

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

TESLA INC.,
Petitioner,

IPR2025-00341
U.S. Patent No. 7,181,743

**DECLARATION OF DR. EREZ ZADOK,
UNDER 37 C.F.R. § 1.68 IN SUPPORT OF PETITION FOR
INTER PARTES REVIEW**

TABLE OF CONTENTS

I.	Introduction.....	8
A.	Materials Considered.....	8
II.	Qualifications and Professional Experience.....	10
III.	Level of Ordinary Skill in the Art	21
IV.	Relevant Legal Standards	22
V.	Technical Background.....	23
A.	General Computer Operations.....	23
1.	Networking Overview.....	26
B.	Networked and Distributed Systems.....	30
1.	Clustering, Quality-of-Service, and Load Balancing.....	32
C.	Virtualization.....	35
VI.	Overview of the '743 Patent.....	39
A.	Summary of the '743 patent	39
B.	Prosecution History	46
C.	Priority Date of the '743 Patent.....	47
VII.	Claim Construction.....	49
VIII.	Identification of how the Claims are Unpatentable.....	49
A.	Ground 1: Claims 10, 11, 14, 16, and 18 are obvious under 35 U.S.C. § 103 over Welch alone and in combination with Ravindran.	51
1.	Summary of Welch	51
2.	Summary of Ravindran	59

3. Motivation to Combine Welch and Ravindran .. 60
4. Claim 10. 61
 - (a) [10.0] Software stored on at least one host for converting N networked hosts into a resource managed system instantiating M managed characteristic application computer programs, each managed characteristic application computer program managed by one of the N networked hosts, the software comprising:61
 - (b) [10.1] a first function group which monitors the host and network resources;71
 - (c) [10.2] a second function group which provides application computer program event reporting and event correlation capabilities;77
 - (d) [10.3] a third function group which provides reasoning and decision making capabilities for the resource managed system,.....85
 - (e) [10.4] wherein the third function group comprises: a first function which determines a state and health of the N hosts, a network operatively coupling the N hosts to one another and the M managed characteristic application computer programs in the distributed environment;.....91
 - (f) [10.5] a second function which determines required allocation and reallocation actions needed to

- maintain a plurality of Quality of Service (QoS) requirements established for the M managed characteristic application computer programs, the QoS requirement dictating parameters regarding service quality of the M management characteristic application programs; and100
- (g) [10.6] a third function which generates automatic control signal requests corresponding to the actions dictated by the QoS requirements, such that the managed characteristic application computer programs are moved, shutdown, and started in accordance with satisfaction of the QoS requirements; and106
- (h) [10.7] a fourth function group which provides program control capabilities permitting starting, stopping, and configuring of selected ones of the M managed characteristic application computer programs on respective ones of the N hosts in the resource managed system,.....110
- (i) [10.8] where M and N are positive integers and where M may be equal to, greater than, or less than N.....113
5. Claim 11..... 115
- (a) [11] The software as recited in claim 10, wherein the first function receives system specification information comprising host configuration and capabilities.115

6. Claim 14..... 119
- (a) [14] The software as recited in claim 10, wherein the first function receives program control information comprising application status and detected application faults for each of the M managed characteristic application computer programs, and detected failures regarding the N hosts.....119
7. Claim 16..... 123
- (a) [16] The software as recited in claim 10, wherein the first function receives application performance data representing each one of the M managed characteristic application computer programs.....123
8. Claim 18..... 124
- (a) [18.0] The software as recited in claim 10, wherein the second function which determines the required allocation and reallocation actions established for the M managed characteristic application computer programs by:.....124
- (b) [18.1] responding to application and host failures by determining if and what recovery actions should be taken;124
- (c) [18.2] determining if and where to place new copies of managed characteristic application computer programs or which managed characteristic application computer

	programs should be shutdown when QoS Manager functions indicate that scale up or scale down ations should be taken based on measured application performance and QoS specifications established for the M managed characteristic application computer programs;.....	126
(d)	[18.3] determining where new application computer programs should be placed when requested to do so by the fourth function group; and	129
(e)	[18.4] determining which and how many application computer programs should run based on application system priorities.....	130
B.	Ground 2: Claims 10, 11, 14, 16, and 18 are obvious under 35 U.S.C. § 103 over Welch, Ravindran, and Hosokawa.	132
1.	Hosokawa	133
2.	Motivation to combine Welch, Ravindran, and Hosokawa	137
(a)	The number of applications are independent from the number of hosts.....	138
(b)	Management software executing on a host.....	142
3.	Claim 10	146
(a)	[10.0] Software stored on at least one host for converting N networked hosts into a resource managed system instantiating M managed characteristic application computer	

	programs, each managed characteristic application computer program managed by one of te N networked hosts, the software comprising:	146
(b)	Limitations [10.1]-[10.7]	148
(c)	[10.8] where M and N are positive integers and where M may be equal to, greater than, or less than N.....	148
4.	Claims 11, 14, 16, and 18.....	154
IX.	Conclusion	155

I, Dr. Erez Zadok, do hereby declare as follows:

I. INTRODUCTION

1. I am making this declaration at the request of Tesla Inc. in the matter of the *Inter Partes* Review of U.S. Patent No. 7,181,743 (“the ’743 patent”) to Werme *et al.*

2. My consulting company is being reimbursed for reasonable and customary expenses associated with my work and testimony in this investigation. My compensation is not contingent on the outcome of this matter or the specifics of my testimony, and I have no other interest in this case or the parties thereto.

3. I have been asked to provide my opinions regarding whether claims 10, 11, 14, 16, and 18 (“the Challenged Claims”) of the ’743 patent are unpatentable as they would have been obvious to a person having ordinary skill in the art (“POSITA”) at the time of the alleged invention, in light of the prior art. It is my opinion that all of the limitations of the challenged claims would have been obvious to a POSITA.

A. Materials Considered

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

Exhibit	Description	Abbreviation
1001	U.S. Patent No. 7,181,743 to Werme <i>et al.</i>	'743 patent
1002	Prosecution history of the '743 patent	'743 File History
1004	Zadok Curriculum Vitae	Zadok CV
1005	“Distributed, Scalable, Dependable Real-Time Systems: Middleware Services and Applications.” Lonnie R. Welch et al., Proceedings 13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing, April 1999	Welch
1006	“Specification and Modeling of Dynamic, Distributed Real-time Systems.” Lonnie R. Welch, Binoy Ravindran, et al. IEEE Real-Time Systems Symposium, 72-81, IEEE Computer Society Press, Dec. 1998	Ravindran
1007	U.S. Patent No. 6,088,727 to Hosokawa et al.	Hosokawa
1008	U.S. Patent No. 6,269,396 to Shah et al.	Shah
1009	U.S. Provisional Patent No. 60/207,891	'891 provisional
1016	U.S. Patent Publication No. 2002/0064126 to Bhaatal	Bhaatal
1017	U.S. Patent No. 6,041,306 to Du	Du
1018	U.S. Patent No. 6,742,020 to Dimitroff	Dimitroff

Exhibit	Description	Abbreviation
1019	Microsoft Computer Dictionary, 3 rd ed., 1997, excerpts	Microsoft Computer Dictionary 1997
1020	Microsoft Computer Dictionary, 4 th ed., 1999, excerpts	Microsoft Computer Dictionary 1999

5. In addition, I considered the relevant legal standards, including the standard for obviousness, and any additional authoritative documents as cited in the body of this declaration.

6. Finally, I considered my own knowledge and experience based upon my work in the field of networking as described below.

7. Unless otherwise noted, all emphasis in any quoted material has been added.

II. QUALIFICATIONS AND PROFESSIONAL EXPERIENCE

8. My complete qualifications and professional experience are described in my *Curriculum Vitae*, a copy of which can be found in EX1004. The following is a brief summary of my relevant qualifications and professional experience.

9. I am a Professor in the Computer Science Department at Stony Brook

University (part of the State University of New York (“SUNY”) system). I direct the File-systems and Storage Lab (FSL) at Stony Brook’s Computer Science Department. My research interests include file systems and storage systems, operating systems, transactional systems including database technologies, information technology and system administration, security/privacy and information assurance, networking, energy efficiency, performance and benchmarking, virtualization, cloud systems, compilers, applied machine learning, and software engineering.

10. I studied at a professional high school in Israel, focusing on electrical engineering (“EE”), and graduated in 1982. I spent one more year at the high school’s college division, receiving a special Certified Technician’s degree in EE. I then went on to serve in the Israeli Defense Forces for three years (1983–1986). I received my Bachelor of Science degree in computer science (“CS”) in 1991, my Master’s degree in CS in 1994, and my PhD in CS in 2001—all from Columbia University in New York.

11. When I began my undergraduate studies at Columbia University, I also started working as a student assistant in the various campus-wide computer labs, eventually becoming an assistant to the head labs manager, who was managing all public computer labs on campus. During that time, I also became more involved with research within the CS Department at Columbia University, conducting

research on operating systems, file and storage systems, distributed and networked systems, security, and other topics. I also assisted the CS department's computer administrators in managing the department's computers, which included storage, IT, networking, and cyber-security related duties.

12. In 1991, I joined Columbia University's CS department as a full-time systems administrator, studying towards my MS degree part-time. My MS thesis topic is related to file system reliability, fault tolerance, replication, and failover in mobile networked storage systems using file virtualization. My main duties as a systems administrator involved installing, configuring, and managing many networked servers, proxies, and desktops running several operating systems, as well as network devices setup; this included many software and hardware upgrades, device upgrades, and BIOS firmware/chipset updates/upgrades. My duties also included ensuring reliable, secure, authenticated access to networked systems/storage and licensed software, as well as software updates, security and bug fixes. Examples of servers and their protocols included email (SMTP), file transfer (FTP), domain names (DNS), network file systems (NFS), network news systems (NNTP), and Web (HTTP).

13. In 1994, I left my systems administrator position to pursue my doctoral studies at Columbia University. My PhD thesis topic was on versatile file system development using stackable (virtualized) file systems, with examples in the fields

of security and encryption, efficiency, reliability, and failover. I continued to work part-time as a systems administrator at the CS department, and eventually I was asked to serve as manager to the entire information technology (“IT”) staff. From 1991 to 2001, I was also a member of the faculty-level Facilities Committee that oversaw all IT operations at the CS department.

14. As part of my PhD studies at Columbia, I collaborated on projects to develop advanced AI-like techniques to detect previously unknown viruses (a.k.a. “zero-day malware”), using data mining and rule-based detection. This work led to several highly cited papers (over 1,600 citations for one of the papers alone) and two patents. I also became a Teaching Assistant (“TA”) for a first-ever Computer Security course given at Columbia University’s CS department with Dr. Matt Blaze as instructor.

15. From 1990 to 1998, I consulted for SOS Corporation and HydraWEB Technologies as a systems administrator and programmer, managing data storage use and backup/restore duties, databases, web servers, as well as information assurance and cyber-security (*e.g.*, malware protection, software licensing). From 1994 to 2000, I led projects at HydraWEB Technologies, and then became the Director of Software Development—overseeing the development of several products and appliances such as stateful firewalls and HTTP load-balancers, utilizing network-virtualization and high-availability techniques. From 2009 to

2019, I have consulted for Packet General Networks, a startup specializing in secure, virtualized, network storage and applications' data security in the cloud.

16. In 2001, I joined the faculty of Stony Brook University, a position I have held since that time. In 2002, I joined the Operations Committee, which oversees the IT operations of the CS department at Stony Brook University. From 2006 to 2010, I was the Director of IT Operations of the CS department. My day-to-day duties included setting policies regarding computing, hiring and training new staff, assisting any staff with topics of my specialty, defining requirements for new software/hardware, and purchasing. From 2010 to 2015, I had served as the Co-Chair to the Operations Committee. From 2016 to 2019, I oversaw the IT Operations as the Chair of the Operations Committee. A significant component of these duties included defining and helping implement policies for data management, to ensure the security of users and their data, and data reliability and availability, while minimizing the inconvenience and performance impact to users. I personally helped setup and maintain an initial virtual-host infrastructure in the department. Since late 2019, I've been a member of the department's Executive Committee that also oversees all IT operations.

17. In 2017, I became the department's Graduate Academic Adviser, advising all Master students (over 400 annually on average) and many other graduate students on an assortment of academic matters. In August 2024, I took over as the

department's Graduate Program Director, overseeing the entire graduate CS program (700-800 students annually on average).

18. Since 2001, I have personally configured and managed my own research lab's network. This includes setting up and configuring multiple storage systems (*e.g.*, NFS, CIFS/SMB, NAS), virtual and physical environments, applications such as database (*e.g.*, MySQL, Postgresql), Web servers (*e.g.*, Apache), and mail servers; user access control (*e.g.*, NIS, LDAP), backups and restores, snapshot policies, and more. I've personally installed, configured, changed, replaced parts, and upgraded components in numerous devices including mobile devices, laptops, desktops, and servers, both physical and virtual.

19. Since 1995, I have taught courses on operating systems, storage and file systems, advanced systems programming in Unix/C, systems administration, data structures, data/software security, and more. My courses often use storage, file systems, distributed systems, and system/network security as key teaching principles and practical examples for assignments and projects. I have taught these concepts and techniques to my students, both to my direct advisees as well as in my courses. For example, in my graduate Operating Systems course, I often cover Linux's kernel mechanisms to protect users, applications, and data files, virtual file systems, as well as distributed storage systems (*e.g.*, NFS). And in the System Administration undergraduate course, I covered many topics such as networking, storage, backups,

and configuring complex applications such as mail, web, and database servers.

20. My research often investigates computer systems from many angles: security, efficiency, energy use, scalability, reliability, portability, survivability, usability, ease-of-use, versatility, flexibility, and more. My research gives special attention to balancing five often-conflicting aspects of computer systems: performance, reliability, energy use, security, and ease-of-use.

21. Since joining Stony Brook University in 2001, my group in the File-systems and Storage Lab (“FSL”) has developed many file systems and operating system extensions; examples include a highly-secure cryptographic file system, a portable copy-on-write (“COW”) versioning file system, a tracing file system useful to detect intrusions, a replaying file system useful for forensics, a snapshotting and sandboxing file system, a namespace unification file system (that uses stackable, virtualized, file-based COW), an anti-virus file system, an integrity-checking file system, a load balancing and replication/mirroring file system, network file system extensions for security and performance, distributed secure cloud-based storage systems, transactional key-value stores and file systems, OS-level embedded databases, a compiler to convert user-level C code to in-kernel efficient yet safe code, GCC plugins, stackable file system templates, and a Web-based backup system. Many of these projects used one form of virtualization or another (storage, network, host, etc.). I continue to maintain and release newer versions of some of

these file systems and software.

22. I have published over 120 refereed publications (in ACM, IEEE, USENIX, and more). To date, my publications have been cited more than 10,000 times (as per Google Scholar as of December 14, 2024). My papers cover a wide range of related technologies such as file systems, storage systems, transactional systems, security, clouds and virtualization, performance benchmarking and optimization, energy efficiency, system administration, web systems, and more. I also published a book titled “Linux NFS and Automounter Administration” (Sybex, 2001), covering systems administration topics related to network storage and data security.

23. Some of my research has led to public software releases that have been used worldwide. I have publicly maintained the Amd Berkeley Automounter in a package called “am-utils” since 1992; this software helps administrators manage the multitude of file system mounts on dozens of different Unix systems, especially helping to automate access to multiple NFS/NAS storage volumes. Since 1997, I have maintained and released several stackable (virtualized) file system software projects for Linux, FreeBSD, and/or Sun Solaris, in a package called FiST. One of my stackable file system encryption projects, called Cryptfs, became the basis for IBM’s public release of eCryptfs, now part of Linux. Packet General Networks, for whom I have provided consulting services between 2009 and 2019, licensed another

encryption file system called Ncryptfs. Another popular file system released in 2003, called Unionfs, offers virtual namespace unification, transparent shadow copying (a.k.a. copy-on-write or COW), file system snapshotting (*e.g.*, useful for forensics and disaster recovery), and the ability to save disk space by sharing a read-only copy of data among several computers, among other features.

24. My research and teaching make extensive use of data security features. For example, each time I taught the graduate operating system course, the first homework assignment includes the creation of a new system call that performs new or added functionality, often for encrypting a file or verifying its integrity; many of my other assignments cover topics of user/process access control, anti-virus filtering, and more. Since 2001, over 1,000 graduate students were exposed to these principles directly through my teaching and research at Stony Brook University.

25. Moreover, in an undergraduate course titled “Advanced Systems Programming in Unix/C,” I cover many topics of system security and vulnerabilities, such as the structure of UNIX processes, and memory segments such as the heap and stack. This course covers details of several hundred Linux system calls. Often, the first assignment for this course is to develop a tool to encrypt/decrypt files using advanced ciphers, use digital signatures to certify the cipher keys used, and reliably recover files in case of failures. Since 2001, several hundred undergraduate students were exposed to these principles directly through my teaching and research at Stony

Brook University.

26. In another undergraduate course, System Administration, I taught network configuration, security, and storage configuration and reliability. In a special topics course on Storage Systems, I covered many topics such as data deduplication, RAID, transactional storage, storage hardware including modern Flash based ones, virtual storage, backup/restore, snapshots and continuous data protection (“CDP”), NAS and SAN, and NFS.

27. Overall, in addition to the aforementioned experience, my technical experience relevant to the '743 patent at the time of the alleged invention included the configuration, use, research, development, and/or programming of the following technologies and systems: distributed storage systems; Web-server load balancing (e.g., Web farms); tracing, logging, event collection, and analysis; and quality-of-service software development. For example, in my work for a startup called HydraWEB, I've developed agents to monitor the load on hosts, then report it to a centralized load-balancer that monitored which Web servers were down or under/overloaded, and appropriately redirect Web requests. In another example, my Master's thesis, titled “Discovery and Hot Replacement of Replicated Read-Only File Systems, with Application to Mobile Computing,” investigated techniques for mobile devices to discover resources in a distributed setting, then transparently switch to using the one closest and least loaded.

28. My research has been supported by many federal and state grants as well as industry awards, including an NSF CAREER award, two IBM Faculty awards, two NetApp Faculty awards, a Western Digital award, a Facebook award, several Dell-EMC awards, and several equipment gifts. I received the 2008 SUNY Chancellor's Excellence in Teaching award, and the 2022 SUNY Chancellor's Award for Excellence in Scholarship and Creative Activities (both awards can be given only once in a lifetime). In 2021, I was named an ACM Distinguished Member for "Outstanding Scientific Contributions to Computing."

29. My service record to the community includes serving as the co-chair for the USENIX Annual Technical Conference in 2020 (ATC'20); serving as the co-chair for USENIX File and Storage Technologies (FAST'15) in 2015 and on the FAST Conference Steering Committee from 2015 to 2023; serving on the ACM HotStorage Steering Committee since 2021; and serving as the co-chair in 2012 and on the Steering Committee of the ACM SYSTOR conference since 2012. I have served as an Associate Editor to the ACM Transactions on Storage ("TOS") journal from 2009 to 2022; in 2022, I was named the Editor-in-Chief for ACM's TOS journal.

30. I am a named inventor on four patents, two titled "Systems and Methods for Detection of New Malicious Executables" (U.S. Patent No. 7,487,544, issued February 3, 2009; and U.S. Patent No. 7,979,907, issued July 12, 2011); and two

more titled “Multi-Tier Caching,” (U.S. Patent No. 9,355,109, issued May 31, 2016; and U.S. Patent 9,959,279, issued May 1, 2018).

31. I have been disclosed as a testifying expert in 21 cases (including *inter partes* review (“IPR”) proceedings) in the past four years. I have been deposed 13 times and testified in trial twice.

III. LEVEL OF ORDINARY SKILL IN THE ART

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

33. It is my understanding that the earliest possible priority date for the ’743 patent is May 25, 2000. A person of ordinary skill in the art (“POSITA”) in the field of the ’743 patent, as of May 25, 2000, would have been someone knowledgeable and familiar with the areas of networking and distributed computing that are pertinent to the ’743 patent. That person would have a bachelor’s degree in computer science, computer engineering, electrical engineering, or equivalent training, and approximately 2-3 years of professional experience relating to distributed computing. Lack of work experience can be remedied by additional education, and vice versa.

34. For purposes of this Declaration, in general, and unless otherwise noted, my statements and opinions, such as those regarding my experience and the understanding of a POSITA generally (and specifically related to the references I consulted herein), reflect the knowledge that existed in the field as of the alleged priority date of the '743 patent (*i.e.*, May 25, 2000). Unless otherwise stated, when I provide my understanding and analysis below, it is consistent with the level of a POSITA as of the alleged priority date of the '743 patent.

IV. RELEVANT LEGAL STANDARDS

35. I am not an attorney. In preparing and expressing my opinions and considering the subject matter of the '743 patent, I am relying on certain basic legal principles that counsel have explained to me. These principles are discussed below.

36. I understand that prior art to the '743 patent includes patents and printed publications in the relevant art that predate the priority date of the alleged invention recited in the '743 patent.

37. I have been informed that a claimed invention is unpatentable under 35 U.S.C. § 103 if the differences between the invention and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which the subject matter pertains. I have also been informed by counsel that the obviousness analysis takes into account factual inquiries including the level of ordinary skill in the art, the scope and

content of the prior art, and the differences between the prior art and the claimed subject matter.

