UNITED STATES PATENT AND TRADEMARK OFFICE

---

BEFORE THE PATENT TRIAL AND APPEAL BOARD

---

ServiceNow, Inc.
Petitioner

v.

BMC Software, Inc.
Patent Owner

U.S. Patent No. 6,816,898
Filing Date: August 16, 2000
Issue Date: November 9, 2004

TITLE: INTERFACING EXTERNAL METRICS INTO A PERFORMANCE MANAGEMENT SYSTEM

# DECLARATION OF TAL LAVIAN, PH.D.

## Table of Contents

**Page**

**Table of Contents**
(continued)

**Page**

**Table of Contents**
(continued)

**Page**

**Table of Contents**
(continued)

**Page**

I, Tal Lavian, Ph.D., declare as follows:

1. I have been retained by counsel for ServiceNow, Inc. ("Petitioner") in this case as an expert in the relevant art.

2. I have been asked to provide my opinions relating to claims 1-12 of U.S. Patent No. 6,816,898 to Joe Scarpelli et al. ("'898 patent"), which I understand is owned by BMC Software, Inc. ("Patent Owner" or "BMC").

## I. BRIEF SUMMARY OF MY OPINIONS

3. Claims 1-12 of the '898 patent, generally speaking, purport to disclose methods for collecting performance management data from monitored computers on a network and displaying that information. The data collection can be performed by a "script-based program." As the title of the '898 patent ("Interfacing External Metrics Into A Performance Management System") suggests, a purpose of the methods of the '898 patent is to facilitate collecting performance management data from external sources and incorporating that external data into a larger performance management system. The claims do not describe anything that was new or non-obvious by August 2000, the earliest date listed on the face of the '898 patent.

4. As explained in detail in **Part VI** of this Declaration, the features described in claims 1-7 and 9-12 were disclosed and/or suggested in a publication entitled "satool—A System Administrator's Cockpit, An Implementation," by Todd Miller et al. ("Miller"), which pre-dates the filing date of the '898 patent by almost seven years. In broad overview, Miller discloses a system called "satool" for monitoring and managing the performance of a large number of computers in a distributed environment. The satool system uses "helper scripts" that collect data from monitored computers and displays that data to a system administrator. A user of satool can extend the functionality of the system by writing his or her own helper scripts and integrating them into the satool system by following a few straightforward steps that Miller outlines. As with the '898 patent, a purpose of the "helper scripts" is to collect data that may not normally be integrated into a performance management system. Claim 8 is disclosed and/or suggested by Miller in view of Brian W. Kernighan et al., The C Programming Language (2d ed. 1988), and Tim O'Reilly et al., Windows 98 in a Nutshell (1999), which describe basic aspects of computer programming and administration.

## II. INTRODUCTION AND QUALIFICATIONS

### A. Qualifications and Experience

5. I possess the knowledge, skills, experience, training and the education to form an expert opinion and testimony in this case. A detailed record of my professional qualifications, including a list of patents and academic and professional publications, is set forth in my curriculum vitae attached to this declaration as **Exhibit A**.

6. I have more than 25 years of experience in the networking, telecommunications, Internet, and software fields. I received a Ph.D. in Computer Science from the University of California at Berkeley in 2006 and obtained a Master's of Science ("M.Sc.") degree in Electrical Engineering from Tel Aviv University, Israel, in 1996. In 1987, I obtained a Bachelor of Science ("B.Sc.") in Mathematics and Computer Science, also from Tel Aviv University.

7. I am currently employed by the University of California at Berkeley and was appointed as a lecturer and Industry Fellow in the Center of Entrepreneurship and Technology ("CET") as part of UC Berkeley College of Engineering. I have been with the University of California at Berkeley since 2000 where I served as Berkeley Industry Fellow, Lecturer, Visiting Scientist, Ph.D.

Candidate, and Nortel's Scientist Liaison, where some positions and projects were done concurrently, others sequentially.

8. I have more than 25 years of experience as a scientist, educator and technologist, and much of my experience relates to computer networking technologies. For eleven years from 1996 to 2007, I worked for Bay Networks and Nortel Networks. Bay Networks was in the business of making and selling computer network hardware and software. Nortel Networks acquired Bay Networks in 1998, and I continued to work at Nortel after the acquisition. Throughout my tenure at Bay and Nortel, I held positions including Principal Scientist, Principal Architect, Principal Engineer, Senior Software Engineer, and led the development and research involving a number of networking technologies. I led the efforts of Java technologies at Bay network and Nortel Networks. In addition, during 1999-2001, I served as the President of the Silicon Valley Java User Group with over 800 active members from many companies in the Silicon Valley.

9. Prior to that, from 1994 to 1995, I worked as a software engineer and team leader for Aptel Communications, designing and developing mobile wireless

devices and network software products. From 1990 to 1993, I worked as a software engineer and team leader at Scitex Ltd., where I developed system and network communications tools (mostly in C and C++).

10. I have extensive experience in the area of network communications and Internet technologies including design and implementation of computer-based systems for managing communications networks, including the ability to monitor and provision networks. While with Nortel Networks and Bay Networks (mentioned above) my work involved the research and development of these technologies. For example, I wrote software for Bay Networks and Nortel Networks Web based network management for Bay Networks switches. I developed Simple Network Management Protocol (SNMP) software for Bay Network switches and software interfaces for Bay Networks' Optivity Network Management System. I wrote software for Java based device management including software interface to the device management and network management for the Accelar routing switch family network management system.

11. I have extensive experience in network communications, including control and management of routing and switching architectures and protocols in

layers 1-7 of the OSI model. Much of my work for Nortel Networks (mentioned above) involved the research and development of network communications technologies. For example, I wrote software for Bay Networks and Nortel Networks switches and routers, developed network technologies for the Accelar 8600 family of switches and routers, the OPTera 3500 SONET switches, the OPTera 5000 DWDM family, and the Alteon L4-7 switching product family. In my lab, I installed, configured, managed and tested many network communications equipment of competitors such as Cisco Systems, Juniper Networks, Extreme Networks, Lucent and Alcatel.

12. I am named as a co-inventor on more than 80 issued patents and I co-authored more than 25 scientific publications, journal articles, and peer-reviewed papers. Furthermore, I am a Senior Member of the Institute of Electrical and Electronics Engineers ("IEEE").

13. I currently serve as a Principal Scientist at my company Telecomm Net Consulting Inc., where I develop network communication technologies and provide research and consulting in advanced technologies, mainly in computer networking and Internet technologies. In addition, I serve as a Co-Founder and

Chief Technology Officer (CTO) of VisuMenu, Inc., where I design and develop architecture of visual IVR technologies for smartphones and wireless mobile devices in the area of network communications. The system is based on cloud networking and cloud computing utilizing Amazon Web Services.

14. Additional details of my background are set forth in my curriculum vitae, attached as **Exhibit A** to this Declaration, which provides a more complete description of my educational background and work experience. I am being compensated for the time I have spent on this matter. My compensation does not depend in any way upon the outcome of this proceeding. I hold no interest in the Petitioner (ServiceNow, Inc.) or the patent owner (BMC Software, Inc.).

**B. Materials Considered**

15. The analysis that I provide in this Declaration is based on my education and experience in the field of computer systems and networks, as well as the documents I have considered including U.S. Patent No. 6,816,898 ("'898 patent") **[Ex. 1001]**, which states on its face that it issued from an application filed on August 16, 2000.

16. I reviewed various documents dated prior to August 2000 describing

the state of the art at the time of the alleged invention of the '898 patent. As explained below, some of these documents are relied upon as actually disclosing the limitations of the '898 patent, while others are being relied upon primarily for background purposes. The prior art documents that I rely upon in this Declaration as actually disclosing the limitations of the claims are:

| Exhibit No. | Description of Document |
|---|---|
| **1003** | Todd Miller et al., "satool – A System Administrator's Cockpit, An Implementation," Proceedings of the 7th USENIX Conference on System Administration (1993), pp. 119-129 |
| **1004** | Brian W. Kernighan et al., The C Programming Language (2d ed. 1988) |
| **1005** | Tim O'Reilly et al., Windows 98 in a Nutshell (1999) |

This Declaration also cites the following additional prior art documents for purposes of describing the relevant technology, including the relevant state of the art at the time of the alleged invention of the '898 patent:

| Exhibit No. | Description of Document |
|---|---|
| **1006** | Evi Nemeth et al., UNIX System Administration Handbook (1989) |
| **1007** | Request for Comments (RFC) 1067, A Simple Network Management Protocol (August 1988) |

| Exhibit No. | Description of Document |
|---|---|
| **1008** | Heinz-Gerd Hegering et al., Integrated Management of Networked Systems (1999) |
| **1009** | Randal L. Schwartz et al., Learning Perl (2d ed. 1997) |
| **1010** | Microsoft Press Computer Dictionary (3d ed. 1997) |
| **1011** | Random House Webster's College Dictionary (2000) |

## III. PERSON OF ORDINARY SKILL IN THE ART

17. I understand that an assessment of claims of the '898 patent should be undertaken from the perspective of a person of ordinary skill in the art as of the earliest claimed priority date, which I understand is August 16, 2000.

18. As I explain in more detail in **Parts IV-V** below, the '898 patent relates to the field of network management and generally describes a method for collecting performance management data, such as server response time, having accompanying meta data. ('898, Abstract, 1:63-2:1.) The data can be collected using a script-based program that can be integrated into a larger performance management system. ('898, 7:28-30, 8:17-19.) This field of technology was well developed by 2000. In fact, in the "Background of the Invention," the patent acknowledges the existence of prior art performance management systems that

collect performance management data. ('898, *e.g.*, 1:38-41 ("Typical network management systems collect information regarding the operation and performance of the network and analyze the collected information to detect problems in the network.").)

19. In my opinion, a person of ordinary skill in the art as of August 2000 would possess at least a bachelor's degree in electrical engineering or computer science (or equivalent degree or experience) with at least four years of computer programming experience. Such a person would also have had an understanding of computer systems and networks, the software for monitoring the operation and performance of those systems and networks (such as Network Management Systems software), and the tools for creating such software, including experience with various scripting and programming languages (such as shell scripting languages (e.g., csh and sh), Perl and C).

20. My opinions regarding the level of ordinary skill in the art are based on, among other things, the disclosure, technology and background of the '898 patent as I explain in **Parts IV-V** below, my experience in the field as described in **Part II** above, and my understanding of the basic qualifications that would be

relevant to an engineer or computer programmer tasked with implementing the methods described in the ’898 patent.

21. Although my qualifications and experience exceed those of the hypothetical person having ordinary skill in the art defined above, my analysis and opinions regarding the ’898 patent have been based on the perspective of a person of ordinary skill in the art as of August 2000.

## IV. RELEVANT BACKGROUND

22. The ’898 patent, entitled “Interfacing External Metrics into a Performance Management System,” relates to the field of network management and generally describes a method for collecting performance management data, such as server response time, having accompanying metadata. (’898, Abstract, 1:63-2:1.) The data can be collected using a script-based program that can be integrated into a larger performance management system. (’898, 7:28-30, 8:17-19.)

23. In this section, I provide a brief background of the state of network management prior to August 2000 pertinent to the ’898 patent.

### A. Network Management

24. By the 1980s, businesses and enterprises frequently used computers that were connected to a network (commonly known as a “Local Area Network”

(LAN)), which allowed the computers to communicate with each other using a network communications technology such as Ethernet. As the number of devices connected to a network increased, so did the need for software tools to monitor and manage those devices. An entire field referred to as "network management" grew out of this need. For example, in August 1988, a standardized protocol known as the Simple Network Management Protocol ("SNMP") was introduced and remains in wide use today. (*See* RFC 1067, A Simple Network Management Protocol (August 1988), Ex. 1007, § 1.) SNMP, as its name suggests, provides a simple protocol that allows one "network element" (such as a computer) to gather information about and monitor another network element.

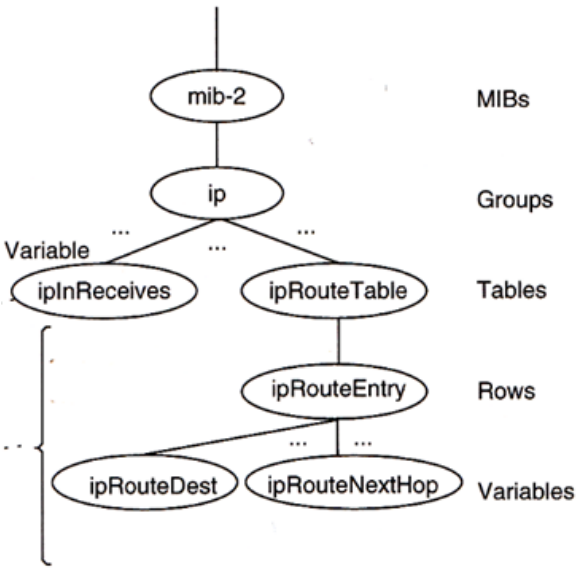
25. The protocol provides a means by which "management information for a network element may be inspected or altered by logically remote users." (*Id.*, § 1.) Each network element contains a software program known as an "agent" that is responsible for responding to SNMP requests from the network management "station" ("NMS"). The SNMP agent can, for example, gather information about the network element and report that information back to the management station. (*Id.*, § 3.)

26. As suggested by the name, the SNMP protocol is very simple. In general, the NMS requests information from a device, and the SNMP software agent for the device gathers the specific information and sends it back to the NMS. SNMP defines simple "get" and "set" functionality to get (or set) a specific value on a device. For example, an NMS can ask a network device for the number of ports using "get" request, and the and the device will reply with the value (e.g., request: "how many ports do you have?"; response: "I have 12 ports").

27. A key component of network management using SNMP is a "**Management Information Base**," also referred to by its acronym, "MIB." Generally speaking, a MIB is a collection of variables that describe characteristics about a particular network element being monitored (such as a computer). (*See* Heinz-Gerd Hegering et al., Integrated Management of Network Systems (1999) ("Hegering"), Ex. 1008, at 158.) MIB "variables" are also sometimes referred to as MIB "objects." (*Id.*) A MIB is typically associated with an SNMP agent.

28. The structure of an SNMP agent's MIB is defined by what is called an "Internet MIB." (*Id.*) The term "***Internet*** MIB" (also known as a "TCP/IP MIB") does not refer to the idea that the MIB is accessible anywhere through the Internet.

The term "Internet MIB" instead refers to the fact that MIBs defined by that structure contain variables related to the TCP/IP standard. (TCP/IP, which stands for Transmission Control Protocol/Internet Protocol, is a basic communications protocol used for communication over the Internet.) An Internet MIB specifies which variables appear in an SNMP agent's MIB, how the variables are structured, which meaning they incorporate, and how they can be identified. (*Id.*) The variables in an Internet MIB are related hierarchically. (*Id.*) An example of this hierarchical relationship is depicted below:




(Hegering, at 160 (Fig. 6.2 (excerpt)).) As the figure above shows, the variable

"ipInReceives" and the table "ipRouteTable" are children of an "ip" group. The "ipRouteTable" includes an "ipRouteEntry" child, which in turn includes "ipRouteDest" and "ipRouteNextHop" child variables.

