is checking for the messages sent by the previous processor $P_{((i-1)\bmod N)}$. Each processor has two independent processes to perform that job.

- 1) The Heartbeat process is periodically awakened by a timer interrupt in order to send a message to the next neighbour.** This timer is presented by T_H and is set to α .
- 2) The detection of failures is done by the Watchdog process, which is listening to the messages sent by the previous neighbour.** Every time it receives an I_AM_ALIVE message it triggers a timer T_W with $((\Omega+1).\alpha + \partial_{\max} - \partial_{\min})$, where Ω is the number of omission failures or lost messages that can be tolerated. If the timer expires, this means that the previous node has crashed and the Node Down Protocol is started for that node.

Ex.1005, 172-173.

87. Therefore, Silva renders obvious the “*method as claimed in claim 4, wherein the ‘Alive’ information is sent periodically.*”

6. Claim 6

[6.0] *The method as claimed in claim 5, wherein the functional incapacity of the neighboring component is determined if the respective neighboring component does not*

send any “Alive” information for a predetermined period of time.

88. Silva renders this limitation obvious.

89. **First**, as described in [1.2], Silva teaches that the Watchdog process detects a Node Down condition (“*current condition*” is “*functional incapacity*”) when an I_AM_ALIVE message (“*any ‘Alive’ information*”) from its neighboring predecessor node is not received before the expiration of the Watchdog Timer, T_W (“*for a predetermined period of time*”).

90. **Second**, Silva teaches that Watchdog Timer, T_W , has a “*predetermined*” time period according to the mathematical equation, $T_W = ((\Omega+1).\alpha + \partial_{\max} - \partial_{\min})$:

Each processor P_i sends periodically an I_AM_ALIVE message to the next neighbour $P_{((i+1)\bmod N)}$ while it is checking for the messages sent by the previous processor $P_{((i-1)\bmod N)}$. Each processor has two independent processes to perform that job.

- 1) The Heartbeat process is periodically awakened by a timer interrupt in order to send a message to the next neighbour. This timer is presented by T_H and is set to α .
- 2) The detection of failures is done by the Watchdog process, which is listening to the messages sent by the previous neighbour. Every time it receives an I_AM_ALIVE message it triggers a **timer T_W with**

$((\Omega+1).\alpha + \partial_{\max} - \partial_{\min})$, where Ω is the number of omission failures or lost messages that can be tolerated. If the timer expires, this means that the previous node has crashed and the Node Down Protocol is started for that node.

Ex.1005, 172-173.

91. Silva teaches that each variable of the mathematical formula, $T_W = ((\Omega+1).\alpha + \partial_{\max} - \partial_{\min})$, is determined prior to setting the timer T_W (“predetermined”).

Each processor P_i sends periodically an I_AM_ALIVE message to the next neighbour $P_{((i+1)\bmod N)}$ while it is checking for the messages sent by the previous processor $P_{((i-1)\bmod N)}$. Each processor has two independent processes to perform that job.

- 1) The Heartbeat process is periodically awakened by a timer interrupt in order to send a message to the next neighbour. This timer is presented by T_H and is **set to α** .
- 2) The detection of failures is done by the Watchdog process, which is listening to the messages sent by the previous neighbour. Every time it receives an I_AM_ALIVE message it triggers a **timer T_W with $((\Omega+1).\alpha + \partial_{\max} - \partial_{\min})$, where Ω is the number of omission failures or lost messages that can be tolerated**. If the timer expires, this means that the

previous node has crashed and the Node Down Protocol is started for that node.

Ex.1005, 172-173.

We consider that there exists a **maximum message delay** $\hat{d}_{\max}$, which reflects the expected maximum transmission time of a single message between two nodes, including the processing delay. The minimum message delay is represented by $\hat{d}_{\min}$. Every node should have a physical clock in order to implement timers but we do not require clocks to be synchronized.

Ex.1005, 172.