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

V. TECHNICAL BACKGROUND

39. In this section, I provide an overview of the background art as understood by a POSITA during the relevant timeframe.

A. General Computer Operations

40. Computers are hardware and software devices that execute instructions. The hardware often includes a central processing unit (“CPU”) or other

microprocessors, typically called “processors,” volatile memory (*e.g.*, RAM (random-access memory), or DRAM (dynamic random-access memory)), non-volatile (persistent) storage (*e.g.*, a hard disk or flash drive), and input/output (“I/O”) devices (*e.g.*, network interface card, mouse, keyboard, display). RAM, hard disks, and flash drives are considered, generally, “storage” or “memory” devices because they can store and hold bits of information (whether persistently or not). Computers can have multiple storage devices. Processors are coupled to main memory using a memory bus and to peripherals such as hard disks and network interface cards using one or more I/O busses.

41. Memory and storage devices have considerable differences in their speeds, capacities, and costs. Non-volatile storage devices such as hard disks have the largest capacity, are least expensive per gigabyte, and are slowest (typically operating at millisecond speeds). Volatile DRAM devices are smaller, more expensive, and faster than hard disks (typically operating at microsecond speeds). A CPU’s own memory caches and operations are the smallest, most expensive, and fastest (typically operating at nanosecond speeds). Overall, CPUs are about 1,000× faster than DRAM, which in turn is about 1,000× faster than hard disks.

42. Computer programs are often written in a high-level human-readable language (*e.g.*, C, C++, Java, Perl, PHP, Python), then translated using a compiler or script processor to machine instructions understood by the CPU (*e.g.*, Intel or

AMD processor). Machine instructions—generally called “software”—are stored in files on persistent media (*e.g.*, HDD or ROM), loaded into DRAM, and then loaded into the CPU where they can be executed. All modern computers operate in this manner, regardless of their purpose: a small laptop, a server in a data center, a mainframe computer, a personal workstation, a proxy device, a gateway, a firewall, a router, an appliance, a virtual machine, a handheld or mobile device, a smart card, a SD card, etc. Computer systems may run multiple software programs, from background servers and daemons to user applications.

43. Users (including administrators) can login to one or more computers (*e.g.*, desktop, laptop, server) and execute one or more applications (*e.g.*, word processing, web browsing, backup tool). Some computers offer external interfaces and APIs to communicate with certain devices and especially peripherals. For example, both a USB storage device as well as a printer use a well-defined “protocol” to allow users to communicate with them through the main computer.

44. “Middleware” is “1. Software that sits between two or more types of software and translates information between them. Middleware can cover a broad spectrum of software and generally sits between an application and an operating system, a network operating system, or a database management system. Examples of middleware include COREA and other object broker programs and network control programs. *See also* COREA. 2. Software that provides a common

application programming interface (API). Applications written using that API will run in the same computer systems as the middleware. An example of this type of middleware is ODNC, which has a common API for many types of databases. *See also* application programming interface, ODBC. 3. Software development tools that enable users to create simple programs by selecting existing services and linking them with a scripting language. *See also* scripting language.” EX1019 (Microsoft Computer Dictionary 1997) at 308.

1. Networking Overview

45. A Local Area Network (“LAN”) is a typically a network at a local site where network latencies are short and physical distances between computers are small. A LAN and its computers are often protected from the rest of the internet using a firewall. EX1019 (Microsoft Computer Dictionary 1997) at 276 (“**LAN** *n.* Acronym for **local area network**. A group of computers and other devices dispersed over a relatively limited area and connected by a communications link that enables any device to interact with any other on the network. LANs commonly include microcomputers and shared resources such as laser printers and large hard disks. The devices on a LAN are known as nodes, and the nodes are connected by cables through which messages are transmitted. *See also* baseband network, broadband network, bus network, collision detection, communications protocol, contention, CSMA/CD, network, ring network, star network, token bus network, token passing,

token ring network. *Compare* wide area network.”).

46. Another form of a local area network is one that operates inside a single computer. A “localhost” is “[t]he name that is used to represent the same computer on which a TCP/IP message originates. An IP packet sent to localhost has the IP address 127.0.0.1 and does not actually go out to the Internet. *See also* IP address, packet (definition 1), TCP/IP.” EX1019 (Microsoft Computer Dictionary 1997) at 287. Often, when users/processes want to communicate with other users/processes, using TCP/IP communications on the same machine, they can use the localhost address: it is considered a trusted, high-speed internal network.

47. Conversely, a Wide-Area Network (“WAN”) refers to computers across the entire Internet, which are often distant from each other, exhibit longer latencies, and afford little protection. EX1019 (Microsoft Computer Dictionary 1997) at 507 (“**wide area network** *n.* A communications network that connects geographically separated areas. *Acronym:* WAN.”).

48. A firewall typically intercepts all traffic between computers in a LAN and the WAN, monitors and controls access, and can filter out undesired traffic. Often, LAN computers are far more trusted than WAN computers; therefore, a firewall can be said to separate the trusted network (LAN or localhost) from the untrusted network (WAN). EX1019 (Microsoft Computer Dictionary 1997) at 197 (“**firewall** *n.* A security system intended to protect an organization’s network against

external threats, such as hackers, coming from another network, such as the Internet. A firewall prevents computers in the organization's network from communicating directly with computers external to the network and vice versa. Instead, all communication is routed through a proxy server outside of the organization's network, and the proxy server decides whether it is safe to let a particular message or file pass through to the organization's network.”).

49. A server is a computer or software that provides a service to other computers, called clients. There are many servers on the Internet, for example, Web servers and Mail servers. EX1019 (Microsoft Computer Dictionary 1997) at 430 (“**server** *n.* 1. On a local area network (LAN), a computer running administrative software that controls access to the network and its resources, such as printers and disk drives, and provides resources to computers functioning as workstations on the network. 2. On the Internet or other network, a computer or program that responds to commands from a client. For example, a file server may contain an archive of data or program files; when a client submits a request for a file, the server transfers a copy of the file to the client. *See also* client/server architecture. *Compare* client (definition 3).”

50. A client is typically an end-point computer or software, sometimes used by a user, that communicates across LAN and WAN networks; a client may contact servers to request some service. A user's desktop, laptop, and handheld device are

examples of clients. (Note that computer can be both a client and a server.) EX1019 (Microsoft Computer Dictionary 1997) at 92 (“**client** *n.* 1. In object-oriented programming, a member of a class (group) that uses the services of another class to which it is not related. *See also* inheritance (definition 1). 2. A process, such as a program or task, that requests a service provided by another program—for example, a word processor that calls on a sort routine built into another program. The client process uses the requested service without having to ‘know’ any working details about the other program or the service itself. *Compare* child (definition 1), descendant (definition 2). 3. On a local area network or the Internet, a computer that accesses shared network resources provided by another computer (called a server). *See also* client/server architecture, server.”).

51. A client/server architecture describes how client computers interact with server computers. EX1019 (Microsoft Computer Dictionary 1997) at 92 (“**client/server architecture** *n.* An arrangement used on local area networks that makes use of distributed intelligence to treat both the server and the individual workstations as intelligent, programmable devices, thus exploiting the full computing power of each. This is done by splitting the processing of an application between two distinct components: a ‘front-end’ client and a ‘back-end’ server. The client component is a complete, stand-alone personal computer (not a ‘dumb’ terminal), and it offers the user its full range of power and features for running

applications. The server component can be a personal computer, a minicomputer, or a mainframe that provides the traditional strengths offered by minicomputers and mainframes in a time-sharing environment: data management, information sharing between clients, and sophisticated network administration and security features. The client and server machines work together to accomplish the processing of the application being used. Not only does this increase the processing power available over older architectures but it also uses that power more efficiently. The client portion of the application is typically optimized for user interaction, whereas the server portion provides the centralized, multiuser functionality. *See also* distributed intelligence.”).

B. Networked and Distributed Systems

52. A single computer’s performance is limited by the physical maximum speeds of its network, processor, memory, and I/O devices. A distributed system, however, improves overall performance by using multiple computers concurrently to process many requests, thus increasing the aggregate throughput manyfold.

53. A single computer also represents a single point of failure: if that one computer is down, off, or broken into, then the service it offers is unavailable. A distributed system, conversely, improves resiliency and security by ensuring that the failure of any one (or even a few) components do not affect the overall service provided. This principle is called “fault tolerance.” EX1019 (Microsoft Computer

Dictionary 1997), p. 190 (“**fault tolerance** *n.* The ability [] to respond to a catastrophic event or fault, such as a power outage or a hardware failure, in a way that ensures that no data is lost and any work in progress is not corrupted. [] In a fault-tolerant network, the system has the ability either to continue the system’s operation without loss of data or to shut the system down and restart it, recovering all processing that was in progress when the fault occurred.”).

54. Generally, a “distributed network” is “[a] network in which processing, storage, and other functions are handled by separate units (nodes) rather than by a single main computer.” EX1019 (Microsoft Computed Dictionary 1997), p. 154.

55. The “Distributed Computing Environment” is “[a] set of standards from the Open Group (formerly the Open Software Foundation) for development of distributed applications that can operate on more than one platform. *Acronym*: DCE. *See also* distributed processing.” EX1019 (Microsoft Computed Dictionary 1997), pp. 153-154.

56. “[D]istributed processing” is “[a] form of information processing in which work is performed by separate computers linked through a communications network. Distributed processing is usually categorized as either plain distributed processing or true distributed processing. Plain distributed processing shares the workload among computers that can communicate with one another. True distributed processing has separate computers perform different tasks in such a way

that their combined work can contribute to a larger goal. The latter type of processing requires a highly structured environment that allows hardware and software to communicate, share resources, and exchange information freely.” EX1019 (Microsoft Computed Dictionary 1997), p. 154.

57. Finally, a “distributed file system” is “[a] file management system in which files may be located on multiple computers connected over a local or wide area network.” EX1019 (Microsoft Computed Dictionary 1997), p. 154.

1. Clustering, Quality-of-Service, and Load Balancing

58. When a distributed system comprises two or more hosts, it becomes more efficient to consider those hosts as a single, managed unit or system. “Clustering” describes “[t]he grouping of multiple servers in a way that allows them to appear to be a single unit to client computers on a network. Clustering is a means of increasing network capacity, providing live backup in case one of the server fails, and improving data security. *See also* cluster (definition 4), server.” EX1020 (Microsoft Computed Dictionary 1999), p. 90.

59. A “cluster” is designed specifically to handle application/system failures, for example: “[] 4. A group of independent network servers that operate- and appear to clients-as if they were a single unit. A cluster network is designed to improve network capacity by, among other things, enabling the servers within a cluster to shift work in order to balance the load. By enabling one server to take

over for another, a cluster network also enhances stability and minimizes or eliminates downtime caused by application or system failure. *See also* client/server architecture.” EX1020 (Microsoft Computed Dictionary 1999), p. 90.

60. A cluster of hosts has to be managed consistently, often by some centralized management software. A “cluster controller” is “[a]n intermediary device that is situated between a computer and a group (cluster) of subsidiary devices, such as terminals on a network, and is used to control the cluster.” EX1020 (Microsoft Computed Dictionary 1999), p. 90.

61. A cluster’s management software often measures and monitors the load on each node as well as each node’s uptime and connectivity, to ensure the overall cluster’s health. The “load” on a system is “1. The total computing burden a system carries at one time. ... 3. In communications, the amount of traffic on a line.” EX1020 (Microsoft Computed Dictionary 1999), p. 271 (definition 1).

62. When the load on some nodes become too high (or a node is deemed down or nonresponsive), a cluster’s management software would attempt to rebalance the load, by moving some applications from more loaded hosts to less loaded ones, shutting down some applications, and/or [re]starting some applications elsewhere.

63. “Load balancing” is “1. In distributed processing, the distribution of activity across two or more servers in order to avoid overloading any one with too

many requests from users. Load balancing can be either static or dynamic. In the former, the load is balanced ahead of time by assigning different groups of users to different servers. In the latter, software refers incoming requests at runtime to whichever server is most capable of handling them. 2. In client/server network administration, the process of reducing heavy traffic flows either by dividing a busy network segment into multiple smaller segments or by using software to distribute traffic among multiple network interface cards working simultaneously to transfer information to a server. 3. In communications, the process of routing traffic over two or more routes rather than one. Such load balancing results in faster, more reliable transmissions.” EX1020 (Microsoft Computed Dictionary 1999), p. 271.

64. Cluster management software, through load-balancing, can monitor and assure the quality of the aggregated service that the cluster provides. “Quality of Service” or “QoS” is “1. Generally, the handling capacity of a system or service; the time interval between request and delivery of a product or service to the client or customer. 2. In computer technology, the guaranteed throughput (data transfer rate) level.” EX1020 (Microsoft Computed Dictionary 1999), p. 367.

65. Many examples of cluster-management software systems existed by 2000, including commercial offering. These had built-in resource- and load-monitoring mechanisms, and load-balancing techniques. In one example, “Windows 2000 Advanced Server” is “Microsoft’s network server for larger organizations.

Designed to replace Windows NT 4.0 Enterprise Edition, it supports up to four-way SMP, large physical memories, and database-intensive work. It integrates clustering and load balancing support. *See also* SMP, Windows.” EX1020 (Microsoft Computed Dictionary 1999), p. 482.

66. In another example, “Windows 2000 Datacenter Server” is “Microsoft’s network server for larger organizations. Considered as the most powerful and functional server operating system ever offered by Microsoft, it supports up to 16-way SMP and up to 64GB of physical memory (depending on system architecture). Like Windows 2000 Advanced Server, it provides both clustering and load balancing services as standard features. It is optimized for large data warehouses, econometric analysis, large-scale simulations in science and engineering, OLTP, and server consolidation projects. *See also* OLTP, SMPWindows.” EX1020 (Microsoft Computed Dictionary 1999), p. 482.

C. Virtualization

67. Among computer scientists, there is an axiom stating that “you can solve almost any problem by adding another level of indirection.” Indeed, virtualization is an indirection technique that makes one thing seem like another; virtualization often presents a physical-world artifact as one that appears as an illusion or “fake.” For something to be “virtual”, it is “[o]f or pertaining to a device, service, or sensory input that is perceived to be what it is not in actuality, usually as

more ‘real’ or concrete than it actually is.” EX1019 (Microsoft Computer Dictionary 1997), p. 497.

68. Indeed, the Microsoft Computer Dictionary includes 35 examples and definitions of virtualization. EX1019 (Microsoft Computer Dictionary 1997), pp. 497-500.

69. In one example, a “virtual machine” is “[s]oftware that mimics the performance of a hardware device, such as a program that allows applications written for an Intel processor to be run on a Motorola chip. *Acronym: VM.*” EX1019 (Microsoft Computer Dictionary 1997), p. 498.

70. In another example, “virtual memory” is “[m]emory that appears to an application to be larger and more uniform than it is. Virtual memory may be partially simulated by secondary storage such as a hard disk. Applications access memory through virtual addresses, are translated (mapped) by special hardware and software onto physical addresses. *Acronym: VM. Also called disk memory. See also paging, segmentation.*” EX1019 (Microsoft Computer Dictionary 1997), p. 498.

71. In yet another example, “virtual reality” is “[a] simulated 3-D environment that a user can experience and manipulate as if it were physical. The user sees the environment on display screens, possibly mounted in a special pair of goggles. Special input devices, such as gloves or suits fitted with motion sensors, detect the user’s actions. *Acronym: VR.*”. EX1019 (Microsoft Computer Dictionary

1997), p. 499.

72. Virtualization had been applied to storage systems, disk devices, and drivers. A “virtual device driver” is “[s]oftware in Windows 95 that manages a hardware or software system resource. If a resource retains information from one access to the next that affects the way it behaves when accessed (for example, a disk controller with its status information and buffers), a virtual device driver must exist for it.” EX1019 (Microsoft Computer Dictionary 1997), p. 498.

73. A “virtual disk” or “RAM disk” is “[s]hort for random access memory disk. A simulated disk drive whose data is actually stored in RAM memory. A special program allows the operating system to read from and write to the simulated device as if it were a disk drive. RAM disks are extremely fast, but they require that system memory be given up for their use. Also, RAM disks usually use volatile memory, so the data stored on them disappears when power is turned off. Many portables offer RAM disks that use battery-backed CMOS RAM to avoid this problem. *See also* CMOS RAM. *Compare* disk cache.” EX1019 (Microsoft Computer Dictionary 1997), pp. 395-396, 498.

74. Virtualization is sometimes said to be “logical” representation of physical reality. “Logical” is “[o]f or pertaining to a conceptual piece of equipment or frame of reference, regardless of how it may be realized physically. *Compare* physical.” EX1019 (Microsoft Computer Dictionary 1997), p. 288 (“logical”

definition 2).

75. A “logical drive” or “logical device” is “[a] device named by the logic of a software system, regardless of its physical relationship to the system. For example, a single floppy disk drive can simultaneously be, to the MS-DOS operating system, both logical drive A and drive B.” EX1019 (Microsoft Computer Dictionary 1997), p. 288.

76. Moreover, one can virtualize files and file systems. This was the subject of much of my graduate-level research and became part of my PhD Dissertation. *See* §II above and EX1004 (Zadok Curriculum Vitae).

77. Virtualization, of course, has been applied to networking as well. A “virtual network” is “[a] part of a network that appears to a user to be a network of its own. For example, an Internet service provider can set up multiple domains on a single HTTP server so that each one can be addressed with its company’s registered domain name. *See also* domain name, HTTP server (definition 1), ISP.” EX1019 (Microsoft Computer Dictionary 1997), p. 499.

78. A “virtual LAN” is “[s]hort for **virtual local area network**. A local area network consisting of groups of hosts that are on physically different segments but that communicate as though they were on the same wire. *See also* LAN.” EX1019 (Microsoft Computer Dictionary 1997), p. 498.

79. One of the most famous uses of a (secure) virtual network is a private

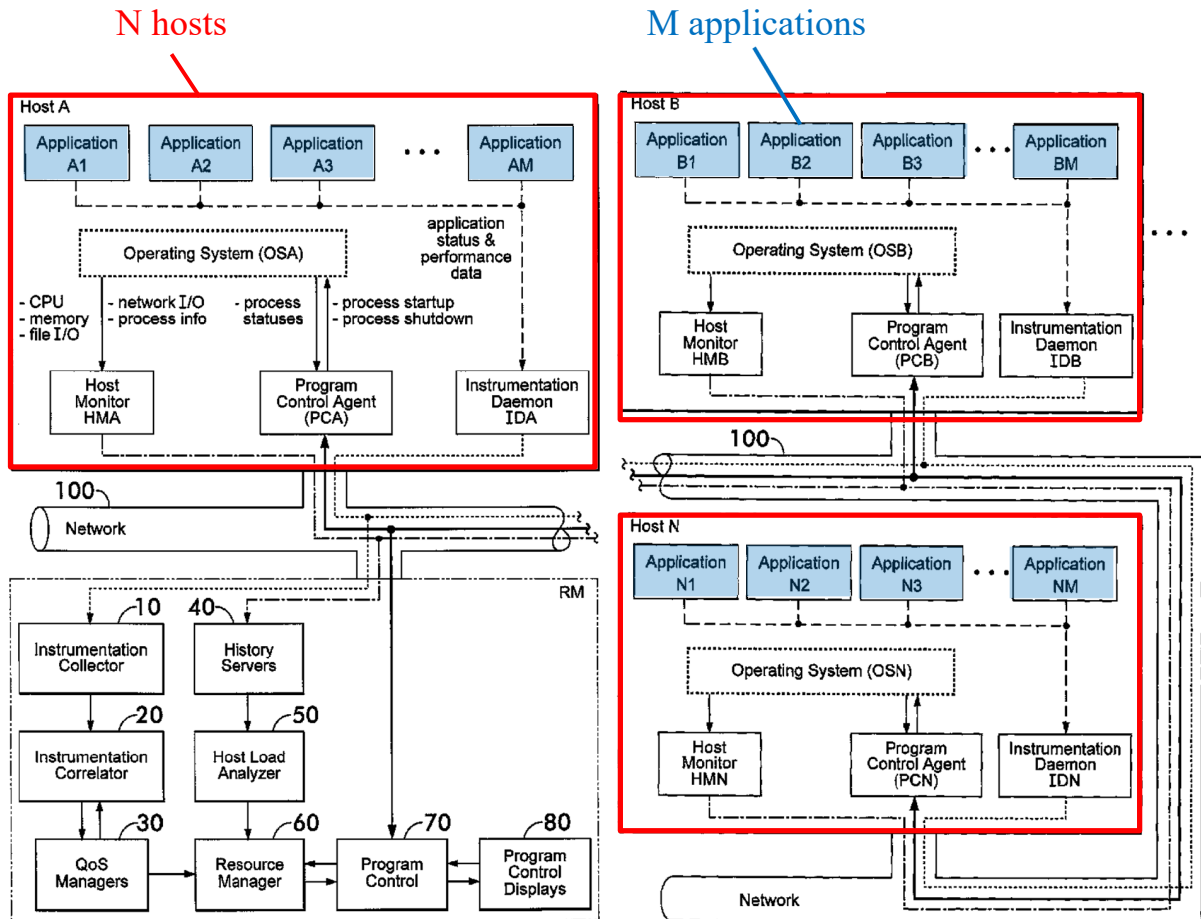
one. A “virtual private network” (“VPN”) is “[a] set of nodes on a public network such as the Internet that communicate among themselves using encryption technology so that their messages are as safe from being intercepted and understood by unauthorized users as if the nodes were connected by private lines. 2. A wide area network formed of permanent virtual circuits (PVCs) on another network, especially a network using technologies such as ATM or frame relay. *Acronym:* VPN. *See also* ATM (definition 1), frame relay, PVC.” EX1019 (Microsoft Computed Dictionary 1997), p. 499. A POSITA would have understood definition 2 of VPN to be a form of an overlay network because it is a collection of individual virtual network segments.