29. As I mentioned above, an Internet MIB, such as the one excerpted above, defines which variables will appear in an SNMP agent's MIB and their structure. The figure below provides a helpful illustration of the conceptual relationship between an Internet MIB and an SNMP agent's MIB:

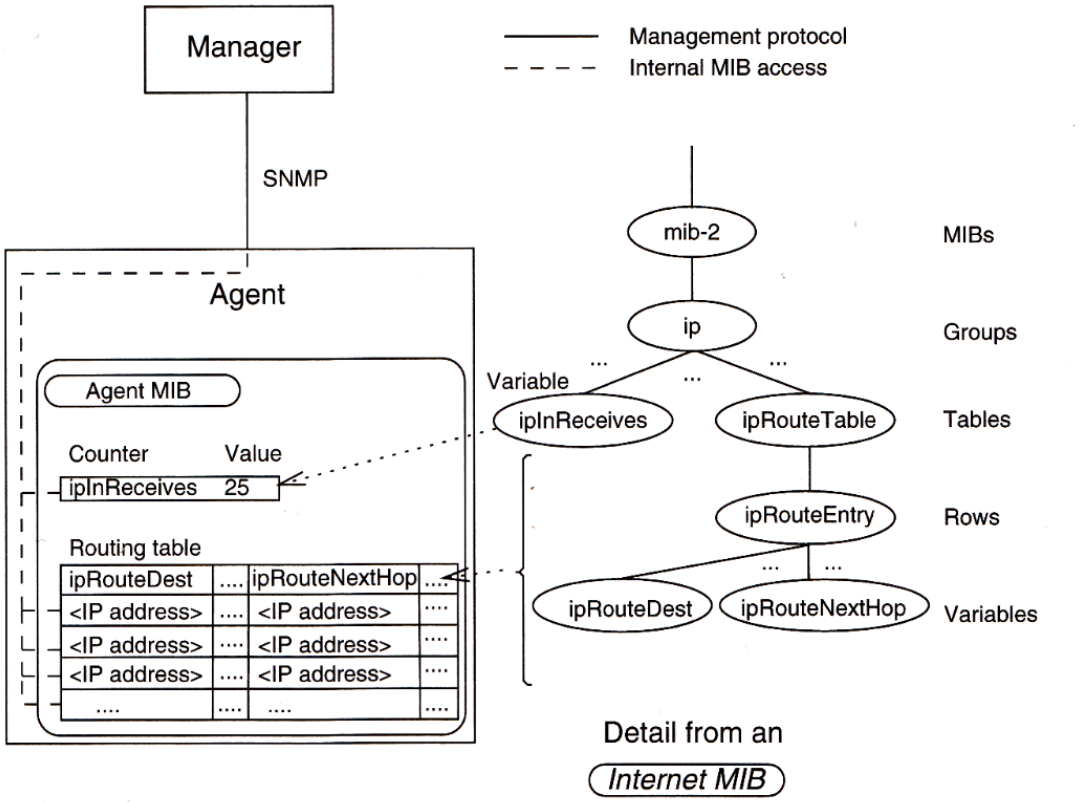



**Figure 6.2**
Example of an agent and Internet MIB

(Hegering, at 160 (Figure 6.2).) The figure above shows, for example, that the "ipInReceives" variable from the Internet MIB has been included in the SNMP agent's MIB and has a value of 25, which can be queried by the manager (or network management station) using SNMP.

### B. Scripts and Scripting Languages

30. "Scripts" were well-known to persons of ordinary skill in the art by

August 2000. As the ’898 patent acknowledges, script-based programs were already being used in a computer and network management context before August 2000. (’898, 2:9-17.) In fact, scripts pre-date the ’898 patent by decades. For example, scripts were used on IBM mainframe computers in the 1950s. The first PCs were introduced with BASIC as a scripting language in the early 1980s.

31. Generally speaking, a script consists of instructions written in a plain text, interpretable language. This means that, in contrast to an executable program (or binary), which is executed directly by a microprocessor (or CPU), the instructions specified in a script are carried out by another program called an interpreter. The interpreter program itself may be an executable program that is executed by the CPU.

32. There are many computer programming languages for writing scripts. The ’898 patent, for example, refers to shell script languages and Perl. (’898, 7:18-20.) One shell script language, called “sh” or the “Bourne shell,” dates from 1977 when it was included in Bell Laboratories’ legendary UNIX operating system. Another, called “csh” or “C Shell,” was initially released in 1978 and distributed with the Berkeley Sockets Distribution (“BSD”) of UNIX. Perl, which is short for

"Practical Extraction and Reporting Language," was originally developed in 1987.

33. To provide a basic illustration how scripts can work in practice, I will now describe two examples of Perl scripts that were created for this Declaration. The first is extremely simple:

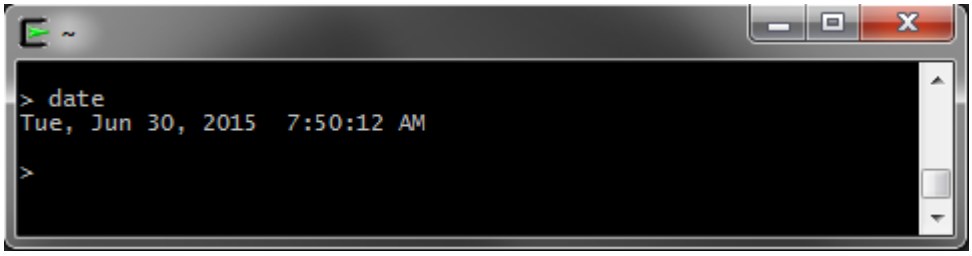
```
#!/usr/bin/perl
print "hello world";
```

The first line, also called the "shebang" line, identifies the name and location of the Perl interpreter software that reads and carries out the instructions specified in the script. In this case, the interpreter is located in the directory /usr/bin and named perl. The second line is simply an instruction to print out the string "hello world." (*See generally* Randal L. Schwartz et al., Learning Perl (2d ed. 1997) ("Schwartz"), Ex. 1009, at 5-6.)

34. This script can then be saved in a text file on the computer, such as a file called "hello_world.pl." (*See generally* Schwartz, Ex. 1009, at 3-4.) The script could then be executed by simply typing the file name into the prompt and pressing the "Enter" key, as shown below:


This will cause the Perl interpreter to be invoked, which in turn will interpret the instructions in the script. In this example, it will interpret and carry out the instruction (print "hello world";) by printing the string to the display and then returning to the command line prompt:

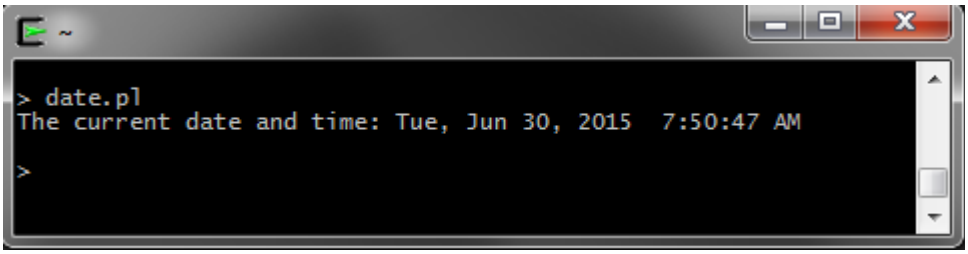

35. A Perl script can also be used to execute another program. (*See generally* Schwartz, Ex. 1009, at 145.) For example, the UNIX operating system typically includes a program called “date” that when executed outputs the current date and time, as shown below:




36. A simple Perl script to execute the "date" program and print the result is as follows:

```
#!/usr/bin/perl
my $date = `date`;
print "The current date and time: $date";
```

(*See generally* Schwartz, Ex. 1009, at 145.) As with the previous example, the first line identifies the name and location of the Perl interpreter program. The second line is an instruction to execute the date program and store the output of that program into a variable named $date. The third line is an instruction to print introductory text "The current date and time: " followed by the value of the $date variable. Saving the above lines of Perl code to a file named "date.pl" and executing it using the command line interface produces the following output:



## V. THE '898 PATENT'S TECHNIQUE FOR COLLECTING PERFORMANCE MANAGEMENT DATA

### A. The Specification

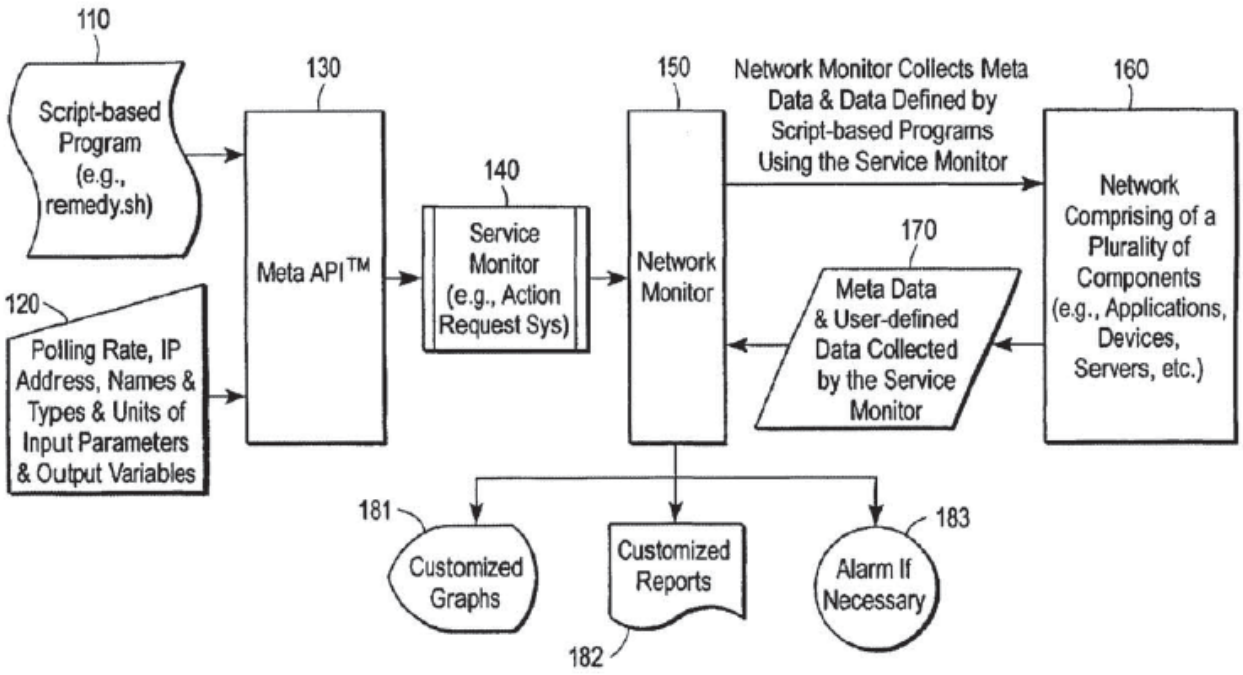
37. As I mentioned in **Part VI** above, the '898 patent relates to the field of network management and generally describes a method for collecting performance management data, such as server response time, having accompanying meta data. ('898, Abstract, 1:63-2:1.) The data can be collected using a script-based program that can be integrated into a larger performance management system. ('898, 7:28-30, 8:17-19.)

38. The '898 patent acknowledges in the Background of the Invention section that performance management systems that collect performance management data already exist. ('898, *e.g.*, 1:38-41 ("Typical network management systems collect information regarding the operation and performance of the network and analyze the collected information to detect problems in the network.").)

39. The '898 patent asserts, however, that developing software for prior art systems is time consuming and expensive, and thus it would be advantageous to reduce software development in network management systems. ('898, 1:52-62.)

The ’898 patent also asserts that prior art systems typically collected only network infrastructure information, such as server response time, and separate programs were required to gather business-oriented information, such as the number of tickets sold in an airline’s web site. (’898, 1:63-2:12.)

40. Figure 3 of the ’898 patent, below, is a diagram that illustrates key concepts of the ’898 patent:




(’898, Fig. 3; *see also id.*, 2:42-43 (“FIG. 3 illustrates a data flow diagram of monitoring system for monitoring a network of components.”).)

41. The ’898 patent explains that script-based program **110** (at upper left,

above) can be a file containing instructions written in a scripting language, such as Perl or any shell script language. (’898, 7:14-20; *see also id.*, Fig. 4, 9:52-56 (“FIG. 4 shows an example of a script-based program, ‘remedy.sh’, provided by the user. This is a typical simple shell script for collecting management or business oriented network data not covered by the performance management system.”).) Generally speaking, the script-based program contains instructions that, when invoked, will collect data defined by the user from components on the network. (’898, *e.g.*, 7:28-30, 9:52-56.)

42. The ’898 patent explains that the script-based program **110** can be incorporated into the network management system (e.g., network monitor **150**). (’898, *e.g.*, 7:15-18, 8:17-19.) To do so, the script-based program is registered with the system as a service monitor **140**. (’898, 8:44-46, 8:52-54.) The network management system runs the service monitor automatically to collect customized data from the network components. (’898, 11:13-16; *see also id.*, 10:46-48.)

43. The ’898 patent further explains that “meta data” accompanies the data collected from the network components. (’898, 4:11-22.) The meta data “indicates what it is that is being collected and what is to be done with the data

being collected." ('898, 4:15-17.) "For example, the meta data may indicate that an alarm is to be associated with the data or may indicate how to administer." ('898, 4:17-19.)

44. The network management system processes the collected data by, for example, "generat[ing] customized graphs **181**, customized records **182**, and/or setting an alarm **183** if necessary." ('898, 7:30-34.)

### B. The Claims of the '898 Patent

45. This Declaration addresses both independent claims of the '898 patent. Independent claim 1 recites:

> 1. A method comprising:
>
> [a] collecting performance management data having accompanying meta data, the meta data including information defining the performance management data and information indicating operations to be performed on the performance management data; and
>
> [b] generating output data for display using the collected performance management data according to the information indicating the operations to be performed on the performance management data. and a second indicator for the at least one IT subcomponent, wherein the first and second indicator are each separately visible at the same time on a single display window of a display unit.

('898, 12:55-64 (Claim 1).) I added the bracketed notations (e.g., "[a]," "[b]," etc.)

to facilitate identification of these limitations in my Declaration. The second independent claim, claim 6, recites:

> 6. A method for providing an interface between a user and a performance management system, the performance management system being connected with a network, the network including a plurality of components coupled by a plurality of connections, the performance management system collecting data of the components, the method comprising:
>
> [a] receiving at least one script-based program from the user, the script-based programs defining data types not provided by the performance management system;
>
> [b] integrating the program to the performance management system as a service monitor, the performance management system using the service monitor to periodically collect data of the defined data types from the components.

The additional claims addressed in this Declaration—i.e., claims 2-5 and 7-12—depend directly or indirectly from independent claim 1 or 6. I address those claims in detail in **Part VI** below.

### C. Claim Construction

46. I have been informed by counsel that invalidity involves a two-step analysis. In the first step, the scope and meaning of a claim is determined by construing the terms of that claim. In the second step, the claim as interpreted is compared to the prior art. Thus, before I address the application of the prior art to

the claims of the '898 patent in **Part VI** below, I will provide proposed constructions for certain terms in the claims addressed in this Declaration, from the perspective of a person of ordinary skill in the art.

47. I have been informed by counsel that a claim in an unexpired patent subject to *inter partes* review must be given its broadest reasonable construction that is consistent with the specification of the patent in which it appears, which is different from the manner in which the scope of a claim is determined in litigation. I have applied this standard in my analysis below.