The first release of the DIP was implemented on a set of SUN workstations which communicate through an Ethernet local area network. It was written in C using sockets UDP/IP. The protocol itself was running as a background process. The initialization module stores in stable storage the list of the processors in the network. This list is sorted by the Internet host address. As this number is unique to each processor it assures that all the hosts generate identical lists. All the crash and recovery events are stored in the Crash_Log file, which can be inspected for failure analysis. A query interface is also available to users asking for the network status. **The parameters chosen were the following : $\alpha = 1$ second and $\Omega = 2$.**

Ex.1005, 177.

92. A POSITA would have thus found it obvious that each of the parameters α , Ω , $\partial_{\max}$, and $\partial_{\min}$, in the mathematical formula, $T_W = ((\Omega+1).\alpha + \partial_{\max} - \partial_{\min})$ is “*predetermined*” because they are each determined prior to setting the timer, T_W . In particular, the parameter α (i.e., value of heartbeat interrupt timer, T_H) and the parameter Ω (i.e., number of omission failures or lost messages that can be tolerated) are each values “chosen” or “set” as part of the distributed diagnosis protocol (DIP). Ex.1005, 177. Additionally, the parameter $\partial_{\max}$ (i.e., expected maximum transmission time between two nodes) and the parameter $\partial_{\min}$ (i.e., minimum message delay between two nodes) are each operational characteristics of the virtual ring network itself. Therefore, a POSITA would understand that each of the parameters α , Ω , $\partial_{\max}$, and $\partial_{\min}$, would have been determined or set before calculating the value of Watchdog Timer, T_W . Thus, the time period of Watchdog Timer, T_W , is “*predetermined*.”

93. Therefore, Silva renders obvious “*wherein the functional incapacity of the neighboring component is determined if the respective neighboring component does not send any ‘Alive’ information for a predetermined period of time.*”

7. Claim 7

[7.0] *The method as claimed in claim 6, wherein said informing all the other components about the predefined condition of the respective neighboring component is carried out using an “Inform All” method.*

94. Silva renders this limitation obvious.

95. *First*, as discussed in [1.2] and [2.0], Silva teaches that the Watchdog process detects a Node Down condition (“*predefined condition*”) corresponding to an “*offline condition*” when an I_AM_ALIVE message from its neighboring predecessor node is not received before Watchdog Timer, T_W , expires.

96. *Second*, as discussed in [1.2], Silva teaches that each node, upon detection of a Node Down condition of its neighboring predecessor node, initiates a Node Down Protocol to broadcast a “NODE_DOWN” message to each of the other nodes about the failure.

97. *Third*, Silva teaches that each other node that receives a “NODE_DOWN” message replies to the notifying node with an “acknowledge” (e.g. “ND_Ack”) message.

In a meantime, a timer T_{ack} will be set to $2 \cdot \partial_{max}$. **The node will inspect its status table to find the available nodes and waits for the acknowledge of every UP node. On the other hand, when a node receives a NODE_DOWN broadcast it replies with a ND_Ack message to node #3.** It should be noted that each acknowledge is also a broadcast message. This may be useful if a particular node does not receive the broadcast message of #3 it may be informed of that by one of the acknowledge messages. When node #3 receives all acknowledges this phase is finished and the protocol returns to the steady state. However, if the timer T_{ack} expires and some reply is

missing, the message `NODE_DOWN(2, 1)` is retransmitted again (until a maximum number of retries is reached).

Ex.1005, 173.

98. *Fourth*, Silva teaches that if an “acknowledge” message is not received by the node that broadcasted the “`NODE_DOWN`” message before a timer, T_{ack} , expires, the node whose acknowledgement is missing is classified as DOWN.