VI. OVERVIEW OF THE ’743 PATENT

A. Summary of the ’743 patent

80. The ’743 patent relates to “a distributed environment comprised of hosts instantiating copies of a scalable application.” EX1001, ’743 patent, 2:46-47. A scalable application is one that uses multiple copies (*i.e.*, instances) of an application that scale up or down to maintain a desired performance. *See e.g.*, EX1001, ’743 patent, 18:12-58. “[T]he present invention is illustrated in FIGS. 1A, 1B, which includes a plurality of Host computers A, B, ..., N operatively connected to one another and Resource Management hardware RM via a Network 100.” EX1001, ’743 patent, 5:31-36. “[I]t will be appreciated that hosts A-N each can

instantiate applications 1-M.” EX1001, ’743 patent, 5:45-49. The N hosts (red) and M applications (blue) are shown below in annotated Figs. 1A and 1B.

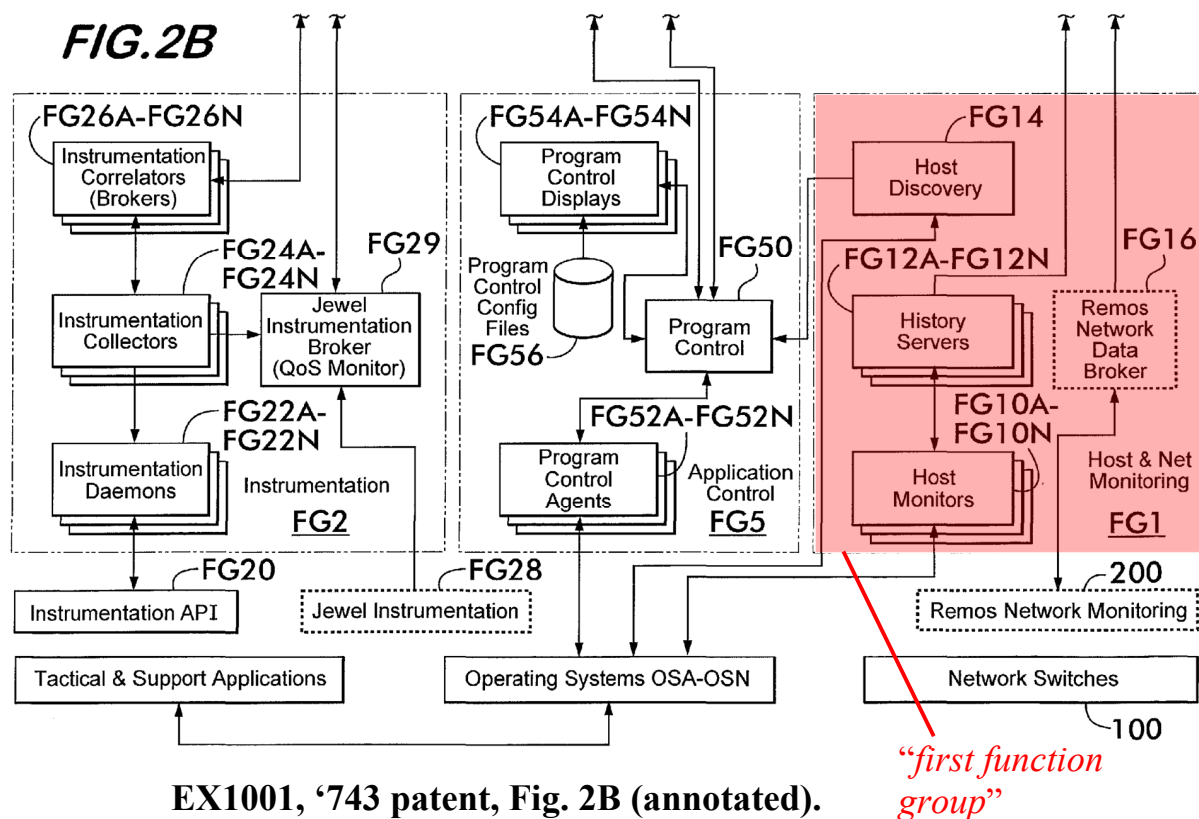


EX1001, ’743 patent, Figs. 1A, 1B (annotated).

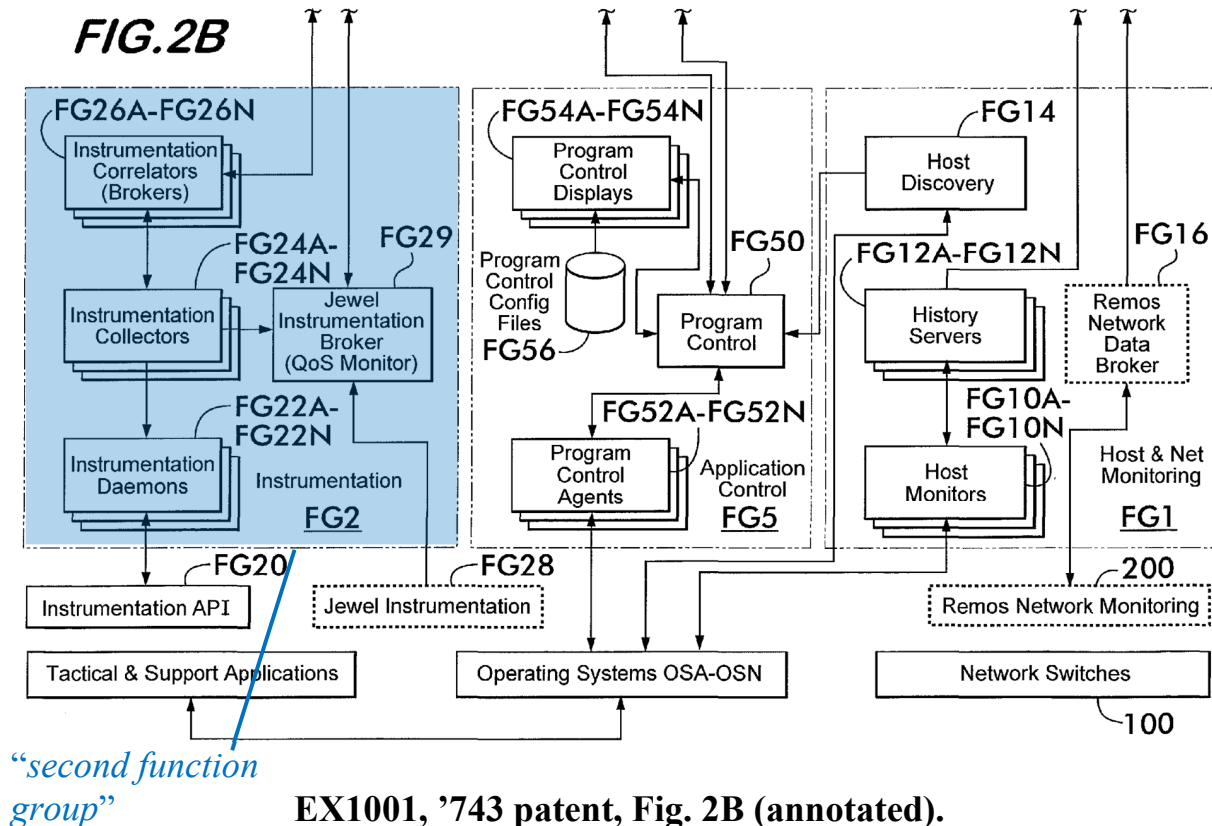
81. The distributed environment, shown above, is managed by “a resource management device generating signals which start up, shutdown or move a selected one of the copies responsive to [] information regarding performance of all copies of the scalable application and [] information regarding performance of the hosts.” EX1001, ’743 patent, 2:47-52.

82. The ’743 patent describes management of the distributed computing

system in terms of “functions” and “function groups” (described in FG# format). EX1001, '743 patent, 7:15-8:61. For example, the '743 patent describes a function group for monitoring the hosts and the network: “FG1—Host and Network Monitoring. This function group consists of software which monitors the host and network resources within the distributed environment.” EX1001, '743 patent, 7:29-31. FG1 “collects extensive run-time information on host and network configuration, statuses, and performance.” EX1001, '743 patent, 7:31-32. FG1 is recited in claim 10, for example, as “a first function group which monitors the host and network resources.” EX1001, '743 patent, claim 10. The first function group is illustrated in annotated Fig. 2B shown below.



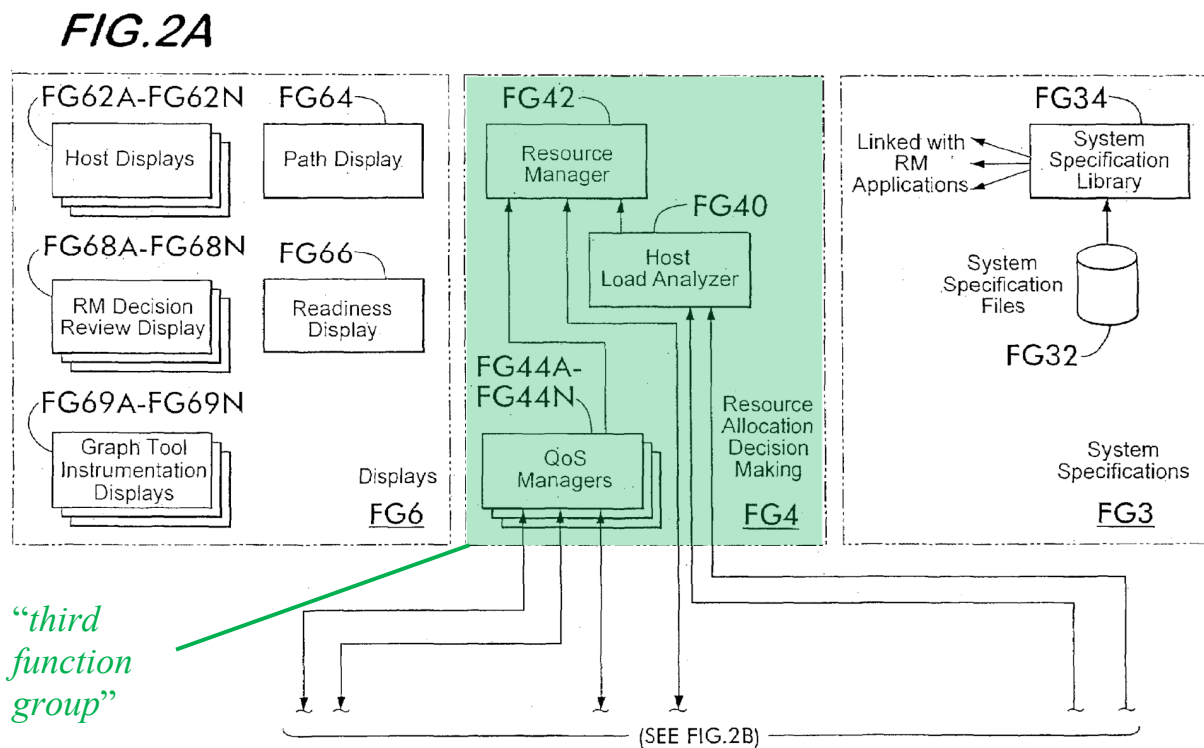
83. The '743 patent also describes a second function group for monitoring the applications that are running on the hosts: “FG2—Application-Level Instrumentation. The instrumentation function group provides general-purpose application event reporting and event correlation capabilities.” EX1001, '743 patent, 7:40-42. “This application data is forwarded to other components of the instrumentation subsystem which collect data from applications on hosts throughout the distributed environment.” EX1001, '743 patent, 7:47-49. The second function group is recited in claim 10, for example, as “*a second function group which provides application computer program event reporting and event correlation capabilities.*” EX1001, '743 patent, claim 10. The second function group is illustrated in annotated Fig. 2B shown below.



84. The '743 patent also describes a function group for making decisions and allocating resources: “FG4—Resource Allocation Decision-Making. This subsystem provides the reasoning and decision-making capabilities of the Resource Management architecture.” EX1001, '743 patent, 8:5-7. “The components of this subsystem use information from other subsystems in order to determine the health and state of the distributed environment and the options that are available for attempting to recover from faults or unacceptable performance.” EX1001, '743 patent, 8:7-9.

85. For example, the third function group may decide “1) where new applications should be started; 2) whether and where failed applications should be

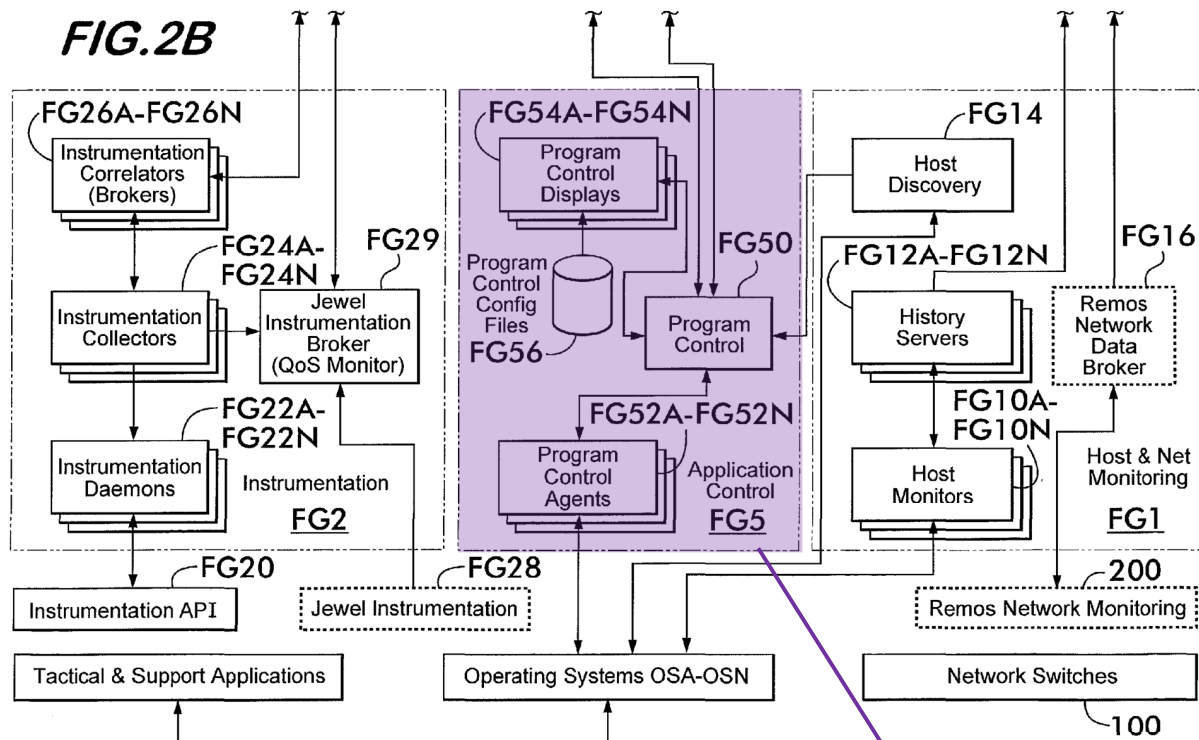
restarted; 3) based on application inter-dependencies, whether and where additional applications should [] be started prior to starting a particular application,” etc. EX1001, '743 patent, 8:14-23. This function group is recited in claim 10, for example, as “a third function group which provides reasoning and decision making capabilities for the resource managed system.” EX1001, '743 patent, claim 10. The third function group is illustrated in annotated Fig. 2A, shown below.



EX1001, '743 patent, Fig. 3 (annotated).

86. Finally, the '743 patent describes a fourth function group for enacting the decisions made by the third function group: “FG5—Application (Resource) Control. This subsystem provides application control (i.e., Program Control) capabilities which permit starting, stopping, and configuring applications on each of

the hosts in the distributed environment.” EX1001, ’743 patent, 8:31-35. This function group is recited in claim 10, for example, as “a fourth function group which provides program control capabilities permitting starting, stopping, and configuring of selected ones of the M managed characteristic application computer programs on respective ones of the N hosts in the resource managed system.” EX1001, ’743 patent, claim 10. The fourth function group is illustrated in Fig. 2B below.



EX1001, ’743 patent, Fig. 2B (annotated).

“fourth function group”

87. Accordingly, the claims of the ’743 patent merely recite the basic concepts of managing a distributed computing system. See e.g., EX1001, ’743 patent, claim 10. In particular, the claims recite the basic concepts of 1) monitoring the hosts and network, 2) monitoring the applications, 3) making decisions based on

the monitoring, and 4) enacting those decisions. *See* EX1001, '743 patent, claim 10. *See also* §V.B above.

88. However, as I explain below, the functionality for managing a distributed system was not new as of the time of filing. Indeed, some of the inventors of the '743 patent, along with others, had already disclosed these concepts in various papers before the '743 patent was filed.

B. Prosecution History

89. The '743 patent was filed as U.S. Application No. 09/864,825 on May 24, 2001. It claims priority to U.S. provisional application No. 60/207,891 ("891 provisional") (EX1009), filed on May 25, 2000. *See* EX1001, '743 patent, title page.

90. In a first Office Action, the claims were rejected as being indefinite under 35 U.S.C. § 112 and as being unpatentable as obvious over U.S. Patent Publication No. 2002/0064126 to Bhaatal and U.S. Patent No. 6,041,306 to Du. EX1002 ('743 File History), 208-222. The Applicant then made significant amendments to the claims, particularly to original claims 16 and 26 (issued claims 1 and 10), to overcome the rejections. EX1002 ('743 File History), 182-192.

91. In the second Office Action, the Examiner maintained the obviousness rejection with the addition of U.S. Patent No. 6,742,020 to Dimitroff. EX1002 ('743 File History), 164. The Applicant made additional amendments to the claims in response to the second Office Action. EX1002 ('743 File History), 123-33.

However, these amendments were ultimately ignored due to a typographical error that prompted a notice of non-compliant amendment. EX1002 ('743 File History), 121.

92. The Office then allowed some claims, including what is now issued claim 10, in a later Office Action. The Office did not give any specific reasons for allowance. EX1002 ('743 File History), 14-17. The most recent amendment to what is now issued claim 10 related to “the QoS requirements dictating parameters regarding service quality of the M management characteristic application programs” and “that the managed characteristic application computer programs are moved, shutdown, and started in accordance with satisfaction of the QoS requirements.” EX1002 ('743 File History), 150-51.

93. However, for the reasons explained in this declaration, these concepts were already known, as evidenced by the papers published by some of the inventors and other non-inventor co-authors.

C. Priority Date of the '743 Patent

94. The Challenged Claims of the '743 patent are not entitled to their provisional filing date because the Challenged Claims recite subject matter that was not described in the '891 provisional (EX1009).

95. It has been explained to me that for a claim to be entitled to the date of a provisional application, the earlier application must provide written description

support for the claimed subject matter. To satisfy the written description requirement of pre-AIA 35 U.S.C. §112 ¶ 1, a patent specification must describe the claimed invention in sufficient detail that one skilled in the art can reasonably conclude that the inventor **had possession** of the claimed invention. An applicant shows possession of the claimed invention by describing the claimed invention with **all of its limitations**. Moreover, it is not sufficient that the specification merely lists each claimed element. Rather, the specification must support the particular claimed combination of elements.

96. Here, the '891 provisional does not provide written description support for the specific combination of elements recited in the claims. Claim 10, for example, organizes various functions of the distributed computing management system into specific “functions” and “function groups.” EX1001, '743 patent, claim 10. However, the '891 provisional does not **describe** these specific function groups. Indeed, the term “function group” and the concept of function groups are entirely absent from the provisional application. *See* EX1009 ('891 provisional).

97. For example, the '891 provisional application does not describe “*a third function group which provides reasoning and decision making capabilities for the resource managed system.*” *See* EX1009 ('891 provisional). Moreover, while claim 10 recites that the “*third function group*” includes a “*first function,*” a “*second function,*” and a “*third function,*” the '891 provisional does not describe these three

specific functions as being within a “third function group.” EX1001 (’743 patent), claim 10. In other words, no such arrangement of functions exists within the provisional application. *See* EX1009 (’891 provisional).

98. Accordingly, because the ’891 provisional does not describe the combination of elements as arranged in at least claim 10, the ’891 provisional does not provide written description support for the Challenged Claims. As such, the Challenged Claims of the ’743 patent are entitled to a priority date no earlier than that of the May 24, 2001 filing date of the ’743 patent.