**1. "meta data"**

48. The term "meta data" is recited in independent claim 1. In my opinion, the proper construction of "**meta data**" is "**data about other data**." This construction is consistent with the ordinary meaning of the term in the field at the time. For example, the Microsoft Press Computer Dictionary defines "meta data" is "[d]ata about data. For example, the title subject, author, and size of a file

constitute meta data about the file." (Microsoft Press Computer Dictionary (3d ed. 1997), Ex. 1010, at 305 (emphasis added).)[1]

49. The written description of the '898 patent provides examples of "meta data" consistent with this ordinary meaning. For example, the specification states that "the meta data may indicate that an alarm is to be associated with the data or may indicate how to administer." ('898, 4:15-19.) As another example, the patent also discusses "meta data that indicates what operations or actions to perform on or with the data." (*Id.*, 4:26-27.) In my opinion, therefore, one of ordinary skill in the art would have understood "**meta data**" to mean "**data about other data**."

**2. "accompanying"**

50. The term "accompanying" is recited in independent claim 1 as part of the larger phrase, "collecting performance management data having accompanying meta data." As I explain below, consistent with its plain meaning, the broadest

[1] Persons of ordinary skill in the art use the term "meta data" and "metadata," the latter lacking any intervening space character, interchangeably.

reasonable interpretation of the term "**accompanying**" is "**associated**."

51. Although in one sense the term "accompanying" in the real world may imply some kind of physical togetherness (such as two people "accompanying" each other on a business trip), the term has a broader meaning in the computer context to a person of ordinary skill in the art. For example, the Random House Webster's College Dictionary (2000) **[Ex. 1011]**, provides a number of definitions for the verb "accompany":

> **1.** to go along or in company with. **2.** to exist or occur in association with: *Thunder accompanies lightning*. **3.** to cause to be associated with or attended by: *He accompanied his speech with gestures*. . . . [additional definitions related to musical accompaniment]

(Random House Webster's College Dictionary (2000), Ex. 1011, at 9.) As these definitions confirm, the term "accompanying" includes simply being "associated" with something else. This definition is the most accurate in the context of the computer-implemented system of the patent. The '898 patent explains, for example, that performance management data is collected from the network and analyzed using associated meta data already present within the performance

management system:

> Once the [service] monitor is activated, the network performs [sic] monitoring and management system runs it periodically to collect the type of data defined in the service monitor. Network monitor also analyzes the data with meta data, which has already been defined within performance monitoring and management system.

('898, Ex. 1001, 10:67-11:6, 4:11-13 ("In the present invention, generic modules, as well as the data storage, receive data and are able to process the data using information that accompanies the data.").)

The passage quoted above states that the "meta data" has "already been defined within [the] performance monitoring and management system," which confirms to a person of ordinary skill in the art that the "performance management data" and the "meta data" are associated with each other, and also indicates that the metadata not "accompany" the performance management data in a physical sense (such as travelling with it). Accordingly, the broadest reasonable construction of "**accompanying**" as used in the patent is "**associated**."

**3. "service monitor"**

The term "service monitor" is recited in dependent claim 3 and independent

claim 6. Dependent claim 3, for example, recites the step of "integrating the at least one script-based program into the performance management system as a service monitor; and using the service monitor to periodically collect the performance data." The term "service monitor" does not have a commonly understood meaning to persons of ordinary skill in the art. The specification describes a process of adding a new service monitor, stating that "[t]he new service monitor can then be activated to monitor any applicable devices, applications or servers in the network." ('898, Ex. 1001, 8:52-54.) In my opinion, based on this description, the broadest reasonable construction of "**service monitor**" is a "**program for monitoring a device, application or server in a network**."

52. The specification describes a number of other characteristics of a "service monitor," including the fact that it automatically collects data from components on the network ('898, 11:13-16), and that the collected data includes user-defined data ('898, 7:22-30). These additional details are appropriately considered part of the definition of a "service monitor" under the narrower standards applicable in litigation, but not under the "broadest reasonable construction" standard that I have been instructed to apply here. In my opinion,

those details are not part of the "broadest reasonable construction."

**4. "performance management data"**

53. The term "performance management data" is recited in each independent claim. In my opinion, the broadest reasonable construction of "**performance management data**" is "**information regarding the operation or performance of a network environment**." This definition is consistent with the specification:

> FIG. 2 illustrates an embodiment of a network monitor **22** capable of detecting problems or potential problems in a network environment. Network monitor **22** includes a data collection module **30** that collects information from various devices or applications, such as information regarding network utilization (or device utilization), lost packets, response time, or number of errors. Data collection module **30** collects information regarding the operation or performance of the network environment on one or more communication links **31**. Data collection module **30** can collect data from any number of networks and any number of network devices or applications.

('898, Ex. 1001, 6:1-12 (underlining added).)

54. In my opinion, therefore, the broadest reasonable construction of

"**performance management data**" is "**information regarding the operation or performance of the network environment**."

**5. "business-oriented performance management data"**

55. The term "business-oriented performance management data" is recited in dependent claim 7, which reads: "The method defined in claim **6** wherein the data collected by the service monitor is business-oriented performance management data of the components." The '898 patent does not define "business-oriented performance management data." The written description does not use this phrase at all. It refers once to "business oriented network data," and uses the phrase "business-oriented data" four times. ('898, 2:1-2, 2:2-4, 9:53-56, 11:41-46, 11:46-49.) With respect to "business-oriented data," the patent states:

> In fact, management or business-oriented data is usually neglected by current network management system. For example, a typical network management system does not keep track of the number of tickets sold in an airline's web site, even though the statistics on the number of tickets sold would be very useful to an airline using the network management system.

('898, 2:2-8.)

56. It is not clear whether the above passage actually relates to "business-

oriented **performance management** data," but in any case, the patent is clear that the passage above is intended as an example only. Thus, in my opinion, the phrase "**business-oriented performance management data**" should be interpreted under its broadest reasonable construction as "**performance management data relating to a business**." The term "performance management data" should be defined in accordance with the definition above (i.e. "information regarding the operation or performance of the network environment").

57. The following table identifies the above claim constructions:

| Claim Term | Broadest Reasonable Construction |
|---|---|
| "**meta data**" | "data about other data" |
| "**accompanying**" | "associated" |
| "**service monitor**" | "program for monitoring a device, application or server in a network" |
| **"performance management data"** | "information regarding the operation or performance of the network environment" |
| **"business-oriented performance management data"** | "performance management data relating to a business" |

## VI. APPLICATION OF PRIOR ART TO CLAIMS 1-12 OF THE '898 PATENT

58. I have reviewed and analyzed the prior art references and materials listed in **Part II.B** above. In my opinion, each limitation of claims 1-7 and 9-12 is disclosed and/or suggested by Todd Miller et al., "satool – A System Administrator's Cockpit (1993) **[Ex. 1003]** ("**Miller**"). Claim 8 is disclosed and/or suggested by Miller in view of Brian W. Kernighan et al., The C Programming Language (2d ed. 1988) **[Ex. 1004]** ("**Kernighan**"), and Tim O'Reilly et al., Windows 98 in a Nutshell (1999) **[Ex. 1005]** (**O'Reilly**").

59. Counsel has informed me that Miller, Kernighan and O'Reilly properly qualify as prior art to the '898 patent. Miller was published in 1993, Kernighan was published in 1988, and O'Reilly was published in 1999, before the application filing date on the face of the '898 patent. Before applying the prior art to claims 1-12, I will provide a brief summary of the references.

### A. Summary of Prior Art References Applied in this Declaration

#### 1. Miller (Ex. 1003)

60. Miller is a paper published by the USENIX Association. Founded in 1975, USENIX is a well-known association that brings together various

communities of computer professionals, such as computer scientists and system administrators ("sysadmins"), to discuss and share ideas on various topics and issues in the computing world. (Miller, inside back cover.) USENIX includes a special interest group and associated annual conference for sysadmins called "LISA," which is short for "Large Installation System Administration." (Miller, at inside front cover, ii, inside back cover.) (The LISA special interest group was formerly called "System Administrators' Guild" or "SAGE." (Miller, inside back cover).) Miller states that it was originally presented at the LISA conference in Monterey, California in November 1993. (Miller, at ii.)

61. Miller describes a software-based system called "**satool**" that allows a system administrator to monitor the operation and performance of a large number of computers on a network. (Miller, at 119.) The system has three parts: (1) a data gathering agent on each computer being monitored; (2) a data collecting server that collects data from each computer into a database; and (3) a graphical user interface ("GUI") that "acts as the interface between the user and the database." (Miller, at 119 (Abstract), 120 (left column, under "Components").)

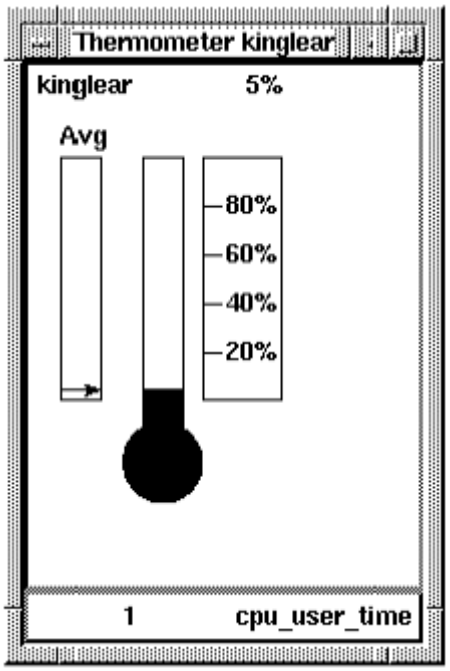
62. The data gathering agent in Miller takes the form of Simple Network

Management Protocol ("SNMP") agent software that uses "helper scripts" to collect performance data. These "helper scripts" collect data that is not normally provided by a standard SNMP agent and MIB and put it into a format compatible with the satool system. (Miller, at 120 ("The agent is an SNMP agent extended to use the **satool** MIB. It uses 'helper scripts' to massage data from standard UNIX commands.") (bold in original) (left column, under "Components"), 121 ("The values for most of the variables are obtained via 'helper scripts' that run standard UNIX commands and parse the output into a form that the agent can use.") (right column, under "satool Variables"), 121 ("We have extended the CMU [Carnegie-Mellon University] SNMP 1.1b agent and MIB to include variables of interest to UNIX system administrators not included in the standard network or host MIBs.") (under "Config File").)

63. The data collecting server (also referred to as "satool daemon" or "satoold") polls the agents periodically at configurable intervals for performance data and stores the data collected from the agents in a database. (Miller, at 120 ("The server polls agent at set intervals and stores the resulting data in an *ndbm*(3) database. . . . **satoold** polls clients for data via SNMP at configurable intervals.")

(emphasis in original) (bottom of left column to top of right column).) The server also services requests from the GUI to access that data. (Miller, at 120 ("[The server] also services requests from the display system using an SMTP-like protocol.") (bottom of left column to top of right column).)

64. The GUI display in Miller enables the user to view the contents of the satool database. (Miller, at 122 (left column, under "Display System").) The GUI contains are "visual widgets representing the health of [a] machine." (Miller, at 119 (right column).) An example of a "thermometer" display object is shown at right. (*See* Miller, at 123 (left column).) Miller explains that the "visual widgets," also called "display objects," include a sliding histogram, a thermometer and a text object. (Miller, at 122 ("**satool's** display objects, currently a sliding histogram, a thermometer, and a text object, are used to view the data in the database.") (emphasis in original) (right column, under "Display Objects").)




65. The display system also includes alarm functionality so that "[i]f the

value of a variable crosses the alarm threshold, a special action (blinking, beeping, reverse video, digital pager, etc.) is taken on the screen to indicate it." (Miller, at 119 (right column); *see also id.*, at 125 (left column, top).)

66. The types of display objects and alarms associated with collected data can be configured by a user with configuration files. (Miller, at 123 ("The sysadmin can configure their **satool** display to select the hosts whose data will be displayed, types of widgets to use, alarm thresholds and actions, and screen layout.") (right column, under "Configuration") (emphasis in original).)

67. Miller also provides an in-depth tutorial for allowing the system to monitor a new type of data, a process that Miller asserts is "fairly easy." (Miller, at 125 (left column, under "Extending satool"); *see generally* Miller, at 125-26.) In the example provided in Miller, the new data type added provides an indication as to whether a network gateway (which allows computers on a local network to access computers outside the network) is operating correctly. (Miller, at 125 (left column, under "Extending satool").)

68. Miller explains that the first step in adding this data type to the system is to write a helper script that the SNMP agent will invoke for execution. (Miller, at

125.) The exemplary helper script, called "traceroute-helper," is written in the Perl scripting language. (Miller, at 125 (left column, under "Agent").) The helper script uses a feature known as "traceroute" to determine whether the network gateway is operating correctly. The term "traceroute" refers to a network diagnostic tool commonly used with Unix-based systems that tracks the route or path of packets across a network on their way to a given destination device.

69. The exemplary "traceroute-helper" script in Miller outputs an integer value to indicate if the network gateway is operating properly. In particular, it outputs a "1" if the gateway is working, and "0" if there is a problem. (*Id.*) This indication provides one example of "**performance management data**."

70. The next step is to add information to the MIB for the SNMP agent that defines the new object type to record whether the gateway is functioning. Miller provides an exemplary MIB for this data type:

```
satoolGatewayOk  OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-only
        STATUS  optional
        ::= {satool 11 }
```

(Miller, at 125.) The first line above defines a MIB object called

"satoolGatewayOk." (Miller, at 121 (right column, under "Sample MIB entry").) The second line specifies that the object is an INTEGER. (*Id.*) The ACCESS line indicates that satoolGatewayOk is read-only. (*Id.*) The STATUS line indicates that the object is "optional" because it is not part of the standard MIB. The last line (beginning with two colons) describes where the object type "fits in" to the MIB hierarchy. (*Id.* at 122.) In this case, the "satoolGatewayOk" object type is the eleventh leaf in the satool branch. (*Id.*) As I will explain below, the MIB described above provides an example of "**meta data including information defining the performance management data**," as claimed.

71. Miller explains that, next, the SNMP agent is updated so that it knows to execute the "traceroute-helper" script when it receives a query for the value of the "satoolGatewayOk" MIB object. (*Id.*, at 125.) Miller explains that "[a]ll we are really doing here is running our helper script and passing what it prints back as an int [integer]" to the SNMP agent. (*Id.* (right column).)

72. The next step is to update the satool server so that it knows to monitor the value of the "satoolGatewayOk" object. (*Id.*, at 125 (right column, under "Server").) To do this, first, a variable named "gateway_ok" is defined. (*Id.*) Next,

functionality is added so that this new variable is updated when the server receives a response to its SNMP query for satoolGatewayOk. (*Id.*, at 125-26 ("This converts the snmp variable into an int and stores it into the correct field of the struct.").) Finally, functionality is added to send the text "gateway_ok" and its value to the display system when requested. (*Id.*, at 126 (left column, top).)

73. The last step is to update the display system so that it knows about the "gateway_ok" variable. This involves specifying the name of the variable and associating a threshold value and a default display object with the variable. Miller explains:

> The final step is to add the new variable to the GUI. The new variable must be added to the default list in procedure BuildList, file IO.tcl:
>
> gateway_ok 0 X

(*Id.*, at 126 (left column, under "GUI").) As shown above, in the case of the variable "gateway_ok," the threshold value is specified as being "0," so that an alarm is triggered when the gateway is down. (*Id.*) The default display object is a text display object ("X"). (*Id.*) As I will explain in more detail below, this information provides an example of "**meta data . . . indicating operations to be**

**performed on the performance management data**," as recited in claim 1.

74. After these steps are carried out, the satool server may poll an SNMP agent for its value for the satoolGatewayOk MIB object, which will cause the SNMP agent's traceroute-helper script to be executed. The script's results will then be returned to the satool server as an SNMP response, which the satool server will then be able to provide to the display system. The display system may then display the results, possibly including an alarm.