In a meantime, a timer T_{ack} will be set to $2 \cdot \partial_{max}$. The node will inspect its status table to find the available nodes and waits for the acknowledge of every UP node. On the other hand, when a node receives a `NODE_DOWN` broadcast it replies with a `ND_Ack` message to node #3. It should be noted that each acknowledge is also a broadcast message. This may be useful if a particular node does not receive the broadcast message of #3 it may be informed of that by one of the acknowledge messages. When node #3 receives all acknowledges this phase is finished and the protocol returns to the steady state. However, **if the timer T_{ack} expires and some reply is missing**, the message `NODE_DOWN(2, 1)` is retransmitted again (until a maximum number of retries is reached).

[...]

It may happen that the unavailability of node #2 is not the crash of a single processor, but instead the network

may be partitioned. In this case, **some nodes may not answer to the message and should also be classified as DOWN.** In fact, if after the retransmission, node #3 does not receive any reply then it will start again this protocol announcing the unavailability of those nodes which remained in the other partition of the network.

Ex.1005, 173.

99. *Fifth*, Silva teaches that after classifying a node whose acknowledgment is missing as DOWN, the node that broadcasted the “NODE_DOWN” message announces the unavailability of the DOWN node to each of the other nodes:

It may happen that the unavailability of node #2 is not the crash of a single processor, but instead the network may be partitioned. In this case, some nodes may not answer to the message and should also be classified as DOWN. **In fact, if after the retransmission, node #3 does not receive any reply then it will start again this protocol announcing the unavailability of those nodes which remained in the other partition of the network.**

Ex.1005, 173.

100. *Sixth*, it would have been an obvious implementation choice whether the node that announces the unavailability of the DOWN node to each of the other nodes may – or may not – expect any further acknowledgment messages. In one

implementation choice, the node announcing the unavailability of the DOWN node *may* expect further acknowledgment messages. Silva teaches that the node announcing the unavailability of a DOWN node may “start again” the protocol of announcing the unavailability of nodes which do not send an acknowledgment message:

It may happen that the unavailability of node #2 is not the crash of a single processor, but instead the network may be partitioned. In this case, some nodes may not answer to the message and should also be classified as DOWN. **In fact, if after the retransmission, node #3 does not receive any reply then it will start again this protocol announcing the unavailability of those nodes which remained in the other partition of the network.**

Ex.1005, 173.

101. Thus, a POSITA would have found an obvious implementation choice would be to repeat – or start again – the protocol announcing the unavailability of nodes that do not send acknowledgment messages to the broadcasted “NODE_DOWN” message, and to listen for subsequent acknowledgments to those announcements. Doing so would advantageously identify other DOWN nodes using further acknowledgment and notification messages.

102. Accordingly, a POSITA would have found it obvious from Silva’s teachings that the node that “announc[ed] the unavailability” of the nodes that did

not send an “acknowledgment” message may – or may not – expect any further acknowledgment messages.

103. Silva’s process for notifying all other nodes in the virtual ring about a node down condition, and determining whether any of the notified nodes are also down based on whether they send an acknowledgment messages, is much like the example of an “*Inform All*” method in the ’630 patent describing a network component that informs all other network components of an offline condition, waits for an acknowledgement message from the informed network components, and determines that a respective network component is offline if an acknowledgment message is not received from that node.

In addition, it is advantageous for the process of informing the other components about the predefinable condition of a neighboring component to be carried out using an method based on an **“Inform All” method**.

Here, in other words with regard to the method based on an “Inform All” method applied, **an acknowledgment process is performed for each other component in the system, in which the respective other component, if it has received the information about the predefinable condition of a neighboring component, confirms the receipt of the information, in particular by using “Acknowledgment” information, specifically an “Acknowledgment” message.** The term

“acknowledgment” has become widely accepted among experts for the “confirmation”.

In this situation, the acknowledgment (of receipt of the information) occurs as a rule with respect to the component which has determined the predefinable condition of the neighboring component, in particular by the fact that an “Acknowledgment” message is sent to the component.

In addition, provision can be made whereby the predefinable condition is also determined for another component which does not acknowledge the receipt of the information about the determined, defined condition of the neighboring component.

In this case too, all the other components can again be informed about this information. This can be done with or also without a corresponding acknowledgment procedure.