VII. CLAIM CONSTRUCTION

99. It is my understanding that in order to properly evaluate the ’743 patent, the terms of the claims must first be interpreted. It is my understanding that for the purposes of this *inter partes* review, the claims are to be construed under the so-called *Phillips* standard, under which claim terms are given their ordinary and customary meaning as would be understood by one of ordinary skill in the art in light of the specification and prosecution history, unless the inventor has set forth a special meaning for a term. For the purposes of my analysis below, I do not believe any claim terms require explicit construction at this time.

VIII. IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE

100. I have been asked to provide my opinion as to whether the Challenged Claims of the ’743 patent would have been obvious in view of the prior art. The

discussion below provides a detailed analysis of how the prior art references identified below teach the limitations of the Challenged Claims of the '743 patent.

101. As part of my analysis, I have considered the scope and content of the prior art and any differences between the alleged invention and the prior art. I describe in detail below the scope and content of the prior art, as well as any differences between the alleged invention and the prior art, on an element-by-element basis for each Challenged Claims of the '743 patent.

102. As described in detail below, the alleged invention of the Challenged Claims would have been obvious in view of the teachings of the identified prior art references as well as the knowledge of a POSITA. The obviousness grounds rely on the combined teachings of the references and not on a physical incorporation of elements. The following table summarizes my grounds of invalidity:

Ground	References and Basis	Claims
1	Welch alone and in combination with Ravindran (obviousness)	10, 11, 14, 16, and 18
2	Welch, Ravindran, and Hosokawa (obviousness)	10, 11, 14, 16, and 18

A. Ground 1: Claims 10, 11, 14, 16, and 18 are obvious under 35 U.S.C. § 103 over Welch alone and in combination with Ravindran.

1. Summary of Welch

103. Welch is a whitepaper that was authored by some of the inventors of the '743 patent, along with other co-authors who are not inventors of the '743 patent. Compare EX1001, '743 patent, field (75) with EX1005, Welch, 1. The following comparison table lists the authors of the '743 patent and Welch and highlights common authors:

'743 patent inventors	Welch authors
Paul V. Werme	Paul V. Werme
Michael W. Masters,	Michael W. Masters,
Lonnie R. Welch	Lonnie R. Welch
Larry A. Fontenot	Binoy Ravindran
	Behrooz A. Shirazi
	Prashant A. Shirolkar
	Robert D. Harrison
	Wayne Mills
	Tuy Do
	Judy Lafratta
	Shafqat M. Anwar
	Steve Sharp
	Terry Sergeant
	George Bilowus
	Mark Swick
	Jim Hoppel
	Joe Caruso

104. Welch describes the same distributed system management software

that is described and claimed in the '743 patent. EX1005, Welch, 1-5. In particular, Welch describes “distributed real-time systems” which are “scalable” and designed to “meet the QoS requirements of each application they manage.” EX1005, Welch, 1.

105. Welch is directed to QoS management software that manages resources for a distributed real-time system. EX1005, Welch, 1. Welch describes “distributed hardware” (*e.g.*, a plurality of hosts) that run several instances of “Real-Time Control System (RTCS) application programs on the specified hosts.” EX1005, Welch, 2-3. These RTCS applications are scalable according to demand. A resource manager is used to “move a program to a different host or LAN, shed a program, or replicate a program) to improve QoS.” EX1005, Welch, 3. Welch illustrates both the “logical architecture” and the “physical architecture” of the QoS management software in Figs. 1 and 2, shown below.

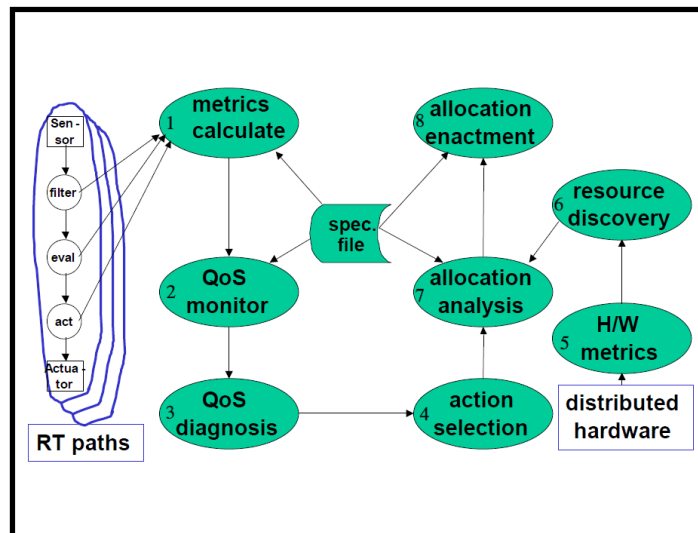


Figure 1. Logical architecture of the resource and QoS management software.

EX1005, Welch, Fig. 1.

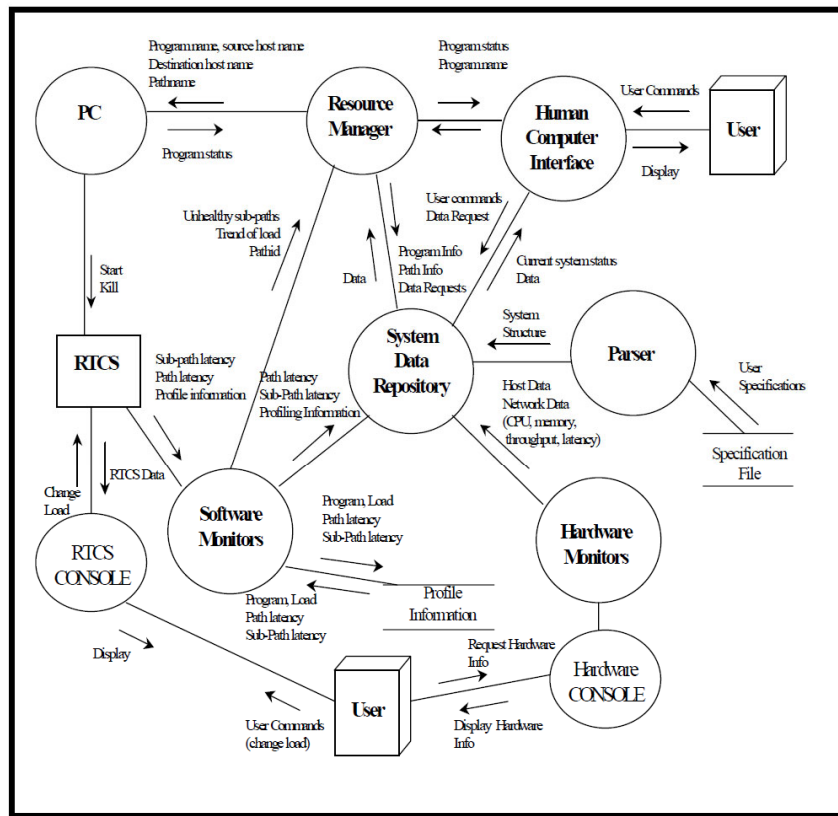
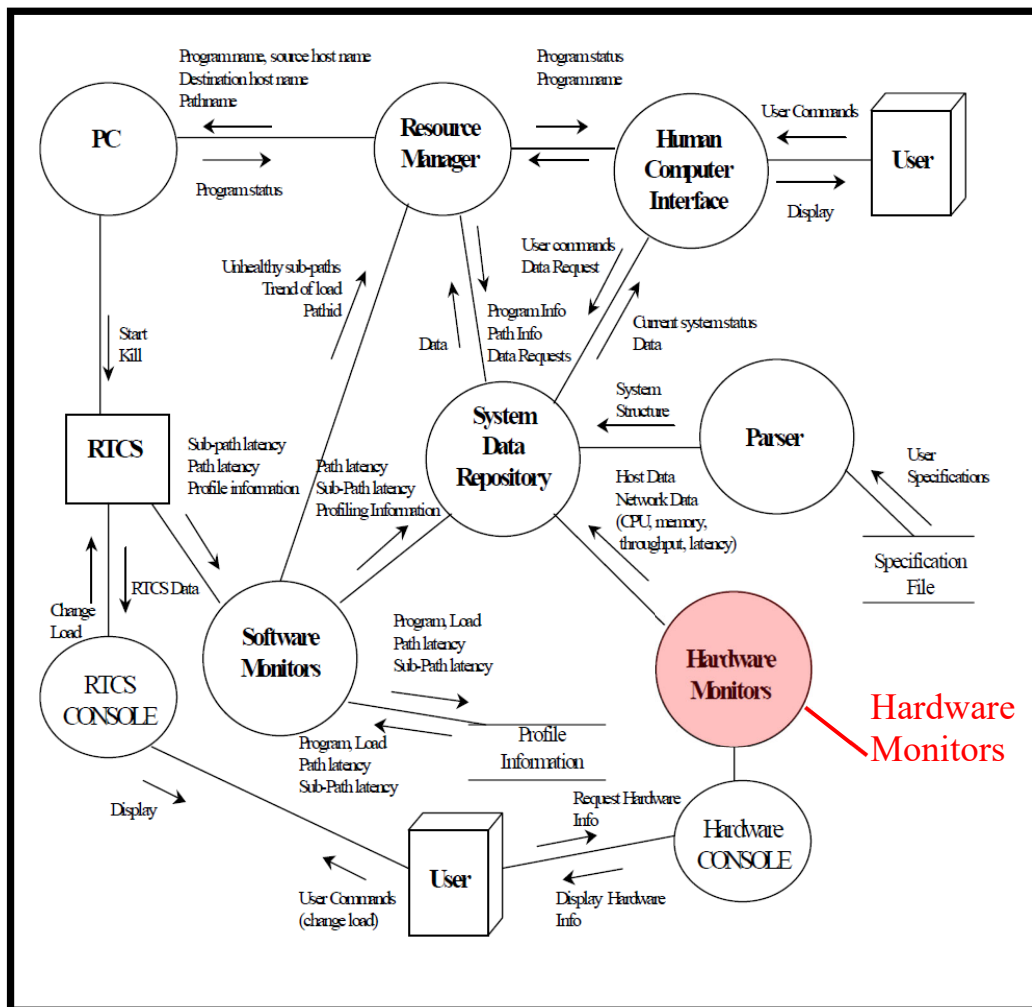


Figure 2. Physical architecture of the resource and QoS management software.

EX1005, Welch, Fig. 2.

106. Welch discloses the same functionality corresponding to the “function groups” that are recited in the Challenged Claims of the ’743 patent. EX1005, Welch, 2-3. For example, Welch describes a first function group in the form of hardware monitors that monitor host and network resources. “**Hardware (HW) Monitors** continuously observe a resource’s load index and pass this information to the Hardware Data Repository (step 2 of Figure 3), which in turn passes such information to the System Data Repository (step 3 of Figure 3).” EX1005, Welch, 3. As illustrated in annotated Fig. 2 shown below, the hardware monitors collect

host data related to “CPU” and “memory” and network data related to “throughout” and “latency” (“*host and network resources*”). EX1005, Welch, Fig. 2.

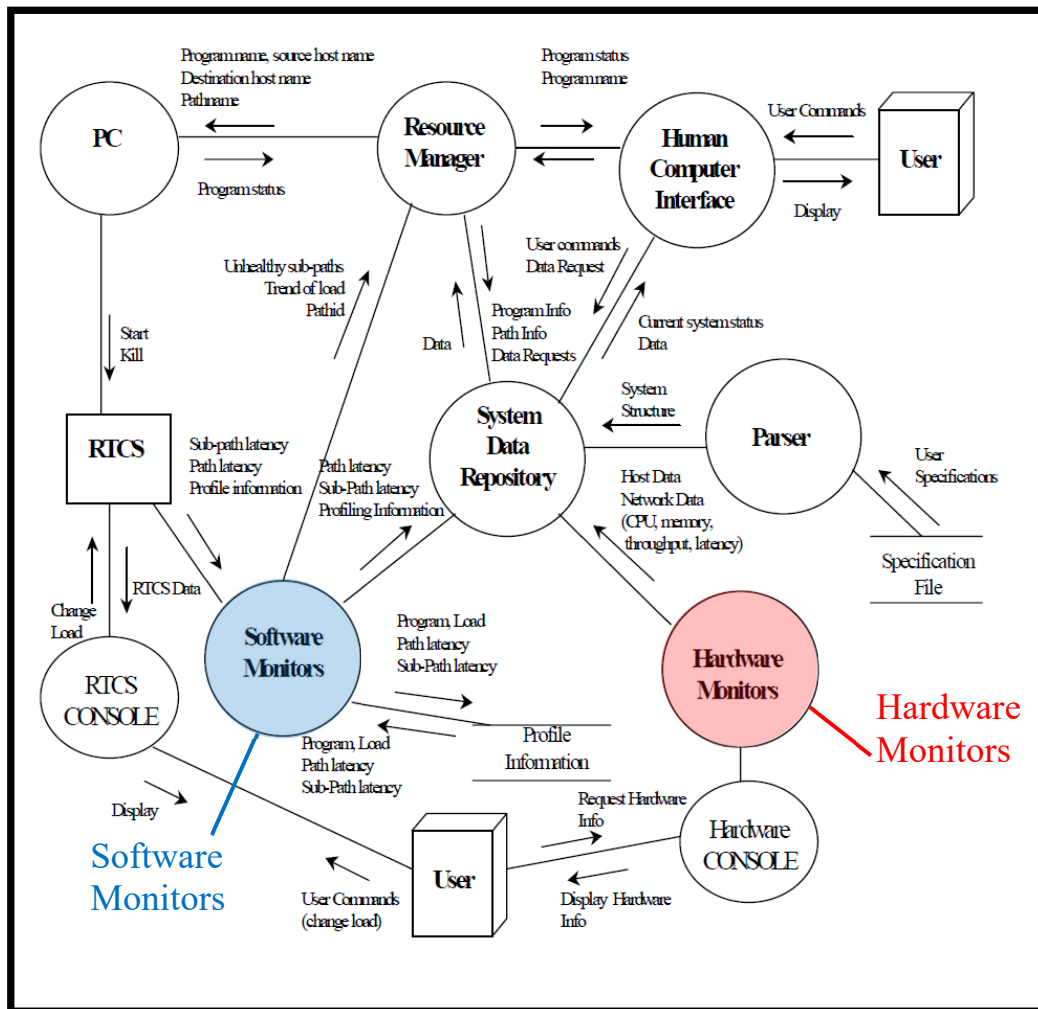


EX1005, Welch, Fig. 2 (annotated).

107. Welch also discloses a second function group in the form of software monitors. EX1005, Welch, 3. “The reallocations made by the resource manager make use of information provided by the hardware and software monitors.” EX1005, Welch, 3. As shown in annotated Fig. 2 below, the RTCS applications provide information to the software monitors, including “Sub-path latency,” “Path

latency,” and “Profile information.” The software monitors then report information to the Resource Manager as “Unhealthy sub-paths,” “Trend of load”, and “Pathid.”

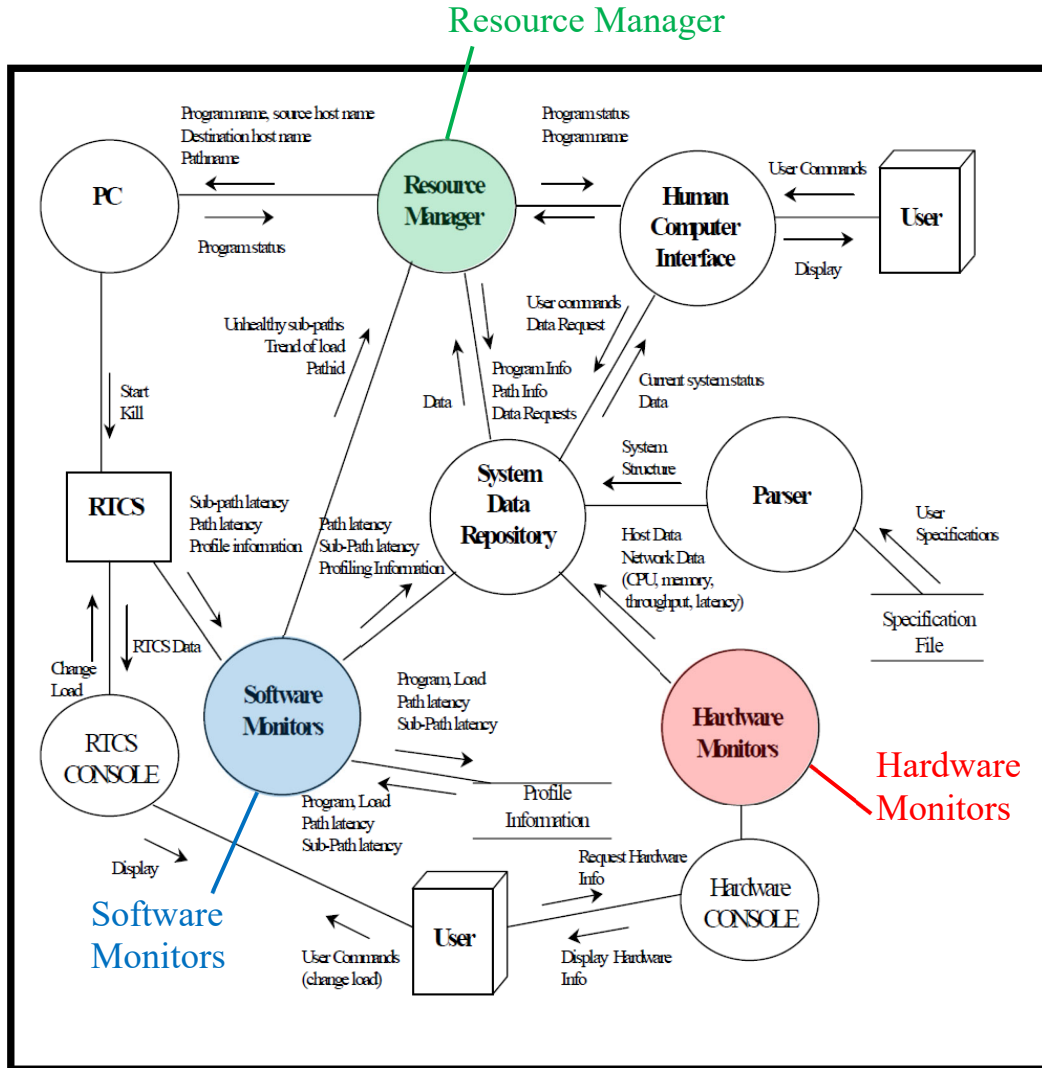
EX1005, Welch, Fig. 2.



EX1005, Welch, Fig. 2 (annotated).

108. Welch further discloses a third function group in the form of a resource manager. EX1005, Welch, 3. “The core component of the middleware is the resource manager. It is activated when programs die and when time-constrained control paths miss their deadlines. In response to these events, it takes appropriate

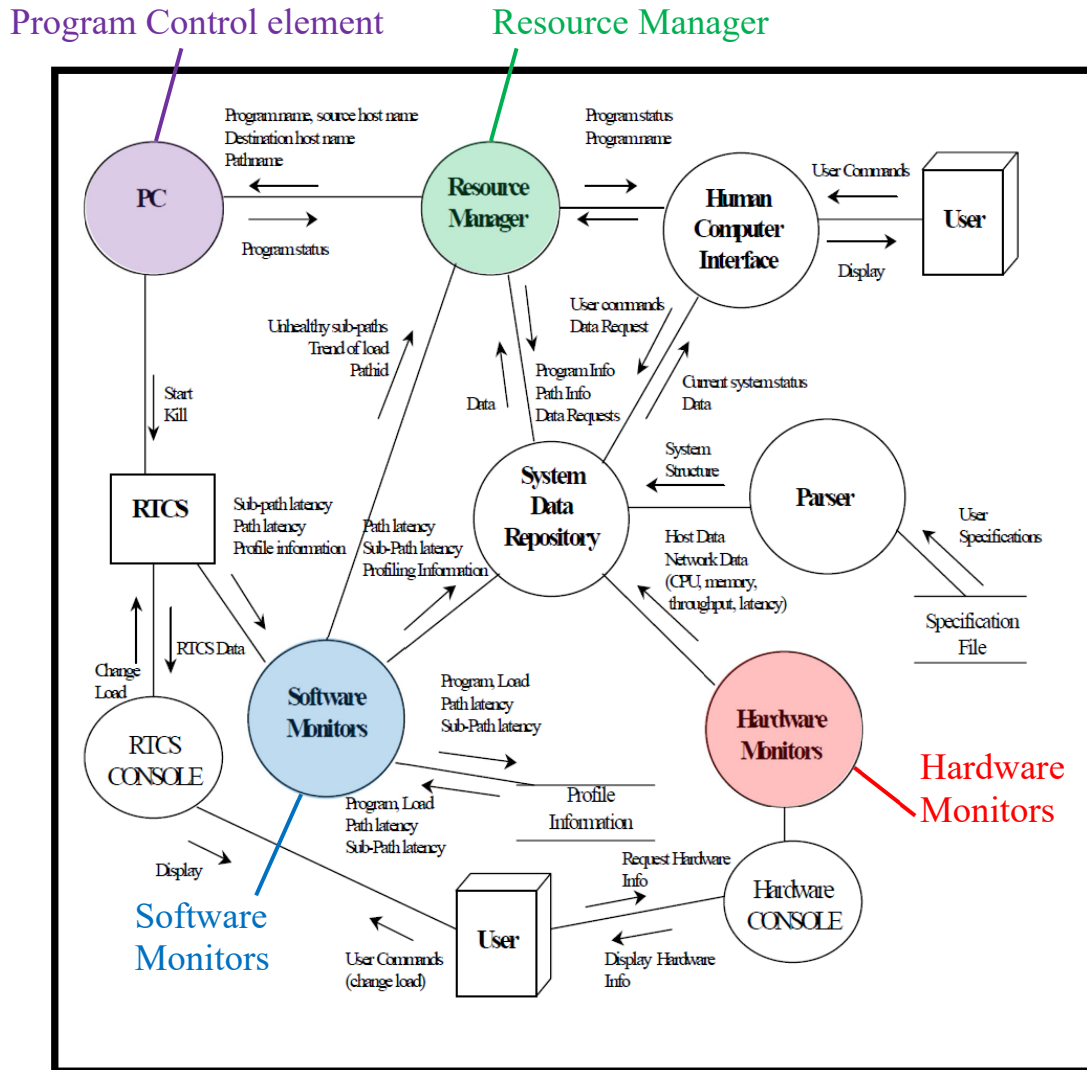
measures to improve the quality of service delivered to the applications.” EX1005, Welch, 3. The resource manager is shown in annotated Fig. 2 below.