### 2. Kernighan and O'Reilly (Exs. 1004, 1005)

75. My Declaration relies upon Kernighan and O'Reilly for certain aspects of dependent claim 8. Kernighan is a famous textbook describing the syntax and features of the C programming language. My Declaration relies on the portion of Kernighan that explains how a computer program can include input parameters such as command line arguments (as recited in claim 8).

76. O'Reilly is a textbook that describes certain features of the Windows 98 operating system. My Declaration relies on a portion of O'Reilly describing a Windows utility called "tracert," which is a Windows version of the "traceroute" program described in Miller. O'Reilly describes the input and output parameters

of the traceroute program. I will discuss both references in my analysis of claim 8 in **Part VI.B.8** below.

### B. Application of the Prior Art to Claims 1-12

#### 1. Claim 1 is Disclosed or Suggested by Miller

##### (a) "collecting performance management data having accompanying meta data, the meta data including information defining the performance management data and information indicating operations to be performed on the performance management data" (Claim 1[a])

77. Because of the length of this limitation, I have divided it into portions that I will separately address below.

###### (i) "collecting performance management data"

78. Miller discloses the step of "**collecting performance management data**." For purposes of my analysis, the "performance management data" includes information that indicates whether a network gateway is working correctly (thus indicating whether computers on the network can contact computers outside the

network). (*See* Miller, e.g., at 125 (left column).)[2] For example, as I explained above, Miller discloses a "traceroute-helper" script that determines if the network gateway is operating properly. In particular, it outputs a "1" if the gateway is working, and "0" is there is a problem. (*Id.* at 125 (left column, under "Agent").)

79. As I explained above, the broadest reasonable construction of "**performance management data**" is "**information regarding the operation or performance of the network environment**." The information output by the "traceroute-helper" program (e.g. the "1" or "0") qualifies as performance management data because it indicates whether or not the network gateway is currently operational. (*Id.* at 125 (left column, under "Extending satool").) Because the network gateway is responsible for allowing computers to

[2] Miller provides other examples of performance information that is obtained by satool, such as memory statistics, CPU activity, free disk space and other information that represents the health (or performance) of a computer or the network. (Miller, e.g., at 119 (right column); see also Miller, at 127-29 (listing dozens of values monitored in the satool MIB), 121 ("The full satool MIB can be found in appendix A of this paper.") (emphasis in original).)

communicate with other networked computers in the outside world (*id.* at 125 (left column, under "Agent")), the information returned from the "traceroute-helper" script relates to "the operation or performance of the network environment."

80. Miller also discloses several examples of "**collecting**" this performance management data. For example, as I explained above, Miller discloses a "**traceroute-helper**" script that determines if the network gateway is operating properly and outputs a "1" if the gateway is working, and "0" if the gateway is down. (*Id.*) The determination of this value by the traceroute-helper program provides one example of "**collecting"** performance management data.

81. Miller discloses at least two further instances in which the performance management data is "collected." First, the SNMP agent returns the result from the "traceroute-helper" program to the satool server. (*Id.* at 125-126 (right column, under "Server").) Second, the performance management data is also "collected" when the display system obtains and uses it to present a display to the system administrator. (*Id.*, e.g., at 120 ("The display system (written in **tcl** and **tk**) interacts with the user and initiates data transfers from the server.") (emphasis in original) (left column, under "Components").) Miller therefore discloses at least

three examples of "collecting" performance management data, any one of which can satisfy this step.

**(ii) the performance management data "having accompanying meta data, the meta data including information defining the performance management data"**

82. As I explained above in **Part V.C.1-2**, the term "meta data" under its broadest reasonable construction means "data about other data," while the term "accompanying" means "associated."

83. Miller discloses that the performance management data has associated "**meta data**" that includes "**information defining the performance management data**." The "meta data" for purposes of Miller takes the form of information that defines the output from the "traceroute-helper" program (e.g. the output value indicating if the network gateway is operational).

84. More specifically, as I explained above, the "performance management data" is obtained using a "traceroute-helper" script. Miller explains that "[a]ll we are really doing here is running our helper script and passing what it prints back as an int"—i.e., an integer. (Miller, at 125 (right column) (emphasis added).) The following source code (in the C programming language) invokes the

traceroute-helper script and defines its output:

```
[1:]  fildes = popen("traceroute-helper","r");
[2:]  if (!fildes) return(NULL);
[3:]  if (fscanf(fildes, "%d", &n) < 0)
[4:]  {
[5:]          pclose(fildes);
[6:]          return(NULL);
[7:]  }
[8:]  long_return = (long)n;
[9:]  pclose(fildes);
[10:] return (u_char *) &long_return;
```

(Miller, Ex. 1003, at 125 (right column; line numbers added; emphasis added).)

85. The first line of the source code above invokes (i.e. opens) the traceroute-helper script for read ("r") access, and the second line terminates the program if it did not open properly. The third line, which executes if "traceroute-helper" opened properly, uses "fscanf," a standard C library function for reading the output from the traceroute-helper script (identified by the "fildes" variable).

The "**%d**" in the third line of source code is a standard feature of the C programming library that tells "fscanf" to read an integer (decimal) from the script output.[3] The "&n" in the above source code is an instruction to save that integer into a particular variable named "n." Thus, the above source code instructions collect the performance management data, and then define it as integer for storage. The "**%d**" string therefore provides one example of meta data associated with the performance management data that defines the data.

86. A second way that Miller discloses "accompanying meta data" that includes "information defining the performance management data" takes the form of information that defines a MIB object that corresponds to the output of a helper script. As I explained **Part VI.A.1** above, in the traceroute-helper script example described in Miller, an associated MIB object named satoolGatewayOk is defined:

[3] *See* Brian W. Kernighan et al., "The C Programming Language" (2d ed. 1988), Ex. 1004, at 157-59. In the example described in the text, Kernighan explains that the "%d" string specifies that the input data is a "decimal integer." (*Id.* at 158, Table 7-2.)

```
satoolGatewayOk  OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-only
        STATUS  optional
        ::= {satool 11 }
```

(Miller, at 125 (left column, under "Agent").) The first line above defines a MIB object called "satoolGatewayOk." (Miller, at 121 (right column, under "Sample MIB entry").) The second line specifies that the object is an INTEGER. (*Id.*) The ACCESS line indicates that satoolGatewayOk is read-only. (*Id.*) The STATUS line indicates that the object is "optional" because it is not part of the standard MIB. The last line describes where the object "fits in" to the MIB hierarchy. (*Id.* at 122.) In this case, satoolGatewayOk is the eleventh leaf in the satool branch. (*Id.*) A MIB object associated with the output of a helper script therefore also clearly qualifies as "accompanying meta data" that includes "information defining the performance management data."

87. A third way that Miller discloses "accompanying meta data" that includes "information defining the performance management data" takes the form of information that identifies the type of data sent from the satool server to the display system. Miller explains that "when the display system requests a host's

data," the server responds with a variable name and its value. (Miller, at 126 (left column).) In particular, Miller explains that the following line of code adds a "variable value" pair to the information sent from the satool server to the display system:

> sprintf((char *)ret + strlen(ret), "%s %d", "gateway_ok", sat_var->gateway_ok);

(Miller, at 126 (left column).) In the code above, "sprintf" is a well-known standard C library function to create a formatted string. The "%s %d" specifies the formatting for the string. In this case, the format is a character string ("%s") followed by space, then an integer ("%d"). Thus, if the value of the gateway_ok variable is 1, for example, then the result of executing the above sprintf function will be that the string "gateway_ok 1" is added to the information sent from the server to the display system. The string "gateway_ok" in that information thus defines the value that follows it and identifies that value as corresponding to the "gateway_ok" variable. (Miller, at 126 (left column).) Thus, Miller clearly discloses "accompanying meta data" that includes "<u>information defining the performance management data</u>."

**(iii) "accompanying meta data" that includes "information indicating the operations to be performed on the performance management data"**

88. In addition to "information defining the performance management data" the "accompanying meta data" in claim 1 also includes "information indicating the operations to be performed on the performance management data." This corresponds in Miller to a variable that instructs the display system how to display the output of the "traceroute-helper" script.

89. More specifically, Miller explains that a user can specify how a piece of performance management data should be displayed, including whether it should trigger an alarm. For example, Miller discloses how to associate a variable called "gateway_ok" that is associated with the output of the "traceroute-helper" program:

> The final step is to add the new variable to the GUI. It must be added to the default list in the procedure BuildList, file IO.tcl:
>
> gateway_ok 0 X
>
> The variable name is followed by a threshold value and its default display object. A threshold value of 0 for a Boolean variable triggers an alarm when the gateway is down; a text

display object is specified.

(Miller, at 126 (emphasis added) (left column, under "GUI").) The string, "gateway_ok 0 X" qualifies as metadata that includes "information indicating the operations to be performed on the performance management data."

90. In particular, "**gateway_ok**" is the name of the variable that holds the output of the traceroute-helper script. The next portion of the string, "**0**," tells the system to trigger an alarm if the value of gateway_ok is zero (indicating a failure of the network gateway). As I noted previously in **Part VI.A.1**, Miller explains that an alarm causes the data object for a variable to change appearance in some way. (Miller, at 125 ("**satool's** alert system notifies the user when data values cross stated thresholds. The data object causing the alert will change appearance (color, reverse video, new icon, etc.)." ) (underlining added, bold in original) (left column).) The last portion of the string, "**X**," specifies that gateway_ok will be treated as a text object for purposes of display. (*Id.* at 126 (left column, under "GUI").) Thus, Miller clearly discloses "accompanying meta data" that includes "information indicating the operations to be performed on the performance management data."

**(b) "generating output data for display using the collected performance management data according to the information indicating the operations to be performed on the performance management data" (Claim 1[b])**

91. Miller discloses this limitation as well. As I explained above, the satool system in Miller includes a display system that generates a display of performance management data. (Miller, e.g., at 122 ("The display system provides an X-based Graphical User Interface (GUI) for viewing the contents of the **satool** database.") (bold in original) (left column, under "Display System").)

92. Miller further discloses that the performance management data is displayed according to information indicating which type of display object to use as well as whether any alarms have been triggered (i.e. the "**according to information indicating the operations to be performed on the performance management data**"). As I explained above, the string "gateway_ok 0 X" will tell the satool system to display an alarm if the network gateway is not functioning, and to display the performance management data as text. (*See* Miller, e.g., 122 ("The GUI . . . uses configuration files . . . to specify thresholds for data values, and to set the type of display objects. The display system also checks for data

values out of bounds and notifies the user when an alarm threshold has been crossed.") (left column, under "Display System"), 126 ("The variable name is followed by a threshold value and its default display object. A threshold value of 0 for a Boolean variable triggers the alarm when the gateway is down; a text display object is specified.") (left column, under "GUI") (emphasis added to both).) Miller thus discloses or suggests each limitation of claim 1.

**2. Claim 2 is Disclosed or Suggested by Miller**

93. Claim 2 depends from independent claim 1 and recites:

> The method defined in claim 1 further comprising:
>
> a performance management system receiving at least one script-based program; and
>
> running the script-based program via the performance management system to periodically collect the performance management data from components in a network.

As explained in **Part VI.B.1** above, Miller discloses each limitation of claim 1. The additional limitations of claim 2 are also disclosed or suggested by Miller.

**(a) "a performance management system receiving at least one script-based program"**

94. The first limitation of dependent claim 2 recites "a performance

management system receiving at least one script-based program." The claimed "**performance management system**" in Miller corresponds to the satool system. The satool system in Miller comprises (among other things) software that runs on the server (satoold), and the SNMP agent software that runs on monitored components. (Miller, at 120 ("**satool** is made up of three distinct components: a data gathering agent, a data collecting server, and a display system.") (underlining added, bold in original).)

95. As I explained previously, the whole point of the satool system is to allow a sysadmin to manage the performance of a large number of machines in a distributed environment. (Miller, e.g., at 119 ("Monitoring large numbers of machines in a distributed environment can be time consuming and inefficient. . . . **satool** provides a way to efficiently monitor groups of machines and find problems and potential problems quickly in a straightforward manner.") (bold in original) (left column, under "Introduction").)

96. Miller further discloses that the satool system (the "performance management system") receives a "**script-based program**." Miller explains that the satool system can be "extended" by adding helper scripts that SNMP agents use to

collect performance management data.

97. In the section titled "Extending satool," Miller explains that "[a]dding a new variable to monitor is fairly easy. . . . The first step is to write a helper script that the SNMP agent will call." (Miller, at 125 (left column).) As I explained in **Part VI.A.1**, the helper script is then incorporated into the satool system to provide the specified data to the system. (*Id.*, e.g., at 125-26.) The "**traceroute-helper**" script, as I described above, is an example of a "script-based program." (*Id.* at 125 (left column, under "Agent").) Miller thus discloses this limitation.

**(b) "running the script-based program via the performance management system to periodically collect the performance management data from components in a network"**

98. Miller explains that the satool system runs the script-based program to collect the performance management data from components in a network. As explained above, Miller discloses C source code that causes SNMP agent to execute the "traceroute-helper" script that collects information about whether a computer can reach an outside computer via the network gateway. (Miller, Ex. 1003, at 125 (right column).) The script collects the performance management data. (Miller, e.g., 121 ("The agent now supports variables defined by the **satool**

MIB. The values for most of the variables are obtained via 'helper scripts' that run standard UNIX commands and parse the output into a form that the agent can use.") (right column, under "satool Variables").)

99. Miller further discloses that the collection of performance management data can occur "**periodically**." In particular, the satool server ("satoold") "polls clients for data via SNMP at configurable intervals." (Miller, at 120 (right column, under "Data Gathering") (emphasis added).) "The polling interval for most hosts is specified by the *interval default* entry. It defaults to 300 seconds if not specified. Individual host values specified using *interval hostname* entries override the default." (Miller, at 121 (italics in original; left column, under "satoold Configuration File").) As noted above, when the computer is polled, its SNMP agent runs the specified helper scripts to collect the performance management data. (Miller, e.g., 125 ("[a]dding a new variable to monitor is fairly easy. . . . The first step is to write a helper script that the SNMP agent will call.") (left column), 121 ("The agent now supports variables defined by the **satool** MIB. The values for most of the variables are obtained via 'helper scripts' that run standard UNIX commands and parse the output into a form that the agent can

use.") (right column, under "satool Variables").) Miller thus discloses or suggests claim 2.

### 3. Claim 3 is Disclosed or Suggested by Miller

100. Claim 3 depends from claim 2 and recites:

> The method defined in claim 2 further comprising:
>
> integrating the at least one script-based program into the performance management system as a service monitor; and
>
> using the service monitor to periodically collect the performance data.

101. As explained in **Part VI.B.2** above, Miller discloses or suggests each limitation of claim 2. Miller also discloses or suggests the additional limitations of claim 3, as explained below.

#### (a) "integrating the at least one script-based program into the performance management system as a service monitor"

102. Miller discloses "integrating the at least one script-based program into the performance management system as a service monitor." As I explained above for claim 2, Miller explains that helper scripts are added to the satool system to collect particular performance management data. As I explained for claim 2, helper

scripts qualify as the claimed "script-based program."

103. A helper script can also qualify as a "service monitor" because a helper script is used for "monitoring a device, application or server in a network." For example, as I explained in **Part VI.A.1** above, Miller discloses an example of extending the satool system using a helper script to include information regarding whether a gateway is working correctly. Because the gateway provides a service—i.e., it enables another computer to connect to computers outside the network—incorporating the helper script into the satool system qualifies as integrating that script into the system as a "service monitor."