Ex.1001, 3:62-4:21.

In this situation, a successor, component 102 for example, periodically sends a so-called “Alive” message 110 to the respective preceding communication server in the ring, component 103 according to the example.

If in the case of one communication server (103 in FIG. 2) this “Alive” message 110 from its successor (102 in FIG.

2) is absent (the “lease” expires), the successor (102 in FIG. 2) will be categorized as “offline”.

The communication server 103 monitoring this “offline” communication server 102 then informs every other communication server 101, 104 to 106 in the system 100 about this by corresponding information messages 211, in other words about the failure 200 of its successor 102.

Every other communication server 101, 104 to 106 in the system 100 must confirm the receipt of the information by an “Acknowledgment” message.

If this acknowledgment is missing from one of the other components 101, 104 to 106, then this communication server will also be categorized as “offline”.

All the communication servers are in turn informed about this further failure by the communication server 103.

At this stage the communication server 103 does not however expect any further acknowledgment.

This mechanism is referred to as “Inform All”.

Ex.1001, 5:30-53.

104. Therefore, Silva renders obvious “*wherein said informing all the other components about the predefined condition of the respective neighboring component is carried out using an ‘Inform All’ method.*”

8. Claim 8

[8.0] *The method as claimed in claim 7, further comprising, with regard to the “Inform All” method, performing an acknowledgment process by each of the other components, in which each of the other components, if it has received information about the predefined condition of the respective neighboring component, confirms receipt of the information, by sending an “Acknowledgment” message.*

105. Silva renders this limitation obvious.

106. As discussed in [7.0] Silva teaches that each other node that receives a “NODE_DOWN” message replies with an “acknowledge” (e.g. “ND_Ack”) message. A POSITA would understand that an “acknowledge” and an “ND_Ack message” are each examples of an “Acknowledgment” message because they are sent to confirm receipt of the “NODE_DOWN” message.

9. Claim 9

[9.0] *The method as claimed in claim 8, wherein the “Acknowledgment” message is sent to the component which has determined the predefined condition of the respective neighboring component.*

107. Silva renders this limitation obvious.

108. As discussed in [7.0], Silva teaches that the acknowledgment message is sent to the node (e.g. node #3) that detected the Node_Down event.

Take as an example the network of figure 2. One can notice that node #2 has crashed and its tester **node will detect that when its timer T_W expires**. The status of node #2 is marked as DOWN in its Network Status Table, and

it will send a broadcast message to the network, which includes the number of the node and the NST(2).IN: NODE_DOWN (2,1).

Ex.1005, 173.

In a meantime, a timer T_{ack} will be set to $2 \cdot \hat{\sigma}_{max}$. **The node** will inspect its status table to find the available nodes and **waits for the acknowledge of every UP node. On the other hand, when a node receives a NODE_DOWN broadcast it replies with a ND_Ack message to node #3.** It should be noted that each acknowledge is also a broadcast message. This may be useful if a particular node does not receive the broadcast message of #3 it may be informed of that by one of the acknowledge messages. **When node #3 receives all acknowledges this phase is finished** and the protocol returns to the steady state. However, if the timer T_{ack} expires and some reply is missing, the message NODE_DOWN(2, 1) is retransmitted again (until a maximum number of retries is reached).

Ex.1005, 173.

109. Therefore, Silva's disclosure that the nodes that receive the NODE_DOWN message from the notifying node (e.g. node #3), also send an acknowledgment message to that node (e.g. node #3), renders obvious "*wherein*

the ‘Acknowledgment’ message is sent to the component which has determined the predefined condition of the respective neighboring component.”

10. Claim 10

[10.0] *The method as claimed in claim 9, further comprising determining the predefined condition for any component which does not acknowledge the receipt of the information about the predefined condition of the respective neighboring component.*

110. Silva renders this limitation obvious.

111. Silva teaches that as part of the “*Inform All*” method discussed in [7.0], if an acknowledgement message is not received by the notifying node (e.g. node #3) from a node that it informed, then the informed node is classified as Node_Down (“*the predefined condition*”).