EX1005, Welch, Fig. 2 (annotated).

109. Welch also discloses a fourth function group in the form of a program control element. EX1005, Welch, 2-3. “The program control (PC) component consists of a central control program and a set of startup daemons. When resource manager needs to start a program on a particular host, it informs the control program,

which notifies the startup daemon on that host.” EX1005, Welch, 3. The program control element is also illustrated in Fig. 2 below.



EX1005, Welch, Fig. 2 (annotated).

110. Accordingly, Welch’s authors—three of which are the same inventors of the ’743 patent—had previously disclosed the same system that is described in the Challenged Claims.

111. Welch is also analogous art to the ’743 patent because it is directed to

the same field of endeavor (distributed computing) as the '743 patent. *Compare* EX1001, '743 patent, abstract (“A resource manager for a distributed environment”) *with* EX1005, Welch, title (“Distributed, Scalable, Dependable Real-Time Systems: Middleware Services and Applications”).

112. In addition, Welch is reasonably pertinent to a problem addressed by the inventors of the '743 patent because Welch describes the same resource management techniques for improving efficiency and scalability that are purportedly solved by the '743 patent. *See* EX1005, Welch, 1. The '743 patent likewise purports to address efficient use of resources. *See* EX1001, '743 patent, 2:15-20 (seeking to “provide a system that is both flexible and adaptive (or at least easily adapted) to changes in system configuration, performance bottlenecks, survivability requirements, scalability, etc.”).

2. Summary of Ravindran

113. Welch references other whitepapers (*see* EX1005, Welch, 5) including Ravindran, EX1006, whose lead author is the same Welch author/inventor. Welch (EX1005) and Ravindran (EX1006) share three of the authors; Ravindran (EX1006) and the '743 patent (EX1001) share one author-inventor (Lonnie R. Welch). These whitepapers describe various aspects of Welch’s distributed system in further detail.

114. For example, Welch refers to a “path-based real-time subsystem (*see* [11]).” Ravindran (EX1006) is Paper 11 in Welch. Ravindran, for example, also

describes hardware and software monitors for monitoring the performance of applications and hosts: “Hardware (HW) Monitors (which run on each host) periodically calculate load indices for hosts and LANS, and pass this information to the Hardware Data Repository (HDR).” EX1006, Ravindran, 7.

3. Motivation to Combine Welch and Ravindran

115. A POSITA would have found it obvious for Welch’s hardware monitors to be implemented as described in Ravindran. Indeed, Ravindran shares many of the same authors as Welch and Welch explicitly cites to Ravindran. Welch states that “[o]ur approach is based on the dynamic path paradigm. A path-based real-time subsystem (see [Ravindran]) typically consists of a detection & assessment path, an action initiation path and an action guidance path.” EX1005, Welch, 1. Accordingly, implementing Welch’s hardware monitors as they are described in Ravindran provides the predictable benefit of implementing the hardware monitors as the Welch authors intended.

116. The combination of Welch and Ravindran is merely the combination of prior art elements (Welch’s and Ravindran’s hardware monitors) according to known methods (for collecting host and LAN metrics) to yield predictable results (providing the resource manager with real-time statuses). Welch explicitly suggests the combination by identifying Ravindran as related to its disclosed system. A POSITA would have had a reasonable expectation of success in combining Welch

and Ravindran because both relate to the same type of path-based system, and Welch expressly references the conventional implementation details in Ravindran. *See* EX1005, Welch 1-4, 5 (Ravindran is document [11] referenced in Welch); EX1006, Ravindran, 1-7. Indeed, it is fairly common for (nearly or) the same authors to publish a series of papers/articles, around the same time frame, detailing different aspects of the same system (as I have many times, see my CV, EX1004).

4. Claim 10.

(a) [10.0] Software stored on at least one host for converting N networked hosts into a resource managed system instantiating M managed characteristic application computer programs, each managed characteristic application computer program managed by one of the N networked hosts, the software comprising:

117. To the extent the preamble is limiting, Welch discloses or at least renders obvious [10.0].

118. First, Welch describes QoS management software (“*Software*”) that manages the resources of a distributed system (“*software ... for converting N networked hosts into a resource managed system*”). Welch “presents middleware services that support such dynamic real-time systems through adaptive resource management.” EX1005, Welch, 1. For example:

Some classes of real-time systems function in environments which cannot be modeled with static approaches. In such environments, the arrival rates of events which drive transient computations may be

unknown. Also, the periodic computations may be required to process varying numbers of data elements per period, but the number of data elements to be processed in an arbitrary period cannot be known at the time of system engineering, nor can an upper bound be determined for the number of data items; thus, a worst case execution time cannot be obtained for such periodics. **This paper presents middleware services that support such dynamic real-time systems through adaptive resource management.** The middleware services have been implemented and employed for components of the experimental Navy system described in [10]. Experimental characterizations show that the services provide timely responses, that they have a low degree of intrusiveness on hardware resources, and that they are scalable.

EX1005, Welch, abstract.

119. A POSITA would have understood that middleware is software. *See* §V.B above.

120. Welch’s middleware is embodied as “QoS management software.”

EX1005, Welch, 2-3. For example:

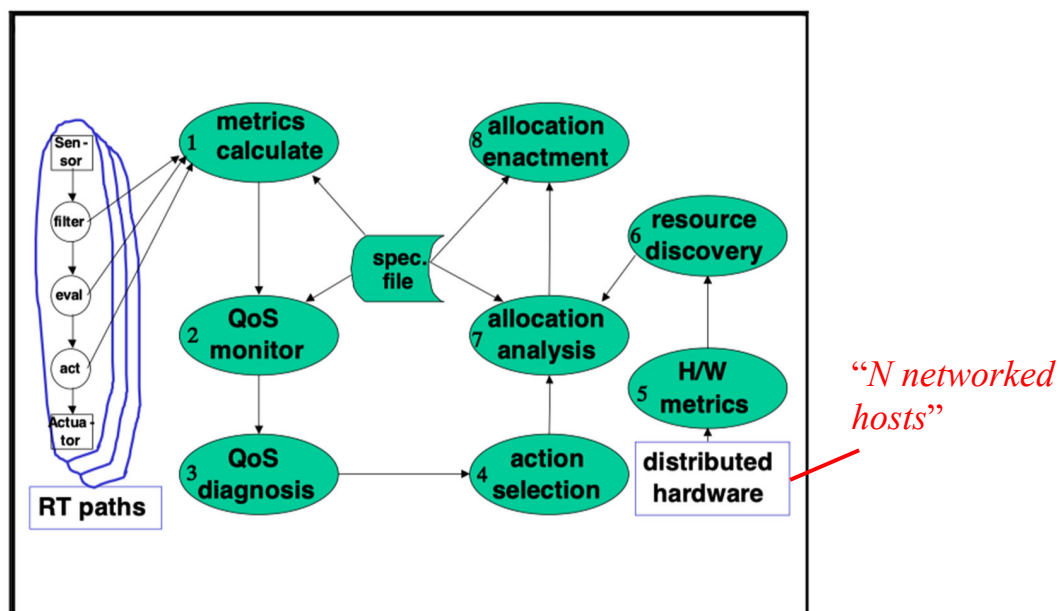
The logical architecture of the **QoS management software** is shown in Figure 1. It behaves as follows. The application programs of real-time control paths send timestamped events to the metrics calculation component, which calculates path-level QoS metrics and sends them to the QoS monitor. The monitor checks for conformance of observed QoS to required QoS, and notifies the QoS diagnosis component when a QoS violation occurs. **QoS diagnosis notifies the action selection component of the cause(s) of poor QoS and recommends actions**

(e.g., move a program to a different host or LAN, shed a program, or replicate a program) to improve QoS. Action selection ranks the recommended actions, identifies redundant actions, and forwards the results to the allocation analysis component; this component consults resource discovery for host and LAN load index metrics, determines a good way to allocate the hardware resources in order to perform the actions, and requests the actions be performed by the allocation enactment component.

EX1005, Welch, 2.

121. A POSITA would have understood that such a QoS management software is part of a resource managed systems because Quality of Service (QoS) systems and their characteristics (*e.g.*, metrics) are a form of managing resources. *See* §V.B.1 above.

122. Welch's Fig. 1, below, illustrates the "logical architecture" of the QoS management software ("*Software*"). EX1005, Welch, 2.



EX1005, Welch, Fig. 1 (annotated).

123. As illustrated in Welch’s Fig. 1 above, one logical function of the QoS management software is to receive hardware metrics from “distributed hardware.” EX1005, Welch, Fig. 1. The distributed hardware represents “*N networked hosts*” because the QoS management software “recommend[s] actions (e.g., move[s] a program to a **different host or LAN** ...)” which indicates that the “distributed hardware” includes multiple hosts interconnected by at least one LAN (Large Area Network). EX1005, Welch, 2.

124. Welch further explains that “the allocation analysis component ... consults resource discovery **for host and LAN** load index metrics.” EX1005, Welch, 2. For example:

The Resource Manager needs to **decide on which host(s) and LAN(s)** the unhealthy sub-path needs to be replicated or moved. This decision is made by **choosing the host(s) and LAN(s)** with the smallest load indices. The Hardware Analyzer ranks the hosts and LANs in ascending order of their load index and passes this information to the Resource Manager (step 9 of Figure 3).

EX1005, Welch, 3.

125. Accordingly, Welch's QoS management software manages "*N networked hosts*" because the software manages applications across multiple hosts and LANs of distributed hardware.

126. The QoS software's resource management of the distributed hardware ("*N networked hosts*") makes that distributed hardware into a resource managed system ("*software ... for converting N networked hosts into a resource managed system*").

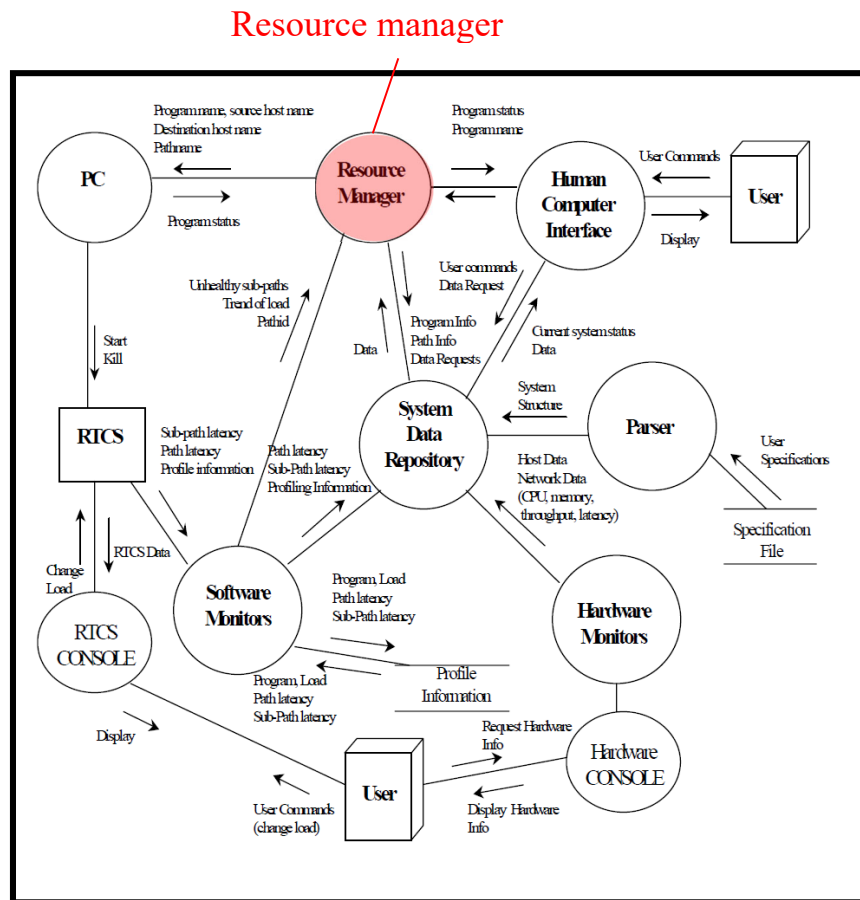
127. Indeed, Welch explains that "the core component of the middleware [QoS management software] is **the resource manager.**" EX1005, Welch, 2. For example:

The physical QoS management architecture is shown in Figure 2. **The core component of the middleware is the resource manager.** It is activated when programs die and when time-constrained control paths miss their deadlines. In response to these events, **it takes appropriate measures to improve the quality of service delivered to the applications.** The reallocations made by the resource manager make

use of information provided by the hardware and software monitors, as well as from a specification file that describes QoS requirements and the structures of the software system and the hardware system. The system data repository component is responsible for collecting and maintaining all system information. The program control (PC) component consists of a central control program and a set of startup daemons. When resource manager needs to start a program on a particular host, it informs the control program, which notifies the startup daemon on that host. The HCI provides information to the user regarding the system configuration, application status, and reallocation decisions. It also allows the operator to dynamically modify the behavioral characteristics of the resource discovery and the resource manager components.

EX1005, Welch, 2-3.

128. Welch illustrates the “resource manager” in Fig. 2, annotated below, which provides additional detail as to the “physical architecture” of the QoS management software. EX1005, Welch, Fig. 2.



EX1005, Welch, Fig. 2 (annotated).

129. Accordingly, the QoS management software (“*Software*”) includes a resource manager that “takes appropriate measures to improve the quality of service delivered to the applications.” EX1005, Welch, 2. Welch explains that “[t]he reallocations made by the resource manager make use of information provided by the hardware and software monitors, as well as from a specification file that describes QoS requirements and the structures of the software system and the hardware system.” EX1005, Welch, 2. A POSITA would have understood that the resource manager of the QoS management software thus makes the distributed

system a resource managed system (“*software ... for converting N networked hosts into a resource managed system*”).

130. Second, the QoS management software (“*Software*”) runs on multiple heterogeneous host platforms and is thus “*stored on at least one host.*” For example:

The software runs on a heterogeneous network configuration that includes Myrinet, ATM, FDDI, and ethernet, **on multiple heterogeneous host platforms**, including Dec Alpha’s with OSF-1, a Dec Sable with OSF-1, TAC-4’s with HP-UX, Sun Sparc 10’s with Solaris, and Pentiums with OSF-1RT.

EX1005, Welch, 4.

131. Accordingly, in Welch, the host platform(s) storing and running the QoS management software is “*at least one host*” as claimed.

132. Third, the resource manager of the QoS management software starts up real-time control programs (“*instantiating M managed characteristic application computer programs*”). EX1005, Welch, 2-3. “When resource manager needs to start a program on a particular host, it informs the control program, which notifies the startup daemon on that host.” EX1005, Welch, 3. “The Resource Manager informs the Program Control to start the Real-Time Control System (RTCS) application programs on the specified hosts.” EX1005, Welch, 3.

133. The RTCS application programs are “*instantiate[ed]*” because they are each instances of a scalable or replicable program. For example:

Response time measurements were also made for path overload detection and overload recovery via automatic scalability. The total Scalability response time (T_c) calculations are divided into 4 phases: (1) Path Overload data transfer time (t_2) is the time taken for the overloaded path information to reach the resource manager. This interval occurs immediately after the overload detection heuristic in the subsystem manager detects the overloaded condition of the path (t_1), (2) Resource Manager processing (t_3) is the same as in T_u , (3) **Scale time (T_s) is the time taken by the Program Manager and Startup Daemons to actually start a new copy of the program.** The scale time (T_s) consists of three phases, identical to the restart component of T_u . In repeated experiments, the average total response time, T_c , for scalability services was 4.51 seconds, with a standard deviation of 0.020731.

EX1005, Welch, 4.

134. Welch explains that these programs can be replicated. For example:

If a path becomes unhealthy (misses its deadline), the QoS Monitors detect such a condition, diagnose the cause of poor health, and suggest an action (such as moving or **replicating an application program**) to the Resource Manager (step 8 of Figure 3).

EX1005, Welch, 3.

135. Furthermore, the RTCS application programs are “*managed characteristic application computer programs*” because the resource manager manages at least their characteristics of scalability, survivability, fault tolerance or

priority. EX1005, Welch, 2-4. This is consistent with how the '743 patent describes a “managed characteristic.” EX1001, '743 patent, 54:20-21 (“managed characteristic may be one of scalability, survivability, fault tolerance or priority”).

136. Welch further explains that “[w]e performed experiments to determine the responsiveness of our QoS and resource management middleware for survivability and scalability services.” EX1005, Welch, 4. Indeed, the goal of Welch’s system is that “resources should be utilized in an efficient manner, and scalability must be provided to address the ever-increasing complexity of scenarios that confront such systems.” EX1005, Welch, 1. “Experiments show that the services provide bounded response times, scalable services, and low intrusiveness.” EX1005, Welch, 4. Accordingly, Welch’s real-time control system (RTCS) application programs are “*each managed characteristic application computer programs*” as claimed.

137. Fourth, the RTCS programs run on distributed hardware (“each managed characteristic application computer program managed by one of the N networked hosts”). EX1005, Welch, 3. For example:

The Resource Manager informs the Program Control to start the Real-Time Control System (RTCS) application programs on the specified hosts (step 5 of Figure 3).

EX1005, Welch, 3.

138. The resource manager manages applications by starting them on one of

the networked hosts: “When resource manager needs to start **a program on a particular host**, it informs the control program, which notifies the startup daemon **on that host**.” EX1005, Welch, 3.

139. Thus, Welch describes QoS management software for managing real-time control system applications on distributed hardware to provide startup, replicating, and other actions related to providing for scalability and survivability, which discloses or renders obvious “*Software stored on at least one host for converting N networked hosts into a resource managed system instantiating M managed characteristic application computer programs, each managed characteristic application computer program managed by one of the N networked hosts*” as claimed.

140. **In sum**, Welch discloses “Software [*e.g.*, QoS management software] stored on at least one host [*e.g.*, one of Welch’s heterogeneous hosts] for converting N networked hosts [*e.g.*, distributed hardware] into a resource managed system [*e.g.*, distributed system] instantiating M managed characteristic application computer programs [*e.g.*, RTCS programs], each managed characteristic application computer program managed by one of the N networked hosts [*e.g.*, distributed hardware].”

(b) [10.1] a first function group which monitors the host and network resources;

141. Welch alone and in combination with Ravindran renders obvious limitation [10.1]. A POSITA would have been motivated to combine Welch and

Ravindran for the reasons discussed in §VIII.A.3 above.