**(b) "using the service monitor to periodically collect the performance data"**

104. Miller also discloses "using the service monitor to periodically collect the performance data." As I explained above for claim 2, the satool server periodically polls the SNMP agent of a computer for performance management data. The SNMP agent in turn uses helper scripts to collect that data. Because, as I explained above, a helper script is integrated as a "service monitor" into the satool system, Miller discloses this limitation. Miller therefore discloses or suggests claim 3.

#### 4. Claim 4 is Disclosed or Suggested by Miller

105. Claim 4 depends from claim 3 and recites "[t]he method defined in claim 3 further comprising the performance management system analyzing and integrating the performance management data." As explained in **Part VI.B.3** above, Miller discloses or suggests each limitation of claim 3. Miller also discloses or suggests the additional limitations of claim 4, as explained below.

106. In particular, as I explained for claim 2 above, the claimed "performance management system" corresponds in Miller to the satool system. The satool system analyzes and integrates performance management data in number of ways.

107. First, as explained previously, Miller discloses helper scripts that collect performance management data, put it into a form that is usable by the satool system, and provide the data in that usable form to the rest of the satool system. (Miller, e.g., at 120 ("[The SNMP agent] uses 'helper scripts' to massage data from standard UNIX commands.") (left column, under "Components"), 121 ("The values for most variables are obtained via 'helper scripts' that run standard UNIX commands and parse the output into a form that the agent can use.") (right column,

under "satool Variables").)

108. Second, Miller explains that the satool server receives performance management data via SNMP from monitored computers. The server analyzes an SNMP message from a monitored computer and integrates the performance management data in that message into a database that it maintains. (Miller, e.g., 120 ("The server polls agents at set intervals and stores the resulting data into an *ndbm*(3) database.") (emphasis in original) (left column, under "Components"), 120 ("**satoold** polls clients for data via SNMP at configurable intervals.") (emphasis in original) (right column, under "Data Gathering").)

109. Miller also discloses that the satool system analyzes and integrates performance management data in a third way. As I explained previously, the satool system includes a display system that requests performance management data from the satool server. The display system analyzes the data returned from the server in order to integrate that information in a graphical display by, for example, determining which display object to use for a given piece of data and whether an alarm should be triggered. (Miller, e.g., at 119 ("After traversing the hierarchy to an individual machine, the display contains visual widgets representing the health

of that machine.") (right column), 122 ("**satool's** display objects, currently a sliding histogram, a thermometer, and a text object, are used to view the data in the database.") (right column, under "Display Objects"), 123 ("The sysadmin can configure their **satool** display to select the hosts whose data will be displayed, types of widgets to use, alarm thresholds and actions, and screen layout.") (right column, under "Configuration").) Miller thus discloses or suggests claim 4.

### 5. Claim 5 is Disclosed or Suggested by Miller

110. Claim 5 depends from claim 3 and recites "[t]he method defined in claim 3 further comprising the performance management system generating an alarm when the performance management data meets certain criteria." As explained in **Part VI.B.3** above, Miller discloses or suggests each limitation of claim 3. Miller also discloses or suggests the additional limitations of claim 5.

111. As I discussed previously, Miller explains that the satool system includes alarm functionality. "**satool's** alert system notifies the user when data values cross stated thresholds. The data object causing the alert will change appearance (color, reverse video, new icon, etc.)." (Miller, at 125 (underlining added, bold in original) (left column, top); *see also id.* at 126 ("The variable name

is followed by a threshold value and its default display object. A threshold value of 0 for a boolean variable triggers the alarm when the gateway is down.") (emphasis added) (left column, under "GUI").) Miller thus discloses claim 5.

### 6. Claim 6 is Disclosed or Suggested by Miller

112. The preamble of independent claim 6 recites "[a] method for providing an interface between a user and a performance management system, the performance management system being connected with a network, the network including a plurality of components coupled by a plurality of connections, the performance management system collecting data of the components." Because of the length of the preamble, I have divided it into separate portions to address below.

#### (a) [Preamble] "[a] method for providing an interface between a user and a performance management system"

113. Miller discloses that the satool system—which as explained previously corresponds to the claimed performance management system—includes a GUI that "acts as the interface between the user and the [satool system] database." (Miller, at 119 (Abstract).) Miller further explains that the GUI is "intended to be run on a sysadmin's workstation or dedicated management

station," thereby providing a user (e.g., a sysadmin) an interface to the satool system. (Miller, at 122 (left column, under "Display System").) Miller thus discloses this portion of the preamble.

**(b) [Preamble] "the performance management system being connected with a network"**

114. Miller also discloses a performance management system connected with a network. The whole point of the satool system described in Miller is to manage computers in a distributed environment, which the system does in part by periodically contacting via SNMP other computers on a network. (Miller, e.g., 119 (Introduction), 120 ("**satoold** polls clients for data via SNMP at configurable intervals.") (emphasis in original) (right column, under "Data Gathering"), 120 ("We wanted **satool** to be used on both small and large networks.") (emphasis in original) (left column).)

**(c) [Preamble] "the network including a plurality of components coupled by a plurality of connections"**

115. Miller also discloses a network that includes "a plurality of components coupled by a plurality of connections." Miller explains that the satool server repeatedly connects to monitored computers on a network using SNMP. (Miller, e.g., 119 ("**satool** provides a way to efficiently monitor groups of

machines and find problems and potential problems quickly in a straightforward manner.") (emphasis in original) (left column, under "Introduction"), 120 ("**satoold** polls clients for data via SNMP at configurable intervals.") (emphasis in original) (right column, under "Data Gathering"), 120 ("We wanted **satool** to be used on both small and large networks.") (emphasis in original) (left column).)

**(d) [Preamble] "the performance management system collecting data of the components"**

116. Miller also discloses that the satool system collects data of the computers that it monitors. Miller explains that the server "polls clients for data via SNMP at configurable intervals" and stores the collected data from each computer in a database. (Miller, at 120 (left column, under "Data Gathering"), 119 ("**satool** is composed three independent parts: . . . a database that collects data from each client machine") (underlining added, bold in original) (Abstract), 120 ("**satool** is made up of three distinct components: a data gathering agent, a data collecting server, and a display system.") (underlining added, bold in original) (left column, under "Components").)

117. Thus, Miller discloses the preamble of independent claim 6.

**(e) "receiving at least one script-based program from the user, the script-based programs defining data types not provided by the performance management system" (Claim 6[a])**

118. The first limitation of claim 6 recites "receiving at least one script-based program from the user, the script-based programs defining data types not provided by the performance management system."

119. The first portion of this limitation is similar to the first limitation of claim 3, but further recites that the script-based program is received "from the user." This difference is not material because in my analysis for claim 3 the script-based program was received from the user. In particular, as I explained for claim 3, Miller discloses that the satool system can be "extended" by a user by adding helper scripts that SNMP agents use to collect performance management data. In the section titled "Extending satool," Miller explains that "[a]dding a new variable to monitor is fairly easy. . . . The first step is to write a helper script that the SNMP agent will call." (Miller, at 125 (left column).) As I explained in **Part VI.A.1**, the helper script is then incorporated into the satool system to provide the specified data to the system. (Miller, e.g., at 125-26.) The "traceroute-helper" script provides one example of a script that can be incorporated into the satool system.

(*Id*. at 126-26 (starting at "Extending satool").) Miller thus discloses the first portion of this limitation.

120. The second portion of this limitation recites, "the script-based programs defining data types not provided by the performance management system." Miller discloses this limitation. Miller explains that a key reason to use helper scripts—the claimed "script-based programs"—is to collect data for data types that are not otherwise provided by the performance management system and to put that data into a format that the system can use:

> We have extended the CMU [Carnegie-Mellon University] SNMP 1.1b agent and MIB to include variables of interest to UNIX system administrators not included in the standard network or host MIBs. . . . The values for most of the variables are obtained via "helper scripts" that run standard UNIX commands and parse the output into a form that the agent can use.

(Miller, at 121 (emphasis added) (left column, under "Config File" and "satool Variables").) Because, as the excerpt above explains, a helper script has to understand the output of a UNIX command in order to parse it and put it into a form that the rest of the satool system can use, a helper script defines a type of data

associated with the output of that UNIX command. As a specific example, Miller explains that a helper script called "traceroute-helper" carries out a traceroute routine (Miller, at 125 (left column, under "Extending satool")), thereby defining a traceroute data type that the script knows how to process into a form compatible with the rest of the satool system (*Id.*).

121. Miller therefore discloses the first limitation of claim 6.

**(f) "integrating the program to the performance management system as a service monitor, the performance management system using the service monitor to periodically collect data of the defined data types from the components" (Claim 6[b])**

122. The second limitation of claim 6 is substantially the same as claim 3, as shown in the table below:

| Claim 3 | Claim 6[b] |
|---|---|
| integrating the at least one script-based program into the performance management system as a service monitor; and | integrating the program to the performance management system as a service monitor, |
| using the service monitor to periodically collect the performance data | the performance management system using the service monitor to periodically collect data of the defined data types from the components |

123. While claim 6, unlike claim 3, specifically recites that it is the

performance management system that "us[es] the service monitor," this difference is not material because in my analysis for claim 3, it was the satool server (i.e., the performance management system) that used the helper scripts that were integrated into the satool system. (Miller, e.g., at 120 ("**satoold** polls clients for data via SNMP at configurable intervals.") (bold in original) (right column, under "Data Gathering"), 121 ("The values for most of the variables are obtained via 'helper scripts' that run standard UNIX commands and parse the output into a form that the agent can use.") (right column, under "satool Variables").

124. Thus, for the reasons explained above for claim 3, Miller discloses this limitation, and therefore discloses or suggests claim 6.

### 7. Claim 7 is Disclosed or Suggested by Miller

125. Claim 7 depends from independent claim 6 and recites "[t]he method defined in claim 6 wherein the data collected by the service monitor is business-oriented performance management data of the components." As explained in **Part VI.B.6** above, Miller discloses or suggests each limitation of claim 6. As explained below, the limitations added by claim 7 were obvious over Miller in view of knowledge of a person of ordinary skill in the art.

126. As I explained in **Part V.C.5** above, the term "business-oriented performance management data" under its broadest reasonable interpretation is "performance management data relating to a business." In my opinion, this claim limitation adds nothing of patentable significance to claim 6, from which claim 7 depends. As I explain below, one of ordinary skill in the art would have appreciated that "performance management data" of any type could be "business-oriented" depending on the purpose for collecting that data.

127. As I explained previously, the performance management system disclosed in Miller collects data (using the "**traceroute-helper**" script) that includes performance management data indicating whether or not a network gateway, which allows computers on the network to access other computers in the outside world, is operable. (Miller, at 125 (left column, under "Extending satool").)

128. It would have been obvious to a person of ordinary skill in the art that performance management data indicating the health of a network connection could qualify as "business-oriented" performance management data. This is because the loss of network connectivity could disrupt the operations of a business. Miller, for

example, explains that a problem with a network or a network link could result in "frustrated user calls." (Miller, Ex. 1003, at 119 (Abstract).) One of ordinary skill in the art would have understood that these "frustrated user calls" could result in potential business disruption.

129. In fact, by the filing of the application for the '898 patent in August 2000, entire industries had developed for the purpose of providing network connectivity to customers. Entities known as Internet Service Providers or "ISPs" provided Internet connectivity to their customers in exchange for a fee. As explained in the Microsoft Computer Dictionary (3d ed. 1997), an ISP is "[a] business that supplies Internet connectivity to individuals, businesses, and other organizations." (Ex. 1010, at 267 (definition of "ISP") (emphasis added).) But in order for an ISP to serve its business purpose, its network gateway must be able to connect the ISP customers to other computers on the Internet.

130. It would have been obvious to a person of ordinary skill in the art that monitoring for connectivity to the Internet, such as for an ISP, could have served a business purpose. If an ISP's servers could not connect to the Internet, for example, this might result in the "frustrated user calls" described in Miller, and if

not corrected, loss of revenue and customers. For all of these reasons, therefore, all limitations of claim 7 are disclosed or suggested by Miller based on the knowledge of a person of ordinary skill in the art.

### 8. Claim 8 is Disclosed or Suggested by Miller, Kernighan and O'Reilly

131. Claim 8 depends from independent claim 6 and recites:

> The method defined in claim 6, further comprising:
>
> **[a]** receiving input from the user, the input specifying a rate at which the service monitor polls the components;
>
> **[b]** receiving input from the user, the input specifying names, types and units of input parameters and output variables of the script-based program;
>
> **[c]** using the input from the user to setup the program as a service monitor of the performance management system.

132. As explained in **Part VI.B.6** above, Miller discloses or suggests each limitation of claim 6. The limitations added by claim 8 are disclosed or suggested by Miller in view of Brian W. Kernighan et al., The C Programming Language (2d ed. 1998) **[Ex. 1004]** ("**Kernighan**"), and Tim O'Reilly et al., Windows 98 in a Nutshell (1999) **[Ex. 1005]** ("**O'Reilly**").

**(a) "receiving input from the user, the input specifying a rate at which the service monitor polls the components" (Claim 8[a])**

133. Miller discloses the first limitation of claim 8, "receiving input from the user, the input specifying a rate at which the service monitor polls the components." Miller explains that the satool server ("satoold") "polls clients for data via SNMP at configurable intervals." (Miller, at 120 (emphasis added) (right column, under "Data Gathering").) Miller discloses that a user of the satool system configures the polling rate by modifying the configuration file to specify a polling interval:

> Parameters to **satoold** can be configured through its configuration file (/usr/local/etc/satoold.conf by default). The polling interval for most hosts is specified by the *interval default* entry. It defaults to 300 seconds if not specified. Individual host values specified using *interval hostname* entries override the default.

(Miller, at 121 (underlining added, bold in original) (left column, under "satoold Configuration File").) One of ordinary skill in the art would have understood that the configuration file is modified by the user (e.g. by using a text editor), indicating that the polling rate is based on input from the user.

**(b) "receiving input from the user, the input specifying names, types and units of input parameters and output variables of the script-based program" (Claim 8[b])**

134. The second limitation of claim 8 recites, "receiving input from the user, the input specifying names, types and units of input parameters and output variables of the script-based program." This limitation is disclosed and suggested by Miller in view of Kernighan and O'Reilly.

135. The step of "**receiving input**" recited in this limitation takes place when the user writes a helper script program and integrates it into the satool system. An example of such a helper script is the "traceroute-helper" script discussed extensively below. As I will explain below, it would have been obvious to one of ordinary skill in the art that writing the helper script and integrating it would have involved receiving input specifying the names, types and units of the input parameters and output variables for the helper script.

136. Miller explains that the user has to write the helper script, copy it to monitored computers, update the SNMP agent MIB for the monitored computers, and update satool display system. Each of these steps involves the receipt of input from the user. (Miller, e.g., 125 ("The first step is to write a helper script that the

SNMP agent will call. . . . The script must be placed in a directory in the agents [sic] path. The next step is to add the new variable to the MIB.") (left column, under "Agent") (emphasis added), 126 ("The final step is to add the new variable to the GUI. It must be added to the default list in the procedure BuildList, file IO.tcl.") (left column, under "GUI"); *see also id.* at 125 ("Following is a list of steps we took to include a new variable . . . . In each case, the area where code needs to be added is marked with EXTEND_HERE in a comment.").) As I explained in detail above, one example of a "script-based program" in Miller is the "**traceroute-helper**" script that determines whether or not a network gateway is working properly. (*Id.* at 125 (under "Extending satool").)

137. Claim 8 further recites that the received input specifies "names, types and units" of (1) "input parameters" and (2) "output variables," of the "script-based program." I will address each of these elements in turn.