In a meantime, a timer T_{ack} will be set to $2 \cdot \partial_{max}$. The node will inspect its status table to find the available nodes and waits for the acknowledge of every UP node. On the other hand, when a node receives a NODE_DOWN broadcast it replies with a ND_Ack message to node #3. It should be noted that each acknowledge is also a broadcast message. This may be useful if a particular node does not receive the broadcast message of #3 it may be informed of that by one of the acknowledge messages. When node #3 receives all acknowledges this phase is finished and the protocol returns to the steady state. However, **if the timer T_{ack} expires and some reply is missing, the message NODE**

DOWN(2,1) is re-transmitted again (until a maximum number of retries is reached).

Ex.1005, 173.

It may happen that the unavailability of node #2 is not the crash of a single processor, but instead the network may be partitioned. In this case, some nodes may not answer to the message and should **also be classified as DOWN**. In fact, if after the **retransmission, node #3 does not receive any reply then it will start again this protocol announcing the unavailability of those nodes** which remained in the other partition of the network.

Ex.1005, 173.

112. Therefore, Silva renders obvious the “*method as claimed in claim 9, further comprising determining the predefined condition for any component which does not acknowledge the receipt of the information about the predefined condition of the respective neighboring component.*”

11. Claim 11

[11.0] *The method as claimed in claim 10, further comprising in each component storing a local list of conditions of the other components.*

113. Silva renders this limitation obvious.

114. Silva teaches that each node stores a local list of conditions of the other nodes.

Each node maintains a **table with the status of the other nodes**, called the **Network Status Table (NST)**. Each entry of the NST has three fields: the host name, **its status** and the Incarnation Number (IN). The IN number is helpful to distinguish duplicates and out-of-order messages, as will be explained in §4.4.

Ex.1005, 172.

In order to reduce the diagnosis latency, the node should notify all others immediately by a broadcast message to the network, announcing the Node_Down event of its neighbour.

Take as an example the network of figure 2. One can notice that node #2 has crashed and its tester (node #3) will detect that when its timer T_W expires. **The status of node #2 is marked as DOWN in its Network Status Table**, and it will send a broadcast message to the network, which includes the number of the node and the NST(2).IN NODE_DOWN (2,1).

Ex.1005, 173.

When a node rejoins the network it must tell the others that it is back in operation and must also determine which nodes are UP and reachable. **Therefore, when a node recovers it reads the NST from stable storage to main memory**. The status of all the other nodes are set to DOWN. Then it increments its IN value, which is also

stored in stable storage, and sets to zero all the counters of the other nodes. Finally, it broadcasts a `NODE_UP` message to notify all the active nodes in the network. A timer T_{up} is activated. This message must be acknowledged by the other available nodes. **The replies will be used by the restarted node to achieve its new network view.** Furthermore, to guarantee a distributed consistency, node #2 must verify whether all the nodes it can reach share the same NST.

Ex.1005, 173-174.

When each of the other nodes receive the `NODE_UP (2,2)` message, they check the 2nd entry of its NST, and the status of node #2 is marked as UP. Then, they send a `ND_Ack` message to node #2, which contains their own Ins Each node also appends a list of its DOWN nodes into that message including the corresponding Ins. **When the timer T_{up} expires, node #2 will have a complete information about the network status.** Likewise, to assure distribution agreement, node #2 must check the information it had received from the `ND_Ack` messages. The DOWN lists which were piggybacked into the acknowledges must be all consistent. Otherwise, node #2 will have to broadcast the result of the comparison to force a distributed consensus.

Ex.1005, 174.