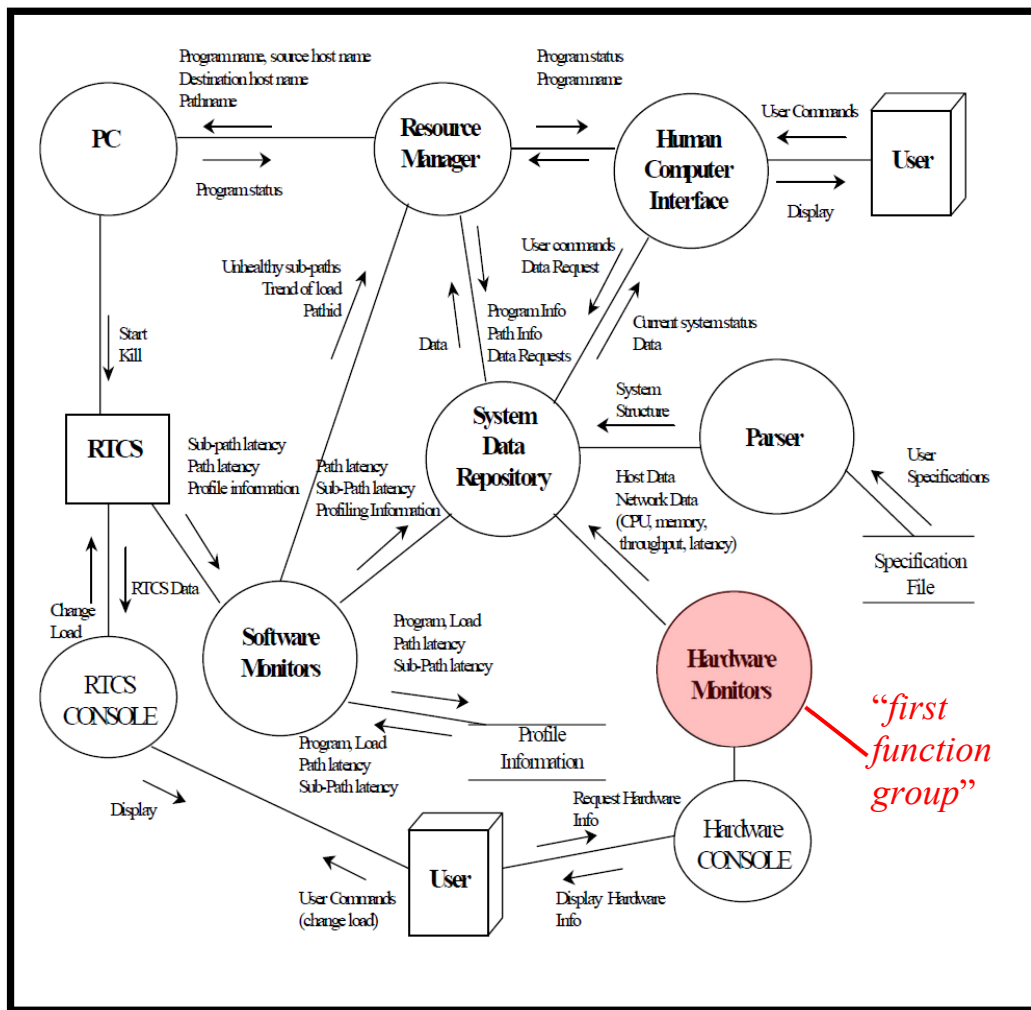
142. Welch’s QoS management software (“*software*”) includes “hardware monitors” (“*a first function group*”) that monitor host and network resources.

EX1005, Welch, 2-3. For example:

Hardware (HW) Monitors continuously observe a resource’s load index and pass this information to the Hardware Data Repository (step 2 of Figure 3), which in turn passes such information to the System Data Repository (step 3 of Figure 3).

EX1005, Welch, 3.

143. Welch illustrates the hardware monitors (“*first function group*”) in annotated Fig. 2 below.



EX1005, Welch, Fig. 2 (annotated).

144. The hardware monitors (“*first function group*”) are also illustrated in annotated Fig. 3 below.

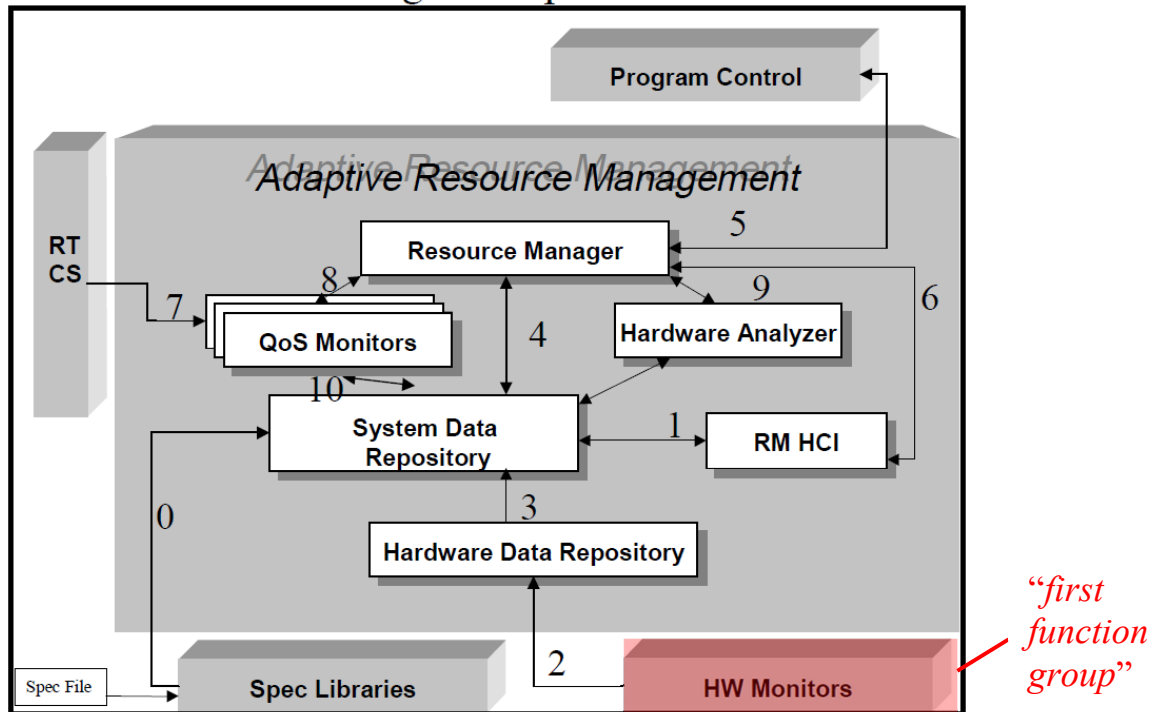
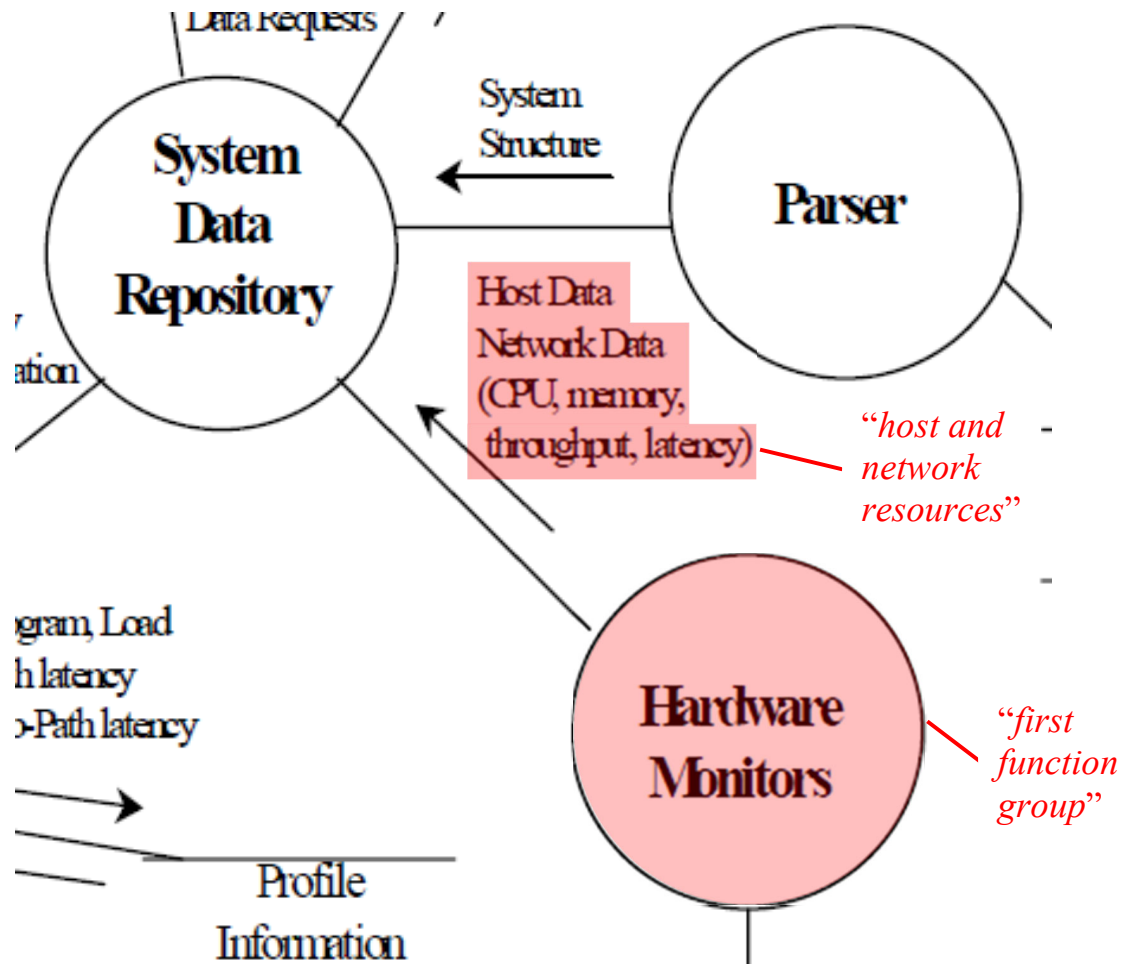


Figure 3. Adaptive resource management scenarios. EX1005, Welch, Fig. 3 (annotated).

145. As can be seen from a close-up view of Fig. 2 annotated below, the hardware monitors collect and send to the repository “Host Data” related to “CPU” and “memory” and “Network Data” related to “throughput” and “latency” (“*host and network resources*”). EX1005, Welch, Fig. 2.



EX1005, Welch, Fig. 2 (partial, annotated).

146. The host data and network data represent “*host and network resources*” because Welch refers to their collection as “network resource discovery” and “host resource discovery.” EX1005, Welch, 4. In particular, Welch describes the process of testing the QoS software system. EX1005, Welch, 4. In that description, Welch refers to obtaining resource discovery of network-level metrics and host level metrics from the host monitors. For example:

The processing time at the resource manager, t_3 , is further decomposed. Preprocessing time (t_{31}) is the time interval after receipt of a dead program message from the program manager and before network discovery begins. This time interval is internal to the resource manager. Network Resource Discovery (t_{32}) is the time interval required to **obtain network-level metrics** from the network controller. Host Resource Discovery (t_{33}) is the time interval required to obtain **host-level metrics** from all eligible host monitors. Allocation Decision time (t_{34}) is the time interval required to choose a good host. This interval is internal to the resource manager. Post-processing time (t_{35}) is the time interval after finding the best host and before sending a program restart instruction to the program manager. This interval is internal to the resource manager.

EX1005, Welch, 4.

147. A POSITA would have understood that Welch's hardware monitors are used to monitor the hosts and LANs. Some of the authors of Welch published other papers that describe the hardware monitors in more detail. For example, Ravindran (EX1006) describes hardware monitors as follows: "**Hardware (HW) Monitors** (which run on each host) periodically **calculate load indices for hosts and LANS**, and pass this information to the Hardware Data Repository (HDR)." EX1006, Ravindran, 7.

148. Accordingly, the system described by Welch and Ravindran includes hardware monitors, which include one sub-function of monitoring host resources,

and another sub-function of monitoring network resources. EX1005, Welch, 2-3. Together, these sub-functions form “*a first function group*” as claimed.

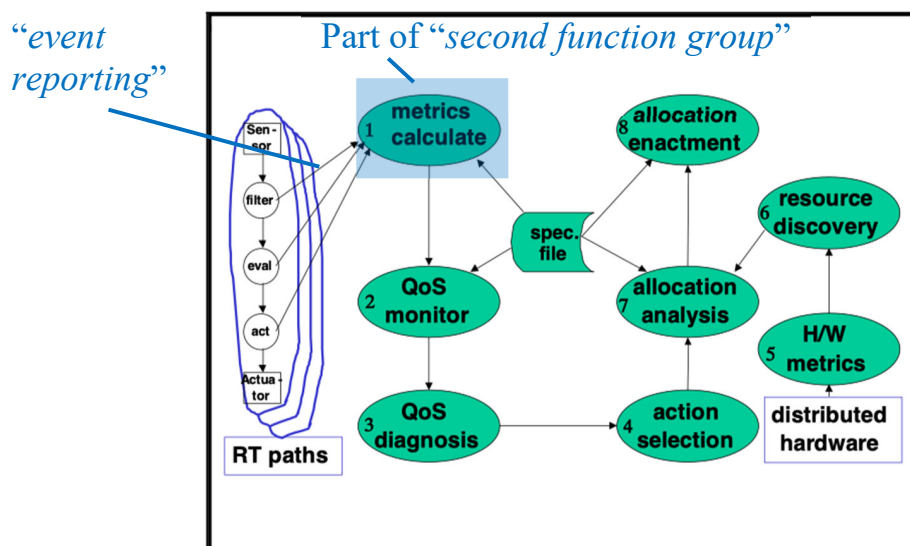
149. Thus, because Welch describes QoS software that includes hardware monitors for monitoring host and network resources, Welch’s QoS software system including hardware monitors, as taught by Ravindran, renders obvious “*a first function group which monitors the host and network resources.*”

150. In sum, Welch alone or in combination with Ravindran discloses “a first function group [*e.g.*, Welch’s hardware (HW) monitors, and/or Ravindran’s detailed monitors] which monitors the host [resources] [*e.g.*, Welch’s CPU, memory] and network resources [*e.g.*, Welch’s throughput, latency].”

(c) [10.2] a second function group which provides application computer program event reporting and event correlation capabilities;

151. Welch discloses or at least renders obvious limitation [10.2].

152. Welch’s QoS management software includes software monitors and a metrics calculation component that together teach a “*second function group*,” as shown in Fig. 1, annotated below. EX1005, Welch, 2-3. The part of the QoS management software corresponding to the “*second function group*” includes the functionality of providing “*application computer program event reporting and event correlation capabilities.*” EX1005, Welch, 2.



EX1005, Welch, Fig. 1 (annotated).

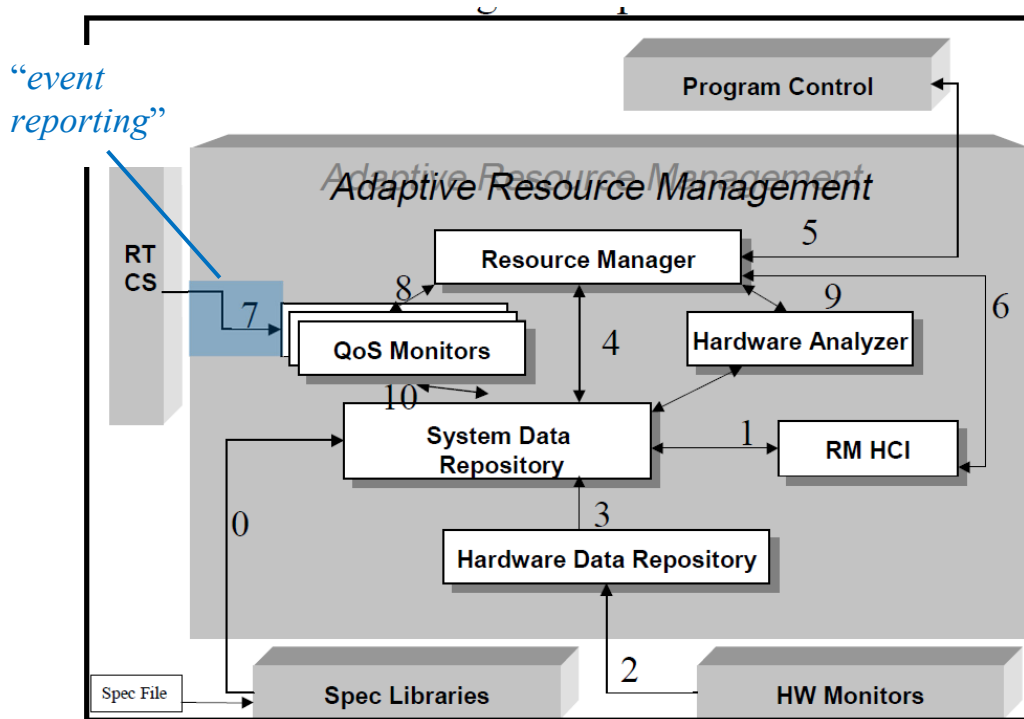
153. Welch's QoS management software provides "application computer program event reporting" because the "application programs of real-time control paths send timestamped events to the metrics calculation component, which calculates path-level QoS metrics and sends them to the QoS monitor." EX1005, Welch, 2. For example:

The logical architecture of the QoS management software is shown in Figure 1. It behaves as follows. The application programs of real-time control paths send timestamped events to the metrics calculation component, which calculates path-level QoS metrics and sends them to the QoS monitor. The monitor checks for conformance of observed QoS to required QoS, and notifies the QoS diagnosis component when a QoS violation occurs. QoS diagnosis notifies the action selection component of the cause(s) of poor QoS and recommends actions (e.g., move a program to a different host or LAN,

shed a program, or replicate a program) to improve QoS. Action selection ranks the recommended actions, identifies redundant actions, and forwards the results to the allocation analysis component; this component consults resource discovery for host and LAN load index metrics, determines a good way to allocate the hardware resources in order to perform the actions, and requests the actions be performed by the allocation enactment component.

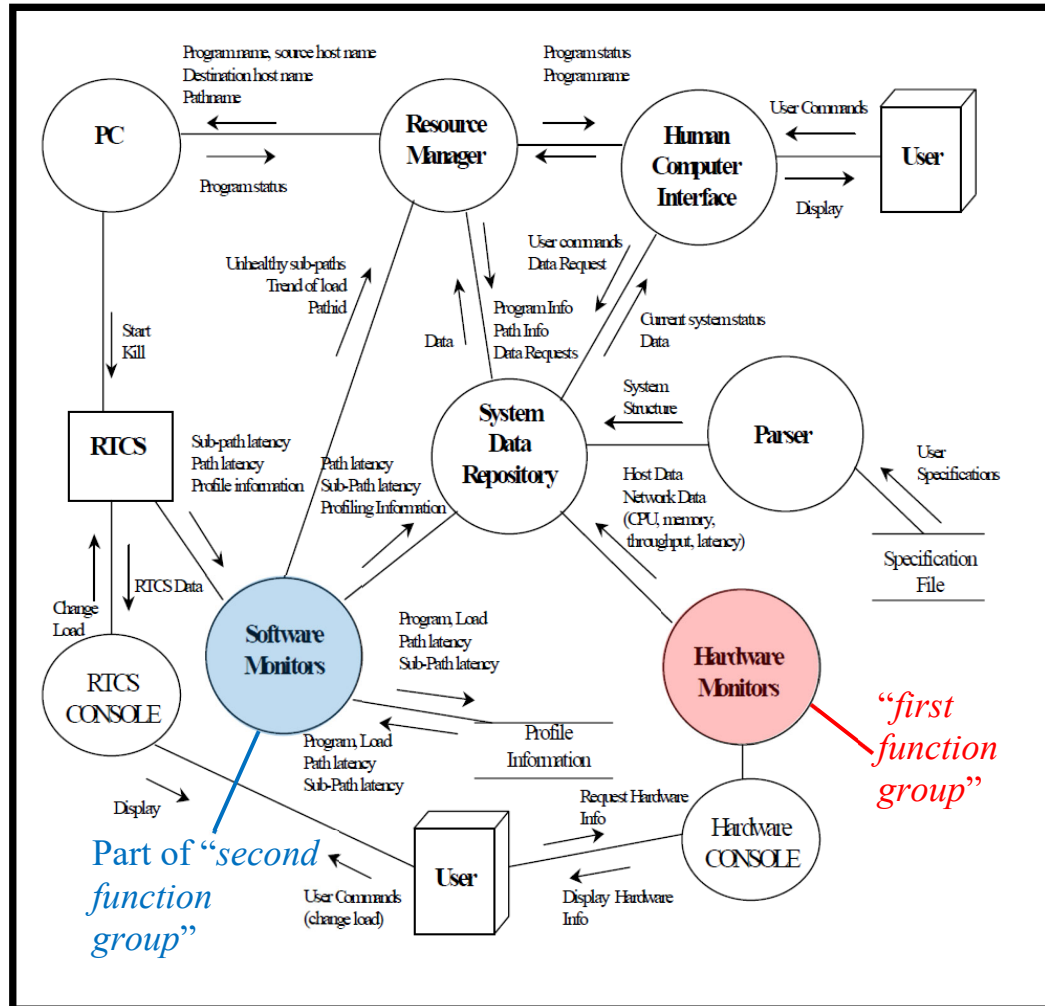
EX1005, Welch, 2.

154. This event reporting is also described and illustrated in Fig. 3, annotated below. “The RTCS continuously sends the application profile (time stamps or program/path latencies) information to QoS Monitors (step 7 of Figure 3).” EX1005, Welch, 3.



EX1005, Welch, Fig. 3 (annotated).