138. **Input Parameters**: As noted, Miller states that the user writes the helper script, but does not specifically describe its input parameters. But it would have been obvious to one of ordinary skill in the art that in the process of writing a helper script, such as the "traceroute-helper" script, the user would specify the

"names, types and units" of input parameters, as recited in the claim.

139. In fact, specifying the "names, types and units" of "input parameters" is often an intrinsic part of writing any computer program. For example, Kernighan explains that a computer program can contain input parameters known as "command-line arguments." (Kernighan, Ex. 1004, at p. 114, § 5.10 ("In environments that support C, there is a way to pass command-line arguments or parameters to a program when it begins executing.") (emphasis added).) Command-line arguments are well-known to persons of ordinary skill in the art, and in fact, almost any computer user who has ever used a command-line interface such as the UNIX or Windows command shell.

140. Kernighan explains that the instructions in the computer program identify the command-line arguments ("input parameters"), and further, provide variable names to the arguments (such as "argv[]"), and define the data types for them (such as a "char" or character string). (*Id.* at 114-16.) Kernighan provides several examples of computer programs that manipulate command-line arguments. (*Id.* at 115 (program for identifying command line arguments and echoing them back to the user), 116-17 (program for specifying text pattern in command line

arguments to be searched within other input data).) One of ordinary skill in the art would have found it obvious, therefore, to specify and describe the input parameters when a computer program is being written. Although Kernighan describes command-line arguments by reference to the C programming language, those teachings are applicable to any other programming language that supports command-line arguments, including Perl.

141. One concrete example of the use of "input parameters" is disclosed by O'Reilly, which discloses a command-line program called "tracert," which is a Windows version of the "traceroute" program described in Miller. (O'Reilly, at pp. 389-90.) The "tracert" program contains a number of command-line arguments including "***target***" which specifies the "[d]estination system whose connectivity you are checking," and "***-w timeout***," which specifies the "[n]umber of milliseconds to wait for a reply before going to the next hop." (*Id.* at 390.) These parameters may be entered by the user through the command line.

142. Rationale to Combine Miller, Kernighan and O'Reilly: It would have been obvious to one of ordinary skill in the art to combine Miller with Kernighan and O'Reilly. This would have predictably resulted in the satool system of Miller

in which the "traceroute-helper," when written by the user, included receiving user input specifying the "names, types and units" of "input parameters," as recited in the claim. Allowing a computer program to accept and process "input parameters" such as command-line arguments was well-known to persons of ordinary skill in the art, and part of the basic knowledge of computer programmers generally. One of ordinary skill in the art would have appreciated that in order for a computer program such as a helper script to properly utilize input parameters, the programmer had to input instructions that name each input parameter and define the data type for the parameter. O'Reilly further confirms that input parameters could have associated "units," such as command-line arguments specifying a timeout value in "milliseconds." (O'Reilly, at 390 ("***-w timeout*** Number of milliseconds to wait for a reply before going to the next hop.") (bold and italics in original).) The computer program in the traceroute example must extract the numerical value in the command line argument (e.g. "-w 200") and apply the appropriate unit to the value (e.g. 200 milliseconds, not 200 seconds). Kernighan and O'Reilly both confirm that a computer program can have a multitude of input parameters of different names, types, and units, depending on the needs of the

program. It therefore would have been obvious to one of ordinary skill in the art to receive input from the user specifying names, types and units of input parameters for a script-based program, such as the "traceroute-helper" script in Miller.

143. One of ordinary skill in the art would have also found Miller, Kernighan and O'Reilly to be combinable. To begin with, Miller explains that the "traceroute-helper" script invokes the "traceroute" program in order to perform its task. (Miller, at 124 ("The script simply does a *traceroute* (8) to the machine enss.ucar.edu and prints 1 of it was successful or 0 if there was a problem. . . .") (italics in original).) Because O'Reilly describes a version of the traceroute program discussed in Miller, a person of ordinary skill in the art would have found it natural to consult O'Reilly to learn more about the program to better understand the "traceroute-helper" script described in Miller.

144. One of ordinary skill in the art would also have found Kernighan combinable with Miller and O'Reilly. Kernighan is a famous textbook on computer programming, and its discussion of command-line arguments falls within the general knowledge of persons of ordinary skill in the art. Kernighan further discloses that the C programming language "has been closely associated with the

UNIX system where it was developed, since both the system and most of the programs that run on it were written in C." (Kernighan, Ex. 1004, at p. 1 ("Introduction").) One of ordinary skill in the art would have found it informative to consult Kernighan's discussion of command-line arguments to better understand how input parameters – such as the input parameters in the traceroute program of O'Reilly – are named, given data types and are otherwise processed by the traceroute program and the SNMP agent.

145. **Output Variables**: Miller in view of Kernighan and O'Reilly also disclose and suggest receiving input from a user specifying the names, types and units of "**output variables**" of the script-based program. The prior art discloses and suggests this limitation for many of the same reasons as the "input parameters" discussed above.

146. Miller specifically discloses that helper scripts output variables that can be processed by the SNMP agent. (Miller, Ex. 1003, at 121 (right column, under "satool Variables").) For example, Miller describes a technique for expediting operation of the satool system in which "the script output[s] all of the variables we might be interested in at once and cache[s] the values." (*Id.*

(emphasis added).) Miller further discloses that a user links the output from a helper script to a particular user-defined MIB object, which specifies a name and type for the corresponding output from the helper script. For example, as I discussed in **Part VI.A.1**, Miller discloses a user-input "satoolGatewayOk" MIB object that is linked to the output of the "traceroute-helper" helper script. (Miller, at 125.) That MIB object, like any MIB object, specifies a name ("satoolGatewayOk") and type ("INTEGER"). (*Id.*; *see also id.*, at 121 (describing the content of a MIB object) (right column, under "Sample MIB Entry").)

147. Miller does not appear to expressly disclose receiving input from a user that specifies "units" of "output variables" of the script-based program, but as I explained above, this would have been obvious to a person of ordinary skill in the art. O'Reilly, for example, specifically discloses the ability to output numerical data specifying the amount of time in milliseconds required for a particular connection to a server. (O'Reilly, at 390 (e.g., "119ms").) Just like the specification of input parameters, the defining of output parameters and the units for those parameters is determined by the instructions input by the programmer when he or she wrote the computer program. It would have been obvious to a

person of ordinary skill in the art to provide a similar output of unit-based data from the "traceroute-helper" program. For example, if the network gateway is functioning properly, it might be useful to specify the number of milliseconds it took the agent to connect to it as a further measure of network performance.

148. Moreover, Miller makes clear that the satool system does not simply monitor dimensionless quantities (i.e., bare numbers). Each numerical value described in satool has an associated "unit," which must be taken into account by the system in order for the numerical value to be used properly. The satool MIB disclosed in Appendix A of Miller includes, for example, "number of processes," "number of page reclaims / sec." and "percent of cpu in user time," among others. (Miller, at 127-28 (underlining added).) Each of these pieces of data is specified in terms of units—"processes," "page reclaims / sec.," "percent." This unit specification provides helpful information about the nature of the number with which it is associated. It would have required no leap of inventiveness to include the units for a particular user-specified monitored variable as part of the graphical display of information about that variable. As a general matter, nearly everyone should remember their grade-school teachers who repeatedly emphasized the

practice and value of labeling the axes of a graph with corresponding units. Moreover, the whole point of Miller is to provide a system for efficiently monitoring and managing a group of computers. (Miller, e.g., at 119 ("**satool** provides a way to efficiently monitor groups of machines and find problems and potential problems quickly in a straightforward manner.") (left column, under "Introduction").) As I explained above, Miller already discloses that the user is able to associate an alarm threshold and default display object with a variable. (Miller, e.g., 126 (left column, under "GUI").) A person of ordinary skill in the art would have readily recognized the benefit from being able to also specify the units associated with a particular variable displayed in the GUI. Adding this information would have only furthered the objective of the satool system in providing an interface for efficiently monitoring and managing a group of computers by providing additional context and meaning for numbers presented in the display and eliminating potential guesswork.

149. Thus, Miller discloses and suggests receiving input from a user that specifies "names, types and units" of "output variables" "of the script-based program" as recited in claim 8.

150. Miller, Kernighan and O'Reilly thus fully disclose and/or suggest claim 8**[b]**.

**(c) "using the input from the user to setup the program as a service monitor of the performance management system" (Claim 8[c])**

151. The last limitation of claim 8 recites, "using the input from the user to setup the program as a service monitor of the performance management system." Miller discloses this limitation.

152. As I explained for the preceding limitations, the claimed "input" corresponds in Miller to particular information specified by a user in the course of writing a helper script and integrating the script into the satool system. (Miller, e.g., 125-26 (describing a user writing a helper script, defining a MIB object and specifying configuration information for the GUI of display system in order to add a new variable to be monitored by the satool system).) As I explained above for claim 3, a helper script that has been integrated into the satool system qualifies as the claimed "service monitor." Miller thus discloses claim 8**[c]**.

153. Thus, for the above reasons, the prior art discloses or suggests each limitation of claim 8.

### 9. Claim 9 is Disclosed or Suggested by Miller

154. Claim 9 depends from independent claim 6 and recites "[t]he method defined in claim 6 wherein the performance management system analyzes the data collected using the service monitor." As explained in **Part VI.B.6** above, Miller discloses or suggests each limitation of claim 6. Miller also discloses or suggests the additional limitations of claim 9.

155. The limitations of claim 9 are an amalgam of similar limitations recited in claims 3 and 4. Claim 3 relates to using a service monitor to collect performance management data while claim 4 (which depends from claim 3) relates to the performance management system analyzing that data. The reasons I explained above for claims 3 and 4 regarding (1) how Miller discloses using the service monitor to collect performance management data and (2) how Miller discloses the performance management system analyzing that data are equally applicable to claim 9. Thus, for the reasons explained above, Miller discloses claim 9.

### 10. Claim 10 is Disclosed or Suggested by Miller

156. Claim 10 depends from claim 9 and recites "[t]he method defined in claim 9, wherein the performance management system displays at least one graph

of the data collected using the service monitor." As explained in **Part VI.B.9** above, Miller discloses or suggests each limitation of claim 9. Miller also discloses or suggests the additional limitations of claim 10, as explained below.

157. Miller explains that data collected by the satool system can be presented graphically in a number of forms, including a "sliding histogram." (Miller, at 122 (right column, under "Display Objects").) A histogram is a well-known type of graph. Miller explains that "[w]e call a histogram which displays the values from the last *n* (10 by default) time intervals a sliding histogram." (*Id.*)

158. I explained in claim 3 above that one particular example of a "service monitor" disclosed in Miller relates to collecting information regarding whether a gateway is working correctly. In that example, the display object is specified as being a text display object, not a sliding histogram object. (Miller, e.g., at 126 ("The final step is to add the new variable to the GUI. . . . [A] text display object is specified.") (left column, under "GUI").)

159. But it would have been obvious to a person of ordinary skill in the art to use the sliding histogram object of Miller (a "graph") to display the operability of the network gateway. Such functionality is already present and described in the

satool system, and would have predictably resulted in gateway status information displayed using a sliding histogram. A person of ordinary skill in the art could readily have adapted the satool system in Miller to display the output of the "traceroute-helper" script as a graph, for example, by changing the display parameters for the "gateway_ok" variable. (Miller, at 126 (left column, under "GUI").) One of ordinary skill in the art would have been motivated to at least try using a sliding histogram instead of a text display object to enjoy the added benefit of being able to see at a glance when a gateway became operable or inoperable through the sliding histogram's display of historical information. Persons of ordinary skill in the art would have understood the potential benefit of displaying information as a graph instead of text.

160. Miller thus discloses or suggests each limitation of claim 10.

**11. Claim 11 is Disclosed or Suggested by Miller**

161. Claim 11 depends from claim 9 and recites "[t]he method defined in claim 9 wherein the performance management system generates at least one report of the data collected using the service monitor, and/or other types of data collected by the performance management system." As explained in **Part VI.B.9** above,

Miller discloses or suggests each limitation of claim 9. Miller also discloses or suggests the additional limitations of claim 11.

162. In particular, as amply discussed earlier, the satool system of Miller corresponds to the claimed "performance management system." This system generates reports of data collected using helper scripts in the form of graphical displays of that data generated by the display system. (Miller, e.g., at 122 ("The display system provides an X-based Graphical User Interface (GUI) for viewing the contents of the **satool** database.") (bold in original) (left column, under "Display System"), 119 ("[T]he display contains visual widgets representing the health of that machine.") (right column), 121 ("The values for most of the variables are obtained via 'helper scripts' that run standard UNIX commands and parse the output into a form that the agent can use.") (right column, under "satool Variables").) Miller explains, for example, that display objects report various pieces of information about collected data, such as the current data value, the maximum and minimum running values, the running average and the value over time (using a sliding histogram). (Miller, at 122-123 (describing sliding histogram, thermometer and text display objects).)

### 12. Claim 12 is Disclosed or Suggested by Miller

163. Claim 12 depends from claim 9 and recites "[t]he method defined in claim 9, wherein the performance management system generates an alarm when the data collected using the service monitor does not meet certain criteria, the certain criteria being provided to the performance management system by the user." As explained in **Part VI.B.9** above, Miller discloses or suggests each limitation of claim 9. Miller also discloses or suggests the additional limitations of claim 12.

164. As explained above for dependent claim 5, Miller explains that the satool system includes functionality to trigger an alarm when a specified threshold is crossed. "**satool's** alert system notifies the user when data values cross stated thresholds. The data object causing the alert will change appearance (color, reverse video, new icon, etc.)." (Miller, at 125 (underlining added, bold in original) (left column, top).)

165. Miller further discloses that the threshold for triggering an alarm is specified by the user: "The sysadmin can configure their **satool** display to select the hosts whose data will be displayed, types of widgets to user, alarm thresholds

and actions, and screen layout." (Miller, at 123 (underlining added, bold in original) (right column, under "Configuration"); *see also id.* at 126 ("The final step is to add the new variable to the GUI. . . The variable name is followed by a threshold value and its default display object. A threshold value of 0 for a boolean variable triggers the alarm when the gateway is down.") (emphasis added) (left column, under "GUI").) Miller thus discloses or suggests claim 12.

**C. Enablement**

166. In my opinion, Miller, Kernighan and O'Reilly provide sufficient detail to have enabled a person of ordinary skill in the art to practice the methods described in claims 1-12 of the '898 patent. As I explained in **Part IV**, SNMP and scripting languages were well-known to persons of ordinary skill in the art, and documentation on those technologies was widely available. These were standard features of UNIX-based systems, and as explained in Nemeth, "[m]ost UNIX systems come with enough documentation to kill a cow." (Nemeth, Ex. 1006, at xxv.)

167. In addition, as Miller notes, the software underlying the satool system was largely already written by others and available for use:

> In designing **satool**, we also wanted to *reuse* tools where possible to avoid reinventing the wheel. To this end **satool** takes advantage of many standard UNIX utilities to gather data. We also use the **tcl** and **tk** languages from John Ousterhout at the University of California, Berkeley for the display and SNMP 1.1b agent and sysadmin MIB (Management Information Base) from Carnegie Mellon University.

(Miller, at 120 (emphasis in original) (left column, under "Components").) To create a system that practiced the claims of the '898 patent would have required relatively little additional work. Miller explains, for example, that "**Tcl/Tk** was chosen for the **satool** project (over **Interview** and **Suit**) because it is mature and easy to use" and even goes so far as to provide a "fairly easy" step-by-step tutorial. (Miller, at 122 (left column, under "Tcl/Tk"), 125-26.)