As the two sub-protocols may execute simultaneously they must interact with each other. Let us consider the following situation : node #2 has crashed; node #3 detects its failure and sends a NODE DOWN broadcast message to the network. However node #2 is DOWN for only a short time, after that it comes up again and rejoins the network by sending a NODE_UP message. Yet, an inconsistent scenario may occur: Node #0 receives the sequence (NODE_DOWN, NODE_UP) while node#1 receives the (NODE_UP,NODE_DOWN) sequence. Fortunately, the Incarnation Number can solve this inconsistency. **Every message is tagged with the IN and each receptive node will compare the incoming IN with the value stored in its NST. When a host receives a NODE UP message from node #K and the incoming IN exceeds NST(K).IN, the table is automatically updated.** If the incoming IN is similar it is recognized as a duplicate message and no action is performed. In either cases a reply (NU_Ack) is sent.

When a node receives a NODE_DOWN message from node #X notifying the crash of its neighbour (node #Y) it checks in its table the status and the current IN of node #Y. If #Y is UP and the incoming IN exceeds the NST(Y).IN, then #Y is marked as DOWN and the local IN is updated. In normal situations, the incoming IN is similar to the local value. **In this case, only the status of**

node #Y is updated. On the other hand, if #Y is already DOWN the NST(Y).IN will only be updated if the incoming IN exceeds the local IN. In either case the receiving node will reply to each message with its own IN. Taking the previous example, let us assume that node #2 was running with IN=1. By the time it had crashed, node #3 would have sent a NODE_DOWN message announcing this event (with IN=1). When the node recovers, it would increment IN to 2.

Node #0 would receive the messages in the correct order. However, when node #1 received the message NODE_UP (2,2) it would realize that node #2 had crashed and came up again without being aware of that. It would set NST(2).IN to 2 and send a NU_Ack message to node #2, which would include its own IN (NST(1).IN).

After having received the NODE_DOWN (2,1) message it would conclude that the message was out-of-date since the incoming IN was less than the stored one. In spite of that, a reply message (ND_Ack) would be sent to node #3.

Ex.1005, 174.

115. Therefore, Silva's disclosure that each node stores in its own storage or memory a table with the status of other nodes, renders obvious "*in each component storing a local list of conditions of the other components.*"

12. Claim 12

[12.0] *The method as claimed in claim 11, wherein the respective neighboring component is one of a predecessor component and a successor component in the logical ring structure.*

116. Silva renders this limitation obvious.

117. As discussed in [1.1], Silva teaches that each node in the virtual ring monitors only its neighboring predecessor node, such as the previous processor, $P_{((i-1)\bmod N)}$.

Each processor P_i sends periodically an I_AM_ALIVE message to the next neighbour $P_{((i+1)\bmod N)}$ while it is checking for the messages sent by the previous processor $P_{((i-1)\bmod N)}$. Each processor has two independent processes to perform that job.

Ex.1005, 172.

118. Figure 1 depicts how each node P_i (e.g. node P_2) monitors only its neighboring predecessor node $P_{((i-1)\bmod N)}$ (e.g. node P_1) for an I_AM_ALIVE message:

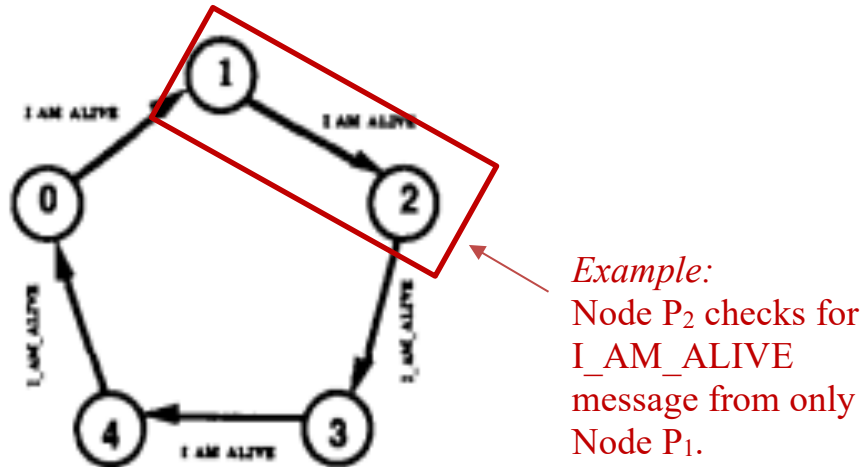


Figure 1 : Virtual Ring

Ex.1005 at Fig. 1 (annotated)