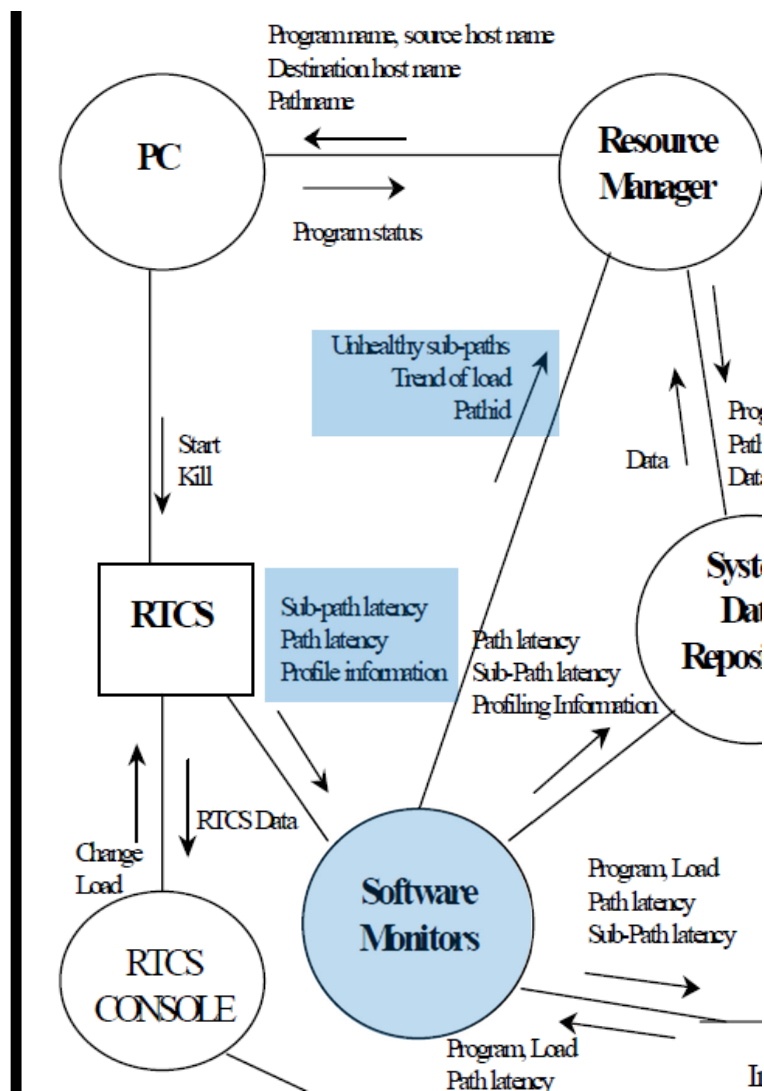
155. Welch explains that the time-stamped events, path latencies, and other pieces of information are provided from the applications to the QoS monitors through “Software Monitors,” as shown in Fig. 2, annotated below. EX1005, Welch, 2.



EX1005, Welch, Fig. 2 (annotated).

156. As shown below, in a close-up of annotated Fig. 2, the RTCS applications provide information to the software monitors, including “Sub-path

latency,” “Path latency,” and “Profile information.” EX1005, Welch, Fig. 2. The software monitors then report information to the Resource Manager as “Unhealthy sub-paths,” “Trend of load,” and “Pathid.” EX1005, Welch, Fig. 2. A POSITA would have recognized that these pieces of information along the arrows are application related events being reported.



EX1005, Welch Fig. 2 (partial, annotated).

157. Accordingly, Welch’s QoS management software includes software

monitors that receive time-stamped events, path latencies, profiles information, and other pieces of information for reporting to a QoS monitor. EX1005, Welch, 2-3.

158. Next, Welch’s metrics calculation component (part of the “*second function group*”) provides “*event correlation capabilities*” because the QoS management software “implementation includes the following **capabilities**: ... (2) track **correlation** and filtering algorithms.” EX1005, Welch, 4. For example:

The technology described in this paper was evaluated within the Naval Surface Warfare Center High Performance Computing (NSWC HiPer-D) Testbed, which contains the experimental Navy system described in [10]. The implementation includes the following capabilities: (1) a simulated track source, (2) **track correlation and filtering algorithms**, (3) track data distribution services, (4) a doctrine server and three types of doctrine processing, (5) an engagement server, (6) a display subsystem including Xwindows based tactical displays, submode mediation, and alert routing surface operations, (7) a simulated weapons control system, and (8) identification upgrade capabilities.

EX1005, Welch, 4.

159. Moreover, given that Welch’s “applications are dependent (communicate with each other),” a POSITA would have understood that the performance information of those applications includes “*event correlation capabilities*”—the ability to communicate messages and events between applications. EX1005, Welch, 2.

160. Welch's software monitors and metrics calculation component aggregate the information received from the real time control systems (RTCS) through the software monitors. For example:

The logical architecture of the QoS management software is shown in Figure 1. It behaves as follows. **The application programs of real-time control paths send timestamped events to the metrics calculation component, which calculates path-level QoS metrics and sends them to the QoS monitor.** The monitor checks for conformance of observed QoS to required QoS, and notifies the QoS diagnosis component when a QoS violation occurs. QoS diagnosis notifies the action selection component of the cause(s) of poor QoS and recommends actions (e.g., move a program to a different host or LAN, shed a program, or replicate a program) to improve QoS. Action selection ranks the recommended actions, identifies redundant actions, and forwards the results to the allocation analysis component; this component consults resource discovery for host and LAN load index metrics, determines a good way to allocate the hardware resources in order to perform the actions, and requests the actions be performed by the allocation enactment component.

EX1005, Welch, 2; *see also* EX1005, Welch, 3 ("The reallocations made by the resource manager make use of information provided by the hardware and **software monitors.**").

161. Moreover, a POSITA would have recognized that Welch's disclosures are similar to how correlation capabilities are described in the '743 patent. *See*

EX1001, '743 patent, 7:51-54 (“The system also provides grammar-driven capabilities for correlating, combining, and reformatting application data into higher-level metrics (composite events) for use by displays or other Resource Management components.”).

162. Accordingly, Welch describes software monitors that receive and report events, which is one sub-function. Welch also describes a metric calculation component, which is another sub-function that provides aggregated data from the software applications to the QoS monitor. These two sub-functions work together to collect and compile data related to the RTCS applications and thus render obvious “*a second function group.*” EX1005, Welch, 2-3.

163. Thus, because Welch’s QoS system includes software monitors and a metrics calculation component that report events and provide aggregated information to a QoS monitor, Welch renders obvious “*a second function group which provides application computer program event reporting and event correlation capabilities.*”

164. In sum, Welch discloses “a second function group [*e.g.*, software monitors] which provides application computer program [*e.g.*, RTCS applications] event reporting [*e.g.*, unhealth sub-paths, trend of load, etc.] and event correlation [*e.g.*, correlation and filtering algorithms] capabilities.”

(d) [10.3] a third function group which provides reasoning and decision making capabilities for the resource managed system,

165. Welch discloses or at least renders obvious limitation [10.3].

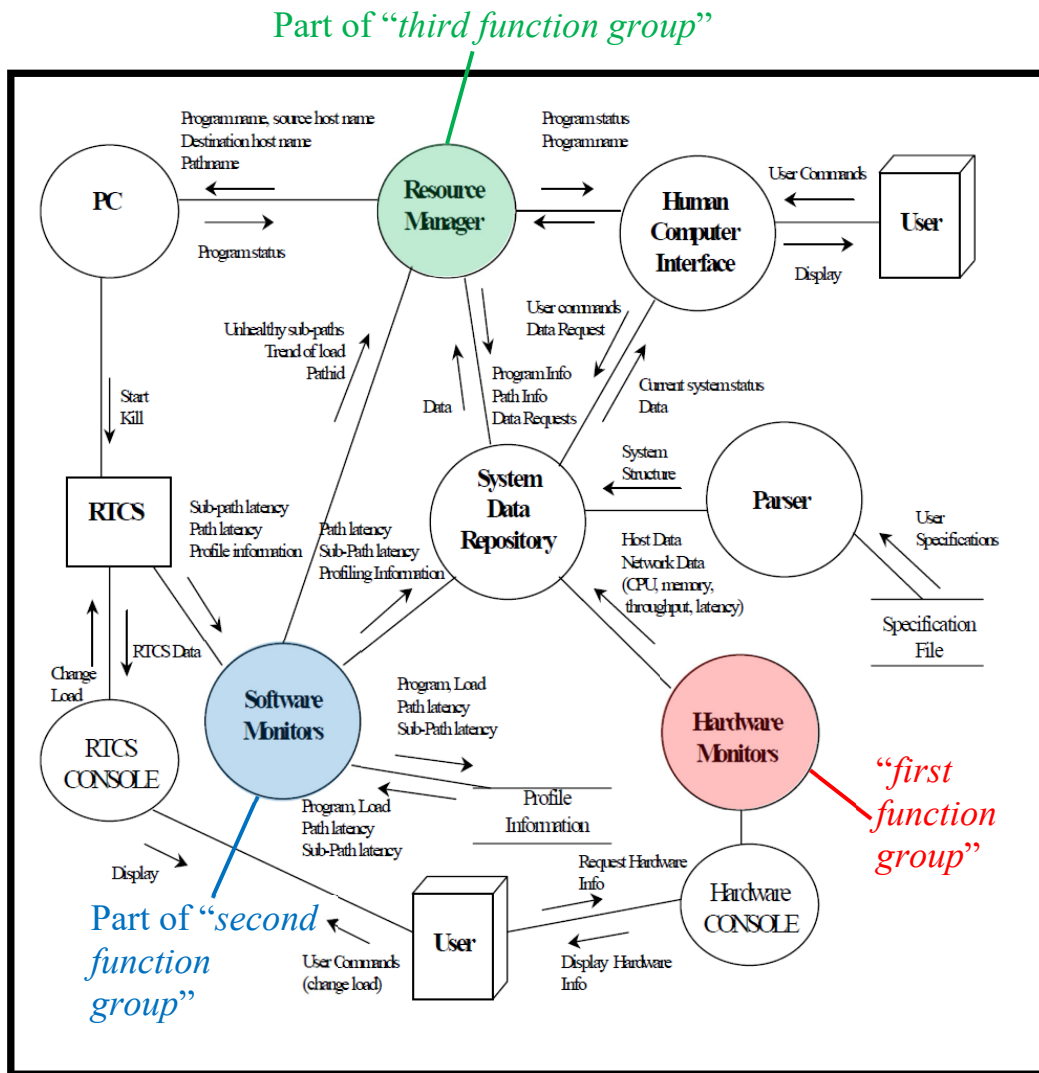
166. First, the “resource manager” and QoS monitors of Welch’s QoS management software teach “*a third function group*,” as shown in Figs. 2 and 3, annotated below. Welch’s resource manager (“*third function group*”) includes functionality that “*provides reasoning and decision making capabilities for the resource managed system.*” EX1005, Welch, 2-3.

167. As an example, the resource manager in Welch’s QoS management software provides “*reasoning*” and “*decision-making capabilities*” in conformance to QoS requirements. EX1005, Welch, 2-3.

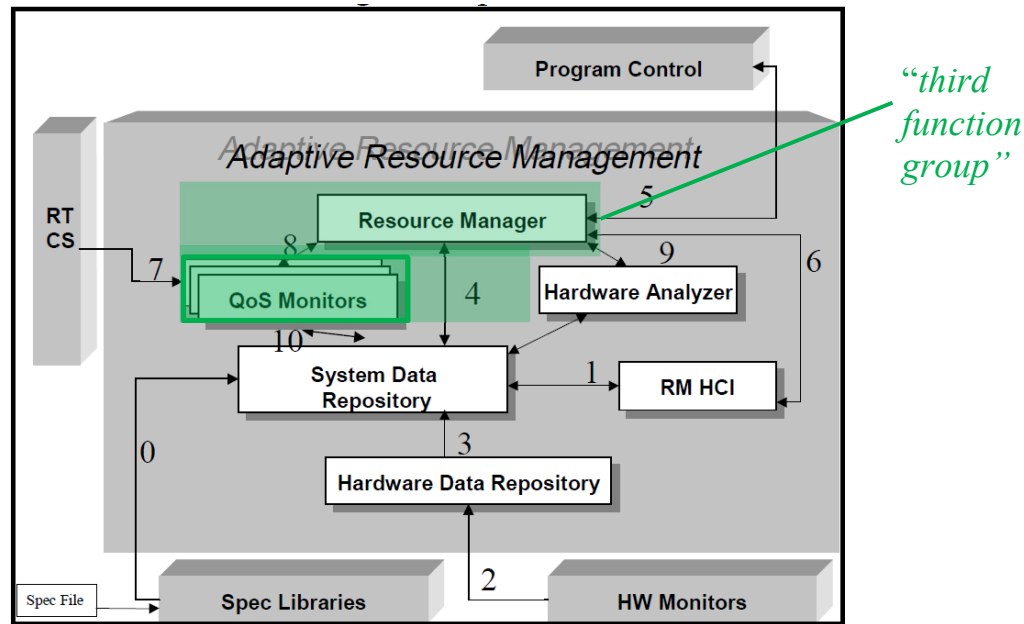
The logical architecture of the QoS management software is shown in Figure 1. It behaves as follows. The application programs of real-time control paths send timestamped events to the metrics calculation component, which calculates path-level QoS metrics and sends them to the QoS monitor. **The monitor checks for conformance of observed QoS to required QoS, and notifies the QoS diagnosis component when a QoS violation occurs. QoS diagnosis notifies the action selection component of the cause(s) of poor QoS and recommends actions (e.g., move a program to a different host or LAN, shed a program, or replicate a program) to improve QoS.** Action selection ranks the recommended actions, identifies redundant actions, and forwards the results to the allocation analysis component; this

component consults resource discovery for host and LAN load index metrics, determines a good way to allocate the hardware resources in order to perform the actions, and requests the actions be performed by the allocation enactment component.

EX1005, Welch, 2.



EX1005, Welch, Fig. 2 (annotated).



EX1005, Welch, Fig. 1 (annotated).

168. Second, Welch's resource manager includes *reasoning ... capabilities* because it includes an action selection component that determines actions to take, prioritizes them, and sends them to be acted on. For example:

The logical architecture of the QoS management software is shown in Figure 1. It behaves as follows. The application programs of real-time control paths send timestamped events to the metrics calculation component, which calculates path-level QoS metrics and sends them to the QoS monitor. The monitor checks for conformance of observed QoS to required QoS, and notifies the QoS diagnosis component when a QoS violation occurs. **QoS diagnosis notifies the action selection component of the cause(s) of poor QoS and recommends actions (e.g., move a program to a different host or LAN, shed a program, or replicate a program) to improve QoS. Action selection ranks the recommended actions, identifies redundant actions, and forwards**

the results to the allocation analysis component; this component consults resource discovery for host and LAN load index metrics, determines a good way to allocate the hardware resources in order to perform the actions, and requests the actions be performed by the allocation enactment component.

EX1005, Welch, 2.

169. The resource manager also includes “*decision making capabilities*” because it includes an allocation analysis component. For example:

The logical architecture of the QoS management software is shown in Figure 1. It behaves as follows. The application programs of real-time control paths send timestamped events to the metrics calculation component, which calculates path-level QoS metrics and sends them to the QoS monitor. The monitor checks for conformance of observed QoS to required QoS, and notifies the QoS diagnosis component when a QoS violation occurs. QoS diagnosis notifies the action selection component of the cause(s) of poor QoS and recommends actions (e.g., move a program to a different host or LAN, shed a program, or replicate a program) to improve QoS. Action selection ranks the recommended actions, identifies redundant actions, and **forwards the results to the allocation analysis component; this component consults resource discovery for host and LAN load index metrics, determines a good way to allocate the hardware resources in order to perform the actions,** and requests the actions be performed by the allocation enactment component.

EX1005, Welch, 2.

170. Additionally, the resource manager decides which actions to take to improve QoS. EX1005, Welch, 2-3. For example:

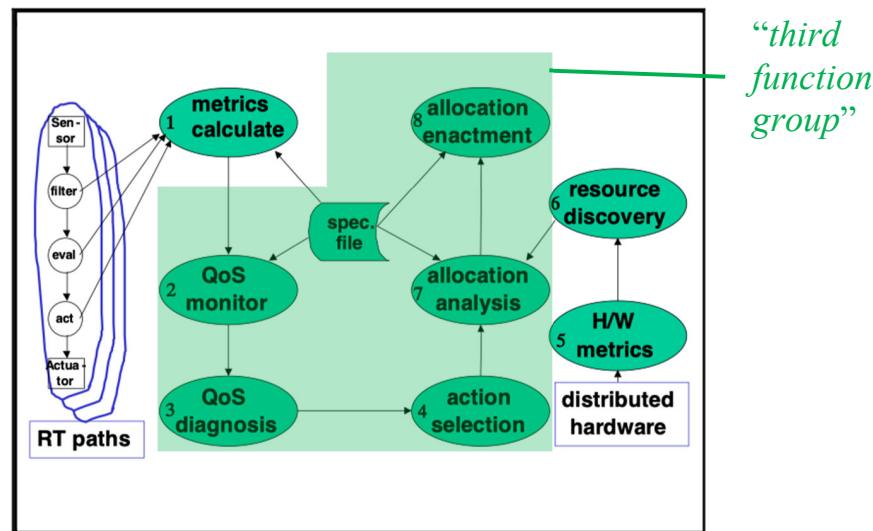
The physical QoS management architecture is shown in Figure 2. **The core component of the middleware is the resource manager.** It is activated when programs die and when time-constrained control paths miss their deadlines. **In response to these events, it takes appropriate measures to improve the quality of service delivered to the applications.** The reallocations made by the resource manager make use of information provided by the hardware and software monitors, as well as from a specification file that describes QoS requirements and the structures of the software system and the hardware system. The system data repository component is responsible for collecting and maintaining all system information. The program control (PC) component consists of a central control program and a set of startup daemons. When resource manager needs to start a program on a particular host, it informs the control program, which notifies the startup daemon on that host. The HCI provides information to the user regarding the system configuration, application status, and reallocation decisions. It also allows the operator to dynamically modify the behavioral characteristics of the resource discovery and the resource manager components.

EX1005, Welch, 2-3.

171. Moreover, “[t]he Resource Manager **needs to decide** on which host(s) and LAN(s) the unhealthy sub-path needs to be replicated or moved. This **decision** is made by choosing the host(s) and LAN(s) with the smallest load indices.”

EX1005, Welch, 3. A POSITA would have clearly recognized these decisions by Welch to be the claimed “*reasoning and decision making capabilities.*”

172. Welch illustrates specific functions that are performed by the Resource Manager in Fig. 3. These specific functions will be discussed with respect to limitations [10.4]-[10.6], in respective §§VIII.A.4(e)-VIII.A.4(g) below.



EX1005, Welch, Fig. 1 (annotated).

173. Thus, because Welch’s QoS management software includes a resource manager that ranks recommended actions and determines a good way to allocate hardware resources, Welch renders obvious “*a third function group which provides reasoning and decision making capabilities for the resource managed system.*”

174. **In sum**, Welch discloses “*a third function group [e.g., resource manager and QoS monitors] which provides reasoning [e.g., ranking recommended actions] and decision making capabilities [e.g., deciding which hosts to place applications] for the resource managed system [e.g., distributed system].*”

(e) [10.4] wherein the third function group comprises: a first function which determines a state and health of the N hosts, a network operatively coupling the N hosts to one another and the M managed characteristic application computer programs in the distributed environment;

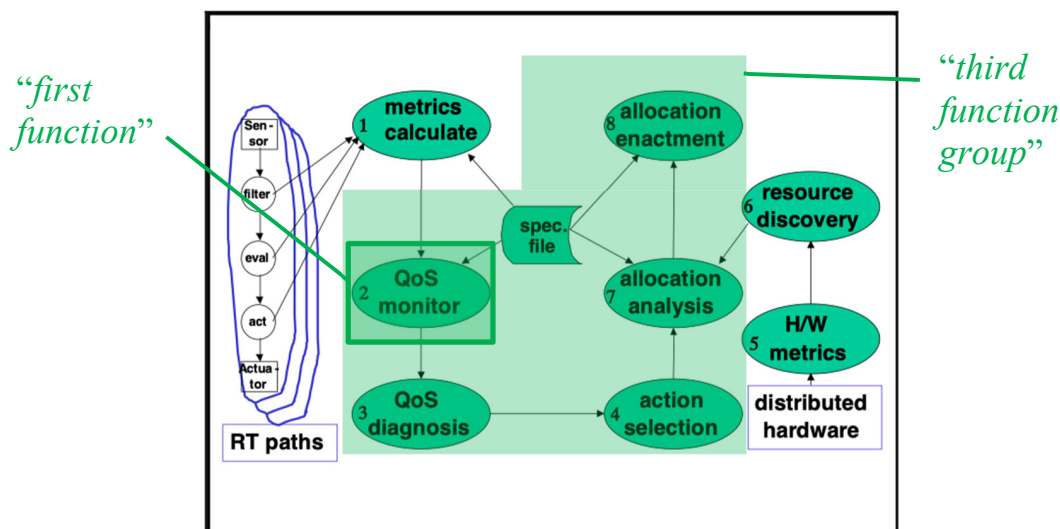
175. Welch discloses or at least renders obvious limitation [10.4].