168. Moreover, the credentials of the lead author of Miller closely track the level of ordinary skill in the art that I identified in **Part III** above. (Miller, at 126 ("Todd Miller is a recent graduate of the University of Colorado, Boulder, where he received a Bachelors Degree in Computer Science and served as a systems administrator for the last two years he spent there.") (right column, under "Author Information").) Likewise, Kernighan and O'Reilly describe techniques for

handling input parameters and the operation of the traceroute program, respectively, which were well-known long before August 2000.

169. Thus, in my opinion, the prior art provides sufficient disclosure to enable a person of ordinary skill in the art to have practiced the methods of claims 1-12 without undue experimentation.

## VII. CONCLUSION

170. As explained above, claims 1-12 of the ’898 patent, generally speaking, purport to disclose methods for collecting performance management data from monitored computers on a network and displaying that information. The data collection can be performed by a “script-based program.”

171. The claims do not describe anything that was new or non-obvious by August 2000, the earliest date listed on the face of the ’898 patent. In my opinion, each element of those claims is disclosed and/or suggested by the prior art described above. In particular, claims 1-7 and 9-12 are disclosed and/or suggested by Miller. Claim 8 is disclosed and/or suggested by Miller in view of Kernighan and O’Reilly.

172. As explained in detail above, Miller discloses a system called “satool” for monitoring and managing the performance of a large number of computers in a distributed environment. The satool system uses “helper scripts” that collect data from monitored computers and displays that data to a system administrator. A user of satool can extend the functionality of the system by writing his or her own helper scripts and integrating them into the satool system. Miller provides an

example of integrating a "traceroute-helper" helper script into the satool system to collect data regarding the operation of a network gateway. The data can be collected periodically and is displayed in accordance with configuration information that specifies the type of display object to use and associated alarm thresholds.

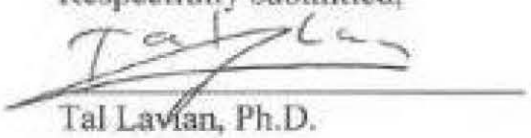
173. Kernighan and O'Reilly describe basic aspects of computer programming and administration, including input parameters and output variables of computer programs such as script-based programs and the traceroute program. As I explained in **Part V.B.8** above, it would have been obvious to combine Miller, Kernighan and O'Reilly.

174. In signing this Declaration, I recognize that the Declaration will be filed as evidence in a contested case before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I also recognize that I may be subject to cross-examination in this proceeding. If required, I will appear for cross-examination at the appropriate time. I reserve the right to offer opinions relevant to the invalidity of the '898 patent claims at issue and/or offer testimony in support of this Declaration.

175. I hereby declare under penalty of perjury that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under 28 U.S.C. § 1001.

Dated: July 17, 2015

Respectfully submitted,



Tal Lavian, Ph.D.

# EXHIBIT A

# Tal Lavian, Ph.D.

http://telecommnet.com
http://cs.berkeley.edu/~tlavian
tlavian@telecommnet.com

1640 Mariani Dr.
Sunnyvale, CA 94087
(408)-209-9112

## Research and Consulting: Telecommunications, Network Communications, and Mobile Wireless technologies

- Scientist, educator, and technologist with over 25 years of experience
- Co-author on over 25 scientific publications, journal articles, and peer-reviewed papers
- Named inventor on over 80 issued and filed patents
- Industry fellow and lecturer at UC Berkeley Engineering – Center for Entrepreneurship and Technology (CET)

### EDUCATION

- **Ph.D.**, Computer Science specializing in networking and communications, UC Berkeley
- **M.Sc.**, Electrical Engineering, Tel Aviv University
- **B.Sc.**, Mathematics and Computer Science, Tel Aviv University

### EXPERTISE

Network communications, telecommunications, Internet protocols and mobile wireless:

- **Communication networks:** Internet Protocols; TCP/IP suite; TCP; UDP; IP; VoIP; Ethernet; network protocols; network software applications; Data Link, Network, and Transport Layers (L2, L3, L4)
- **Internet Software:** Internet software applications; distributed computing; cloud computing; Web applications; FTP; HTTP; Java; C; C++; client server; file transfer; multicast; streaming media
- **Routing/switching:** LAN; WAN; VPN; routing protocols; RIP; BGP; MPLS;OSPF; IS-IS;DNS; QoS; switching; packet switching; network infrastructure; network communication architectures
- **Mobile Wireless:** Wireless LAN; 802.11; cellular systems; mobile devices; smartphone technologies

### LITIGATION SUPPORT SERVICES

- Expert witness in numerous USPTO PTAB – Inter Partes Review (IPR) and CBM cases
- Expert witness in Federal courts and the ITC (over 30 cases)
- Expert reports, depositions, and courtroom testimonies
- Skilled articulation of technical material for both technical and non-technical audiences
- Product and technology analysis, patent portfolios, claim charts, patentability research
- Litigation support and technology education in patent disputes
- Past cases involved Cisco, Juniper, HP, Ericsson, Microsoft, Google, Samsung and Apple

## ACCOMPLISHMENTS

- Selected as Principal Investigator for three US Department of Defense (DARPA) projects
- Led research project on networking computation for the US Air Force Research Lab (AFRL)
- Led and developed the first network resource scheduling service for grid computing
- Led wireless research project for an undisclosed US federal agency
- Managed and engineered the first demonstrated transatlantic dynamic allocation of 10Gbs Lambdas as a grid service
- Spearheaded the development of the first demonstrated wire-speed active network on commercial hardware
- Invented over 80 patents; over 50 prosecuted *pro se* in front of the USPTO
- Created and chaired Nortel Networks' EDN Patent Committee
- Current IEEE Senior Member

## PROFESSIONAL EXPERIENCE

**University of California, Berkeley,** Berkeley, CA 2000-Present

**Berkeley Industry Fellow, Lecturer, Visiting Scientist, Ph.D. Candidate, Nortel's Scientist Liaison**

*Some positions and projects were concurrent, others sequential*

- Serves as an Industry Fellow and Lecturer at the Center for Entrepreneurship and Technology (CET).
- Studied network services, telecommunication systems and software, communications infrastructure, and data centers
- Developed long-term technology for the enterprise market, integrating communication and computing technologies
- Conducted research projects in data centers (RAD Labs), telecommunication infrastructure (SAHARA), and wireless systems (ICEBERG)
- Acted as scientific liaison between Nortel Research Lab and UC Berkeley, providing tangible value in advanced technologies
- Earned a Ph.D. in Computer Science with a specialization in communications and networking

**Telecomm Net Consulting, Inc.** (Innovations-IP) Sunnyvale, CA 2006-Present

**Principal Scientist**

- Consulting in the areas of network communications, telecommunications, Internet protocols, and smartphone mobile wireless devices

- Providing architecture and system consultation for software projects relating to computer networks, mobile wireless devices, Internet web technologies
- Acting as an expert witness in network communications patent infringement lawsuits

**VisuMenu, Inc.** – Sunnyvale, CA 2010-Present

**Co- Founder and Chief Technology Officer (CTO)**

- Design and develop architecture of visual IVR technologies for smartphones and wireless mobile devices in the area of network communications
- Design crawler/spider system for IVR / PBX using Asterisk, SIP and VoIP
- Deploy the system as cloud networking and cloud computing utilizing Amazon Web Services (EC2, S3, VPC, DNS, and RDS)

**Ixia,** Santa Clara, CA 2008-2008

**Communications Consultant**

- Researched and developed advanced network communications testing technologies:
  • IxNetwork/IxN2X — tests IP routing and switching devices and broadband access equipment. Provides traffic generation and emulation for the full range of protocols: routing, MPLS, layer 2/3 VPNs, Carrier Ethernet, broadband access, and data center bridging.
  • IxLoad — quickly and accurately models high-volume video, data, and voice subscribers and servers to test real-world performance of multiservice delivery and security platforms.
  • IxCatapult — emulates a broad range of wireless access and core protocols to test wireless components and systems. When combined with IxLoad, provides an end-to-end solution for testing wireless service quality.
  • IxVeriWave — employs a client-centric model to test Wi-Fi and wireless LAN networks by generating repeatable large-scale, real-world test scenarios that are virtually impossible to create by any other means.
  • Test Automation — provides simple, comprehensive lab automation to help test engineering teams create, organize, catalog, and schedule execution of tests.

**Nortel Networks,** Santa Clara, CA 1996 - 2007

*Originally employed by Bay Networks, which was acquired by Nortel Networks*

**Principal Scientist, Principal Architect, Principal Engineer, Senior Software Engineer**

- Held scientific and research roles at Nortel Labs, Bay Architecture Labs, and in the office of the CTO

**Principal Investigator for US Department of Defense (DARPA) Projects**

- Conceived, proposed, and completed three research projects: Active Networks, DWDM-RAM, and a networking computation project for Air Force Research Lab (AFRL)
- Led a wireless research project for an undisclosed US federal agency

**Academic and Industrial Researcher**

- Analyzed new technologies to reduce risks associated with R&D investment
- Spearheaded research collaboration with leading universities and professors at UC Berkeley, Northwestern University, University of Amsterdam, and University of Technology, Sydney
- Evaluated competitive products relative to Nortel's products and technology
- Proactively identified prospective business ideas, which led to new networking products
- Predicted technological trends through researching the technological horizon and academic sphere
- Developed software for switches, routers and network communications devices
- Developed systems and architectures for switches, routers, and network management
- Researched and developed the following projects:
  - Data-Center Communications: network and server orchestration 2006-2007
  - DRAC: SOA-facilitated L1/L2/L3 network dynamic controller 2003-2007
  - Omega: classified wireless project for undisclosed US Federal Agency 2006
  - Open Platform: project for the US Air Force Research Laboratory (AFRL) 2005
  - Network Resource Orchestration for Web Services Workflows 2004-2005
  - Proxy Study between Web/Grids Services and Network Services 2004
  - Streaming Content Replication: real-time A/V media multicast at edge 2003-2004
  - DWDM-RAM: US DARPA-funded program on agile optical transport 2003-2004
  - Packet Capturing and Forwarding Service on IP and Ethernet traffic 2002-2003
  - CO2: content-aware agile networking 2001-2003
  - Active Networks: US DARPA-funded research program 1999-2002
  - ORE: programmable network service platform 1998-2002
  - JVM Platform: Java on network devices 1998-2001
  - Web-Based Device Management: network device management 1996-1997

**Technology Innovator and Patent Leader**

- Created and chaired Nortel Networks' EDN Patent Committee
- Facilitated continuous stream of innovative ideas and their conversion into intellectual property rights
- Developed intellectual property assets through invention and analysis of existing technology portfolios

**Aptel Communications**, Netanya, Israel 1994-1995

**Software Engineer, Team Leader**

*Start-up company focused on mobile wireless CDMA spread spectrum PCN/PCS*

- Developed a mobile wireless device using an unlicensed band [Direct Sequence Spread Spectrum (DSSS)]
- Designed and managed a personal communication network (PCN) and personal communication system (PCS), the precursors of short text messages (SMS)
- Designed and developed network communications software products (mainly in C/C++)
- Brought a two-way paging product from concept to development

**Scitex Ltd.**, Herzeliya, Israel 1990-1993

**Software Engineer, Team Leader**

*Software and hardware company acquired by Hewlett Packard (HP)*

- Developed system and network communications (mainly in C/C++)
- Invented Parallel SIMD Architecture
- Participated in the Technology Innovation group

**Shalev**, Ramat-HaSharon, Israel 1987-1990

*Start-up company*

**Software Engineer**

- Developed real-time software and algorithms (mainly in C/C++ and Pascal)

## PROFESSIONAL ASSOCIATIONS

- IEEE Senior Member
- IEEE CNSV co-chair Intellectual Property SIG (2013)
- President Next Step Toastmasters (an advanced TM club in the Silicon Valley) (2013)
- Technical Co-Chair, IEEE Hot Interconnects 2005 at Stanford University
- Member, IEEE Communications Society (COMMSOC)
- Member, IEEE Computer Society
- Member, IEEE Systems, Man, and Cybernetics Society
- Member, IEEE-USA Intellectual Property Committee
- Member, ACM, ACM Special Interest Group on Data Communication (SIGCOM)
- Member, ACM Special Interest Group on Hypertext, Hypermedia and Web (SIGWEB)
- Member, IEEE Consultants' Network (CNSV)
- Global Member, Internet Society (ISOC)
- President Java Users Group – Silicon Valley Mountain View, CA,1999-2000
- Toastmasters International

### ADVISORY BOARDS

- Quixey (present) – search engine for wireless mobile apps
- Mytopia – mobile social games
- iLeverage – Israeli Innovations

### PROFESSIONAL AWARDS

- Top Talent Award – Nortel
- Top Inventors Award – Nortel EDN
- Certified IEEE-WCET - Wireless Communications Engineering Technologies
- Toastmasters International - Competent Communicator (twice)
- Toastmasters International - Advanced Communicator Bronze

## Patents and Publications

*(Not an exhaustive list)*

### Patents Issued:

- **US 8,688,796** Rating system for determining whether to accept or reject objection raised by user in social network
- **US 8,572,303** Portable universal communication device
- **US 8,553,859** Device and method for providing enhanced telephony
- **US 8,548,131** Systems and methods for communicating with an interactive voice response system
- **US 8,537,989** Device and method for providing enhanced telephony
- **US 8,341,257** Grid proxy architecture for network resources
- **US8,161,139** Method and apparatus for intelligent management of a network element
- **US 8,146,090** Time-value curves to provide dynamic QoS for time sensitive file transfer
- **US 8,078,708** Grid proxy architecture for network resources
- **US 7,944,827** Content-aware dynamic network resource allocation
- **US7,860,999** Distributed computation in network devices
- **US 7,734,748** Method and apparatus for intelligent management of a network element
- **US 7,710,871** Dynamic assignment of traffic classes to a priority queue in a packet forwarding device
- **US 7,580,349** Content-aware dynamic network resource allocation
- **US 7,433,941** Method and apparatus for accessing network information on a network device
- **US 7,359,993** Method and apparatus for interfacing external resources with a network element
- **US 7,313,608** Method and apparatus for using documents written in a markup language to access and configure network elements
- **US 7,260,621** Object-oriented network management interface

- **US 7,237,012** Method and apparatus for classifying Java remote method invocation transport traffic
- **US 7,127,526** Method and apparatus for dynamically loading and managing software services on a network device
- **US7,047,536** Method and apparatus for classifying remote procedure call transport traffic
- **US7,039,724** Programmable command-line interface API for managing operation of a network device
- **US6,976,054** Method and system for accessing low-level resources in a network device
- **US6,970,943** Routing architecture including a compute plane configured for high-speed processing of packets to provide application layer support
- **US6,950,932** Security association mediator for Java-enabled devices
- **US6,850,989** Method and apparatus for automatically configuring a network switch
- **US6,845,397** Interface method and system for accessing inner layers of a network protocol
- **US6,842,781** Download and processing of a network management application on a network device
- **US6,772,205** Executing applications on a target network device using a proxy network device
- **US6,564,325** Method of and apparatus for providing multi-level security access to system
- **US6,175,868** Method and apparatus for automatically configuring a network switch
- **US6,170,015** Network apparatus with Java co-processor
- **US 8,619,793** Dynamic assignment of traffic classes to a priority queue in a packet forwarding device
- **US 8687,777** Systems and methods for visual presentation and selection of IVR menu
- **US 8,681,951** Systems and methods for visual presentation and selection of IVR menu