119. Therefore, Silva's disclosure of a node monitoring its predecessor by listening for its I_AM_ALIVE messages renders obvious "*wherein the respective neighboring component is one of a predecessor component and a successor component in the logical ring structure.*"

13. Claim 14

- a. [14.0] *At least one non-transitory computer-readable medium encoded with at least one computer program that when executed by at least one processor performs a method comprising:*

120. Silva renders this limitation obvious.

121. Silva teaches that each node in the distributed network has stable storage and a processor for executing the distributed diagnosis protocol.

The first release of the **DIP** was implemented on a set of **SUN workstations** which communicate through an Ethernet local area network. **It was written in C using sockets UDP/IP. The protocol itself was running as a background process. The initialization module stores in stable storage** the list of the processors in the network. This list is sorted by the Internet host address. As this number is unique to **each processor** it assures that all the hosts generate identical lists. All the crash and recovery events are stored in the Crash_Log file, which can be inspected for failure analysis. A query interface is also available to users asking for the network status. The parameters chosen were the following : $\alpha = 1$ second and $\Omega = 2$.

Ex.1005, 177.

The system consists of a set of N stations interconnected by a LAN with broadcast capability. **Each station has access to stable storage** which can be used to store information that must survive to failures. **In perfect conditions each processor can communicate with any other processor in the network.** We assume that the relation "can communicate" between nodes is symmetric and transitive. Two kinds of failures are considered: processor failures and communication link failures. Processor failures are assumed to be clean. When a node fails it simply stops running and does not send any

arbitrary or malicious messages. Thus, the universe of failures includes stop and omission failures and no assumption is made on the number of allowable processor failures.

Ex.1005, 171-172.

122. A POSITA would find it obvious that Silva's teaching of a distributed diagnosis protocol (DIP) written in C code would be stored in the stable storage of each network device such that when executed by the processor, carries out the operations of the distributed diagnosis protocol (DIP) because it was common for computer software written in C to be compiled into an executable file that is then stored on persistent computer media, such as a SUN workstation's hard drive.

123. Therefore, Silva renders obvious, "[a]t least one non-transitory computer-readable medium encoded with at least one computer program" executed by a processor.

b. [14.1] *each component in a logical ring structure*

124. See analysis at [1.0].

c. [14.2] *monitoring only a single neighboring component among distributed components that is a predecessor or successor of said each component in the logical ring structure to determine a current condition of the neighboring component; and*

125. See analysis at [1.1].

d. [14.3] *each component in the logical ring structure informing all other components of the system about the*

current condition of the neighboring component when the current condition corresponds to a predefined condition.

126. See analysis at [1.2].

14. Claim 15

a. ***[15.0] A network with distributed components, comprising:***

127. See analysis at [1.0].

b. ***[15.1] components organized in a logical ring structure,***

128. See analysis at [1.0].

c. ***[15.2] each component monitoring only a single respective neighboring component among components that is a predecessor or successor of said each component in the logical ring structure to determine a current condition of the respective neighboring component and***

129. See analysis at [1.1].

d. ***[15.3] informing all other components of said network when the current condition of the respective neighboring component corresponds to a predefined condition.***

130. See analysis at [1.2].

IX. CONCLUSION

131. I declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and that these statements were made with knowledge that willful false statements and

Declaration of Nader F. Mir, Ph.D.
Inter Partes Review of U.S. 7,916,630

the like so made are punishable by fine or imprisonment, or both, under section 1001
of Title 18 of the United States Code.

Dated: 2/7/2023


Nader Mir, Ph.D.