- **US 8,625,756** Systems and methods for visual presentation and selection of IVR menu
- **US 8,594,280** Systems and methods for visual presentation and selection of IVR menu
- **US 8,548,135** Systems and methods for visual presentation and selection of IVR menu
- **US 8,406,388** Systems and methods for visual presentation and selection of IVR menu
- **US 8,345,835** Systems and methods for visual presentation and selection of IVR menu
- **US 8,223,931** Systems and methods for visual presentation and selection of IVR menu
- **US 8,160,215** Systems and methods for visual presentation and selection of IVR menu
- **US 8,155,280** Systems and methods for visual presentation and selection of IVR menu
- **US 8,054,952** Systems and methods for visual presentation and selection of IVR menu
- **US 8,000,454** Systems and methods for visual presentation and selection of IVR menu
- **EP 1,905,211** Technique for authenticating network users
- **EP 1,142,213** Dynamic assignment of traffic classes to a priority queue in a packet forwarding device
- **EP 1,671,460** Method and apparatus for scheduling resources on a switched underlay network
- **CA 2,358,525** Dynamic assignment of traffic classes to a priority queue in a packet forwarding device

**Patent Applications Published and Pending:**
*(Not an exhaustive list)*

- **US 20140105025** Dynamic Assignment of Traffic Classes to a Priority Queue in a Packet Forwarding Device
- **US 20140105012** Dynamic Assignment of Traffic Classes to a Priority Queue in a Packet Forwarding Device
- **US 20140012991** Grid Proxy Architecture for Network Resources
- **US 20130080898** Systems and Methods for Electronic Communications
- **US 20130022191** Systems and Methods for Visual Presentation and Selection of IVR Menu
- **US 20130022183** Systems and Methods for Visual Presentation and Selection of IVR Menu
- **US 20130022181** Systems and Methods for Visual Presentation and Selection of IVR Menu
- **US 20120180059** Time-Value Curves to Provide Dynamic QOS for Time Sensitive File Transfers
- **US 20120063574** Systems and Methods for Visual Presentation and Selection of IVR Menu
- **US 20110225330** Portable Universal Communication Device
- **US 20100220616** Optimizing Network Connections
- **US 20100217854** Method and Apparatus for Intelligent Management of a Network Element
- **US 20100146492** Translation of Programming Code
- **US 20100146112** Efficient Communication Techniques
- **US 20100146111** Efficient Communication in a Network
- **US 20090313613** Methods and Apparatus for Automatic Translation of a Computer Program Language Code
- **US 20090313004** Platform-Independent Application Development Framework
- **US 20090279562** Content-aware dynamic network resource allocation
- **US 20080040630** Time-Value Curves to Provide Dynamic QoS for Time Sensitive File Transfers
- **US 20070169171** Technique for authenticating network users
- **US 20060123481** Method and apparatus for network immunization
- **US 20060075042** Extensible Resource Messaging Between User Applications and Network Elements in a Communication Network

- **US 20050083960** Method and Apparatus for Transporting Parcels of Data Using Network Elements with Network Element Storage
- **US 20050076339** Method and Apparatus for Automated Negotiation for Resources on a Switched Underlay Network
- **US 20050076336** Method and Apparatus for Scheduling Resources on a Switched Underlay Network
- **US 20050076173** Method And Apparatus for Preconditioning Data to Be Transferred on a Switched Underlay Network
- **US 20050076099** Method and Apparatus for Live Streaming Media Replication in a Communication Network
- **US 20050074529** Method and apparatus for transporting visualization information on a switched underlay network
- **US 20040076161** Dynamic Assignment of Traffic Classes to a Priority Queue in a Packet Forwarding Device
- **US 20020021701** Dynamic Assignment of Traffic Classes to a Priority Queue in a Packet Forwarding Device
- **WO 2007/008976** Technique for Authenticating Network Users
- **WO 2006/063052** Method and apparatus for network immunization
- **WO2000/0054460** Method and apparatus for accessing network information on a network device

## Publications

*(Not an exhaustive list)*

- "R&D Models for Advanced Development & Corporate Research" Understanding Six Models of Advanced R&D - Ikhlaq Sidhu, Tal Lavian, Victoria Howell - University of California, Berkeley. Accepted paper for 2015 ASEE Annual Conference and Exposition- June 2015
- "Communications Architecture in Support of Grid Computing",Tal Lavian,Scholar's Press 2013ISBN 978-3-639-51098-0.
- "Applications Drive Secure Lightpath Creation across Heterogeneous Domains, Feature Topic Optical Control Planes for Grid Networks: Opportunities, Challenges and the Vision." Gommans L.; Van Oudenaarde B.; Dijkstra F.; De Laat C.; Lavian T.; Monga I.; Taal A.; Travostino F.; Wan A.; IEEE *Communications Magazine*, vol. 44, no. 3, March 2006, pp. 100-106.
- *Lambda Data Grid: Communications Architecture in Support of Grid Computing*. Tal I. Lavian, Randy H. Katz; Doctoral Thesis, University of California at Berkeley. January 2006.
- "Information Switching Networks." Hoang D.B.; T. Lavian;*The 4th Workshop on the Internet, Telecommunications and Signal Processing, WITSP2005*, December 19-21, 2005, Sunshine Coast, Australia.
- "Impact of Grid Computing on Network Operators and HW Vendors." Allcock B.; Arnaud B.; Lavian T.; Papadopoulos P.B.; Hasan M.Z.; Kaplow W.; *IEEE Hot Interconnects at Stanford University 2005*, pp.89-90.
- *DWDM-RAM: AData Intensive Grid Service Architecture Enabled by Dynamic Optical Networks*. Lavian T.; Mambretti J.; Cutrell D.; Cohen H.J; Merrill S.; Durairaj R.; Daspit P.; Monga I.; Naiksatam S.; Figueira S.; Gutierrez D.; Hoang D.B., Travostino F.; CCGRID 2004, pp. 762-764.
- *DWDM-RAM: AnArchitecture for Data Intensive Service Enabled by Next Generation Dynamic Optical Networks*. Hoang D.B.; Cohen H.; Cutrell D.; Figueira S.; Lavian T.; Mambretti J.; Monga I.; Naiksatam S.; Travostino F.; Proceedings IEEE Globecom 2004, Workshop on High-Performance Global Grid Networks, Houston, 29 Nov. to 3 Dec. 2004, pp.400-409.
- *Implementation of a Quality of Service Feedback Control Loop on Programmable Routers*. Nguyen C.; Hoang D.B.; Zhao, I.L.; Lavian, T.; Proceedings, 12th IEEE International Conference on Networks 2004. (ICON 2004) Singapore, Volume 2, 16-19 Nov. 2004, pp.578-582.
- *A Platform for Large-Scale Grid Data Service on Dynamic High-Performance Networks*. Lavian T.; Hoang D.B.; Mambretti J.; Figueira S.; Naiksatam S.; Kaushil N.; Monga I.; Durairaj R.; Cutrell D.; Merrill S.; Cohen H.; Daspit P.; Travostino F; GridNets 2004, San Jose, CA., October 2004.
- *DWDM-RAM: Enabling Grid Services with Dynamic Optical Networks*. Figueira S.; Naiksatam S.; Cohen H.; Cutrell D.; Daspit, P.; Gutierrez D.; Hoang D. B.; Lavian T.; Mambretti J.; Merrill S.; Travostino F; Proceedings, 4th IEEE/ACM International Symposium on Cluster Computing and the Grid, Chicago, USA, April 2004, pp. 707-714.
- *DWDM-RAM: Enabling Grid Services with Dynamic Optical Networks*. Figueira S.; Naiksatam S.; Cohen H.; Cutrell D.; Gutierrez D.; Hoang D.B.; Lavian T.; Mambretti J.; Merrill S.; Travostino F.; 4th IEEE/ACM International Symposium on Cluster Computing and the Grid, Chicago, USA, April 2004.

- *An Extensible, Programmable, Commercial-Grade Platform for Internet Service Architecture*. Lavian T.; Hoang D.B.; Travostino F.; Wang P.Y.; Subramanian S.; Monga I.; IEEE Transactions on Systems, Man, and Cybernetics on Technologies Promoting Computational Intelligence, Openness and Programmability in Networks and Internet Services Volume 34, Issue 1, Feb. 2004, pp.58-68.
- *DWDM-RAM: An Architecture for Data Intensive Service Enabled by Next Generation Dynamic Optical Networks*. Lavian T.; Cutrell D.; Mambretti J.; Weinberger J.; Gutierrez D.; Naiksatam S.; Figueira S.; Hoang D. B.; Supercomputing Conference, SC2003 Igniting Innovation, Phoenix, November 2003.
- *Edge Device Multi-Unicasting for Video Streaming*. Lavian T.; Wang P.; Durairaj R.; Hoang D.; Travostino F.; Telecommunications, 2003. ICT 2003. 10th International Conference on Telecommunications, Tahiti, Volume 2, 23 Feb.-1 March, 2003 pp. 1441-1447.
- The SAHARA Model for Service Composition Across Multiple Providers. Raman B.; Agarwal S.; Chen Y.; Caesar M.; Cui W.; Lai K.; Lavian T.; Machiraju S.; Mao Z. M.; Porter G.; Roscoe T.; Subramanian L.; Suzuki T.; Zhuang S.; Joseph A. D.; Katz Y.H.; Stoica I.; Proceedings of the First International Conference on Pervasive Computing. ACM Pervasive 2002, pp. 1-14.
- *Enabling Active Flow Manipulation in Silicon-Based Network Forwarding Engines*. Lavian T.; Wang P.; Travostino F.; Subramanian S.; Duraraj R.; Hoang D.B.; Sethaput V.; Culler D.; Proceeding of the Active Networks Conference and Exposition, 2002.(DANCE) 29-30 May 2002, pp. 65-76.
- *Practical Active Network Services within Content-Aware Gateways*. Subramanian S.; Wang P.; Durairaj R.; Rasimas J.; Travostino F.; Lavian T.; Hoang D.B.; Proceeding of the DARPA Active Networks Conference and Exposition, 2002.(DANCE) 29-30 May 2002, pp. 344-354.
- *Active Networking on a Programmable Network Platform.* Wang P.Y.; Lavian T.; Duncan R.; Jaeger R.; Fourth IEEE Conference on Open Architectures and Network Programming (OPENARCH), Anchorage, April 2002.
- *Intelligent Network Services through Active Flow Manipulation*. Lavian T.; Wang P.; Travostino F.; Subramanian S.; Hoang D.B.; Sethaput V.; IEEE Intelligent Networks 2001 Workshop (IN2001), Boston, May 2001.
- Intelligent Network Services through Active Flow Manipulation. Lavian T.; Wang P.; Travostino F.; Subramanian S.; Hoang D.B.; Sethaput V.; Intelligent Network Workshop, 2001 IEEE 6-9 May 2001, pp.73 - 82.
- *Enabling Active Flow Manipulation in Silicon-based Network Forwarding Engine*. Lavian, T.; Wang, P.; Travostino, F.; Subramanian S.; Hoang D.B.; Sethaput V.; Culler D.; Journal of Communications and Networks, March 2001, pp.78-87.
- *Active Networking on a Programmable Networking Platform*. Lavian T.; Wang P.Y.; IEEE Open Architectures and Network Programming, 2001, pp. 95-103.
- *Enabling Active Networks Services on a Gigabit Routing Switch*. Wang P.; Jaeger R.; Duncan R.; Lavian T.; Travostino F.; 2nd Workshop on Active Middleware Services, 2000.

- *Dynamic Classification in Silicon-Based Forwarding Engine Environments*. Jaeger R.; Duncan R.; Travostino F.; Lavian T.; Hollingsworth J.; Selected Papers. 10th IEEE Workshop on Metropolitan Area and Local Networks, 1999. 21-24 Nov. 1999, pp.103-109.
- *Open Programmable Architecture for Java-Enabled Network Devices*. Lavian, T.; Jaeger, R. F.; Hollingsworth, J. K.; IEEE Hot Interconnects Stanford University, August 1999, pp. 265-277.
- *Open Java SNMP MIB API*. Rob Duncan, Tal Lavian, Roy Lee, Jason Zhou, Bay Architecture Lab Technical Report TR98-038, December 1998.
- *Java-Based Open Service Interface Architecture*. Lavian T.; Lau S.; BAL TR98-010 Bay Architecture Lab Technical Report, March 1998.
- *Parallel SIMD Architecture for Color Image Processing*. Lavian T. Tel – Aviv University, Tel – Aviv, Israel, November 1995.
- *Grid Network Services, Draft-ggf-ghpn-netservices-1.0*.George Clapp, Tiziana Ferrari, Doan B. Hoang, Gigi Karmous-Edwards, Tal Lavian, Mark J. Leese, Paul Mealor, Inder Monga, Volker Sander, Franco Travostino, Global Grid Forum(GGF).
- *Project DRAC: Creating an applications-aware network.*Travostino F.; Keates R.; Lavian T.; Monga I.; Schofield B.; Nortel Technical Journal, February 2005, pp. 23-26.
- *Optical Network Infrastructure for Grid, Draft-ggf-ghpn-opticalnets-1.*Dimitra Simeonidou, Reza Nejabati, Bill St. Arnaud, Micah Beck, Peter Clarke, Doan B. Hoang, David Hutchison, Gigi Karmous-Edwards, Tal Lavian, Jason Leigh, Joe Mambretti, Volker Sander, John Strand, Franco Travostino, Global Grid Forum(GGF) GHPN Standard GFD-I.036 August 2004.
- *Popeye - Using Fine-grained Network Access Control to Support Mobile Users and Protect Intranet Hosts.*Mike Chen, Barbara Hohlt, Tal Lavian, December 2000.

## Presentations and Talks

*(Not an exhaustive list)*

- Lambda Data Grid: An Agile Optical Platform for Grid Computing and Data-intensive Applications.
- Web Services and OGSA
- WINER Workflow Integrated Network Resource Orchestration.
- Technology & Society.
- Abundant Bandwidth and how it affects us?
- Active Content Networking(ACN).
- DWDM-RAM:Enabling Grid Services with Dynamic Optical Networks .
- Application-engaged Dynamic Orchestration of Optical Network Resources .
- A Platform for Data Intensive Services Enabled by Next Generation Dynamic Optical Networks .
- Optical Networks.
- Grid Optical Network Service Architecture for Data Intensive Applications.
- Optical Networking & DWDM.
- OptiCal Inc.
- OptiCal & LUMOS Networks.
- Optical Networking Services.
- Business Models for Dynamically Provisioned Optical Networks.
- Business Model Concepts for Dynamically Provisioned Optical Networks.
- Optical Networks Infrastructure.
- Research Challenges in agile optical networks.
- Services and Applications' infrastructure for agile optical networks.
- Impact on Society.
- TeraGrid Communication and Computation.
- Unified Device Management via Java-enabled Network Devices.
- Active Network Node in Silicon-Based L3 Gigabit Routing Switch.
- Active Nets Technology Transfer through High-Performance Network Devices.
- Programmable Network Node: Applications.
- Open Innovation via Java-enabled Network Devices.
- Practical Considerations for Deploying a Java Active Networking Platform.
- Open Java-Based Intelligent Agent Architecture for Adaptive Networking Devices.
- Java SNMP Oplet.
- Open Distributed Networking Intelligence: A New Java Paradigm.
- Open Programmability.
- Active Networking On A Programmable Networking Platform.
- Open Networking through Programmability.
- Open Programmable Architecture for Java-enabled Network Devices.

- Integrating Active Networking and Commercial-Grade Routing Platforms.
- Programmable Network Devices.
- To be smart or not to be?