176. Welch's resource manager ("*third function group*") includes a QoS monitor ("*first function*") that determines the "*state and health*" of the hosts, network, and applications ("*state and health of the N hosts*"). Specifically, Welch discloses that:

The logical architecture of the QoS management software is shown in Figure 1. It behaves as follows. The application programs of real-time control paths send timestamped events to the metrics calculation component, which calculates path-level QoS metrics and sends them to the QoS monitor. The monitor checks for conformance of observed QoS to required QoS, and notifies the QoS diagnosis component when a QoS violation occurs. QoS diagnosis notifies the action selection component of the cause(s) of poor QoS and recommends actions (e.g., move a program to a different host or LAN, shed a program, or replicate a program) to improve QoS. Action selection ranks the recommended actions, identifies redundant actions, and forwards the results to the allocation analysis component; this component consults resource discovery for host and LAN load index metrics, determines a good way to allocate the hardware resources in order to perform the actions, and requests the actions be performed by the allocation enactment component.

EX1005, Welch, 2.

177. The QoS monitor is shown below in annotated Fig. 2.

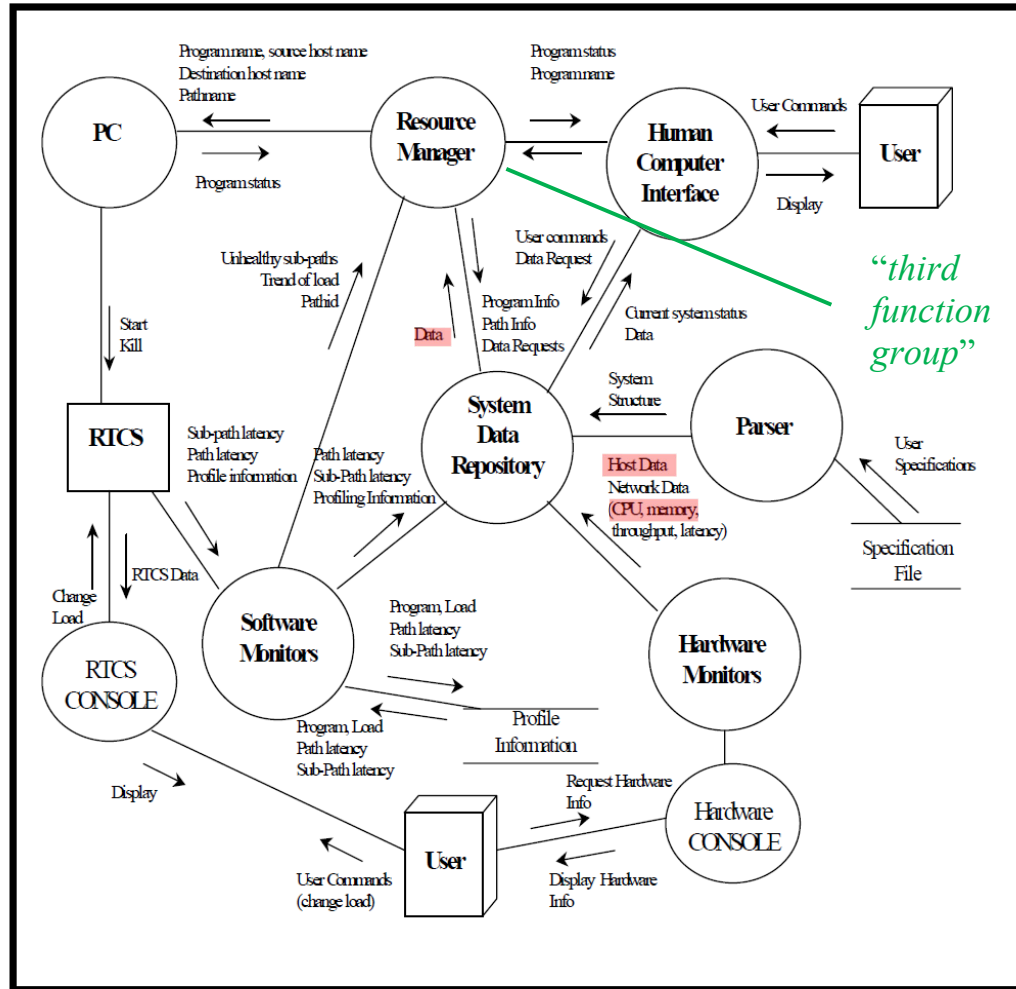


EX1005, Welch, Fig. 1 (annotated).

178. First, the QoS monitor (“*first function*”) of “*the third function group*,” as shown in annotated Fig. 1, above, “*determines a state and health of the N hosts.*” Consistent with the discussion in §VIII.A.4(b) above for limitation [10.1], Welch discloses that the resource manager receives reports from the hardware monitors for various metrics such as the status of the CPU or memory (“*state*” of a host). EX1005, Welch, 3.

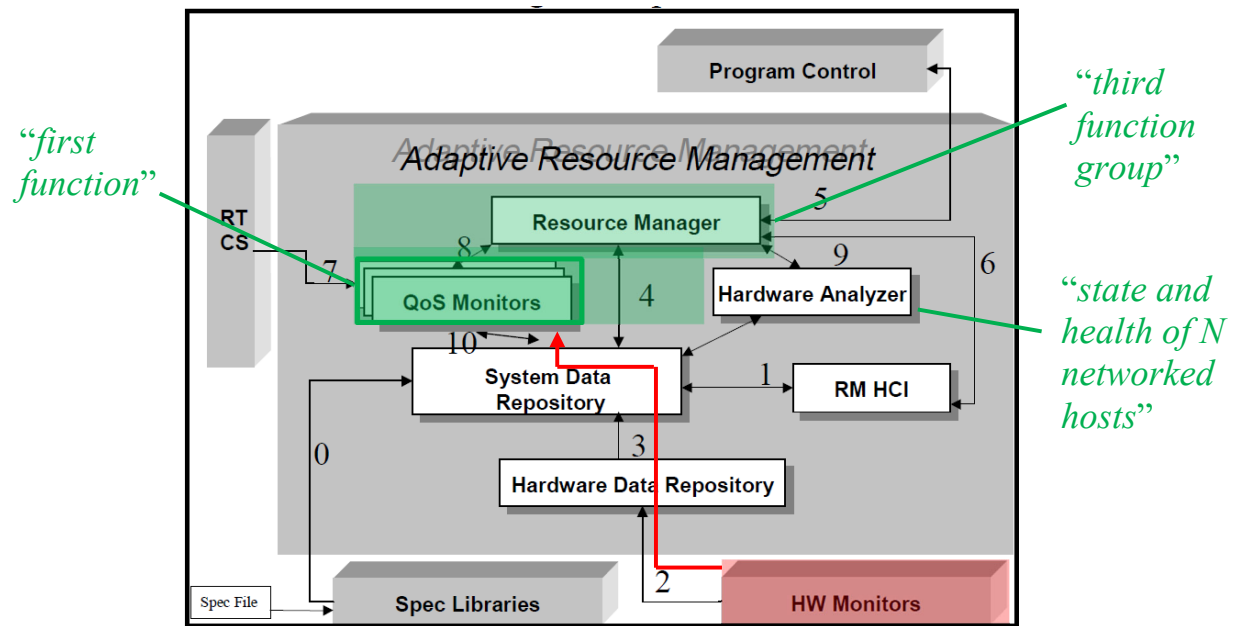
179. Then, “[t]he reallocations made by the resource manager make use of information provided by the hardware and software monitors.” EX1005, Welch, 3. As shown in annotated Fig. 2 below, the resource manager (“*third function group*”) receives data from the system data repository, which includes the

CPU and memory data received from the hardware monitors. EX1005, Welch, Fig. 2.



EX1005, Welch, Fig. 2 (annotated).

180. Welch further illustrates the hardware metrics being provided to the QoS monitor as shown in annotated Fig. 3 below. EX1005, Welch, Fig. 3.



EX1005, Welch, Fig. 1 (annotated).

181. As explained in §VIII.A.4(b) above for limitation [10.1], Welch’s hardware monitors (“*first function group*”) provide information such as a resource’s load index, which represents the “*health*” of the hosts. EX1005, Welch, 2-3. “Hardware (HW) Monitors continuously observe a resource’s load index and pass this information to the Hardware Data Repository (step 2 of Figure 3), which in turn passes such information to the System Data Repository (step 3 of Figure 3).” EX1005, Welch, 3.

182. A POSITA would have recognized Welch’s disclosures to be consistent with how the ’743 patent describes the “health” and “state” of a host. EX1001, ’743 patent, 35:12-18. The ’743 patent explains that status information includes CPU and memory information: “The Hardware Broker FG40 ... advantageously receives operating **system-level statuses** and statistics for each host A-N from the History

Server(s) FG12A-FG12N, respectively. This information can be employed **for calculating CPU**, network, **memory**” EX1001, ’743 patent, 35:12-18.

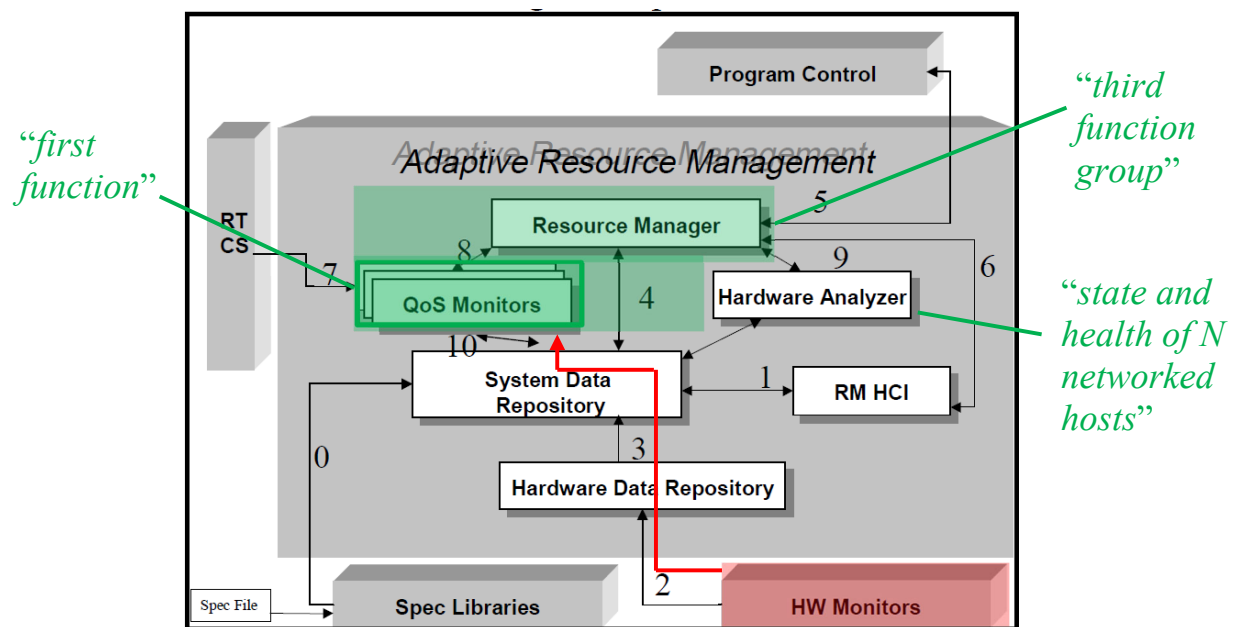
183. The ’743 patent further provides a list of types of information that represent the state and health of the hosts: “The functions associated with this functional group employ information (listed below) to (1) determine the state and health of the distributed environment (hosts, networks, and applications).” EX1001, ’743 patent, 17:9-39 (reproduced and annotated below). This information includes host loads.

System Specifications:
 Host configuration and capabilities
 Application capabilities
 Survivability
 Scalability
 Potential hosts to run on
 Application startup and shutdown dependencies
 Application and path performance requirements
Program Control:
 Application statuses
 Detected application faults
 Detected host failures
 Detection of new host
 Operator initiated requests
 Resolution of application startup or shutdown dependencies
 Selection of application-to-host mappings
History Servers:
 Host statuses, configuration, and loads
 Network link statuses and loads
Remos Network Data Broker:
 Network link statuses and loads
Instrumentation Subsystem:
 Application performance information
Readiness Display:
 Run-time changes to application system priorities

184. Accordingly, just like the ’743 patent, which describes the health and state of a host as the CPU status, memory status, and host load, Welch also describes

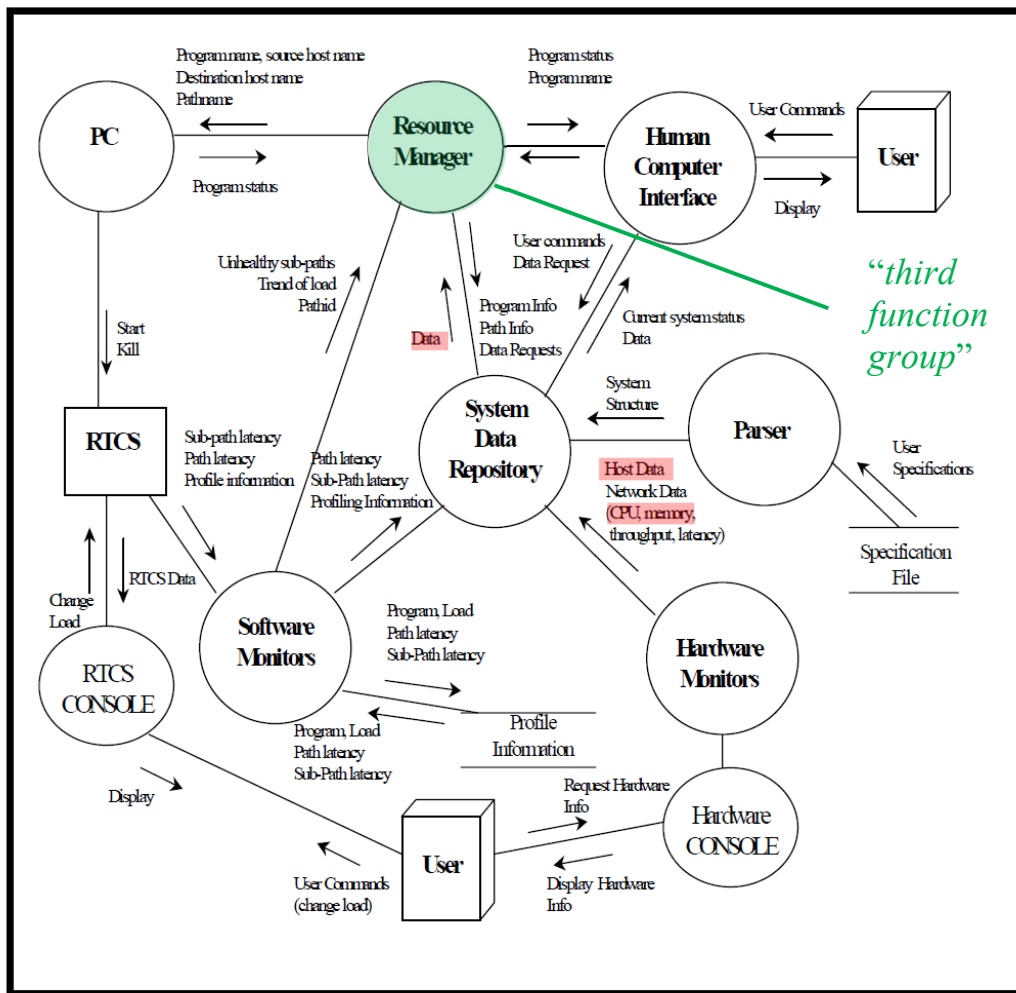
that the QoS monitor (“*first function*” of the “*third function group*”) receives similar data and thus “*determines a state and health of the N hosts.*”

185. Second, Welch’s QoS monitor (“*first function*”) “*determines a state and health of ... a network operatively coupling the N hosts to one another.*” As explained in §VIII.A.4(b) above for limitation [10.1], the hardware monitors report “network data,” which includes throughput (“*state*”) and latency (“*health*”). This information is ultimately received by the QoS monitor within the resource manager (“*third function group*”). EX1005, Welch, 3. The QoS monitors (“*first function*”) is also shown in annotated Fig. 3 below.



EX1005, Welch, Fig. 3 (annotated).

186. As shown in annotated Fig. 2 below, the resource manager receives data from the system data repository, which includes the host and network data.



EX1005, Welch, Fig. 2 (annotated).

187. Third, the QoS monitors (“first function”) of Welch’s resource manager (“third function group”) receive status information from the real-time applications (“determines a state and health of ... the M managed characteristic application computer programs in the distributed environment”). EX1005, Welch, 2-3. In particular, the resource manager receives information from the software monitors as to the status and health of RTCS applications, as described in §VIII.A.4(c) above for limitation [10.2]. For example:

The physical QoS management architecture is shown in Figure 2. **The core component of the middleware is the resource manager.** It is activated when programs die and when time-constrained control paths miss their deadlines. In response to these events, it takes appropriate measures to improve the quality of service delivered to the applications. **The reallocations made by the resource manager make use of information provided by the hardware and software monitors, as well as from a specification file that describes QoS requirements and the structures of the software system and the hardware system.**

The system data repository component is responsible for collecting and maintaining all system information. The program control (PC) component consists of a central control program and a set of startup daemons. When resource manager needs to start a program on a particular host, it informs the control program, which notifies the startup daemon on that host. The HCI provides information to the user regarding the system configuration, application status, and reallocation decisions. It also allows the operator to dynamically modify the behavioral characteristics of the resource discovery and the resource manager components.

EX1005, Welch, 2-3.

188. From the data received from the software monitors, the QoS monitor (“*first function*”) of the resource manager determines the state and health of the RTCS applications. EX1005, Welch, 2 (“The application programs of real-time control paths send timestamped events to the metrics calculation component, which calculates path-level QoS metrics and sends them to the QoS monitor.”); EX1005,

Welch, 3 (“If a path becomes unhealthy (misses its deadline), the QoS Monitors detect such a condition, diagnose the cause of poor health, and suggest an action (such as moving or replicating an application program) to the Resource Manager.”).

189. The resource manager determines the state of the applications through time stamp and path latency information. EX1005, Welch, 2-3. “The RTCS continuously sends the application profile (time stamps or program/path latencies) information to QoS Monitors (step 7 of Figure 3).” EX1005, Welch, 3.

190. The resource manager determines whether the paths (*i.e.*, applications) are healthy or not as well. **“If a path becomes unhealthy (misses its deadline), the QoS Monitors detect such a condition**, diagnose the cause of poor health, and suggest an action (such as moving or replicating an application program) to the Resource Manager (step 8 of Figure 3).” EX1005, Welch, 3.

191. Thus, because Welch describes a QoS monitor that determines the status and health of the hosts, network, and real-time applications, Welch renders obvious “*wherein the third function group comprises: a first function which determines a state and health of the N hosts, a network operatively coupling the N hosts to one another and the M managed characteristic application computer programs in the distributed environment.*”

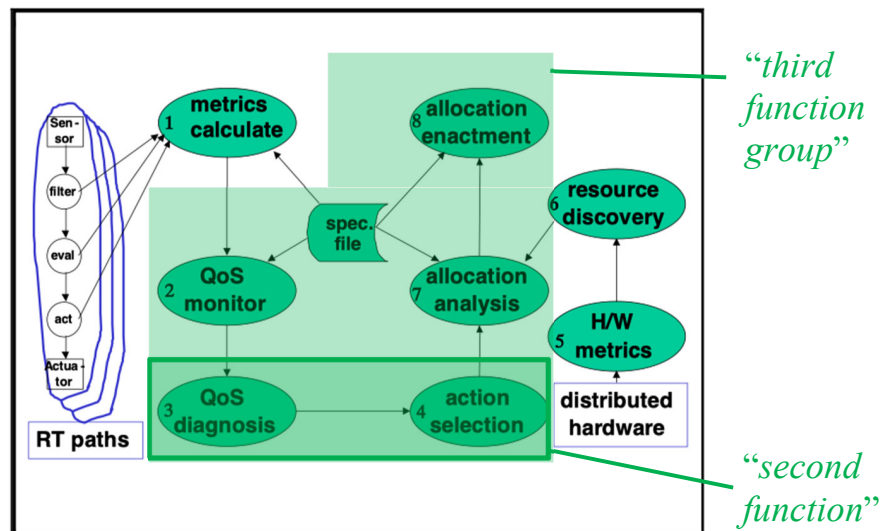
192. **In sum**, Welch discloses “a first function [*e.g.*, QoS Monitors] which determines a state and health [*e.g.*, status and performance] of the N hosts [*e.g.*,

heterogeneous hosts], a network operatively coupling the N hosts to one another [e.g., LAN] and the M managed characteristic application computer programs [e.g., RTCS applications] in the distributed environment [e.g., distributed system].”

(f) [10.5] a second function which determines required allocation and reallocation actions needed to maintain a plurality of Quality of Service (QoS) requirements established for the M managed characteristic application computer programs, the QoS requirements dictating parameters regarding service quality of the M management characteristic application programs; and

193. Welch discloses or at least renders obvious limitation [10.5].

194. First, Welch’s resource manager (“*third function group*”) includes “QoS diagnosis” and “action selection” components (a “*second function*”) that allocate and reallocate resources to meet QoS requirements (“*determines required allocation and reallocation actions needed to maintain a plurality of Quality of Service (QoS) requirements established for the M managed characteristic application computer programs*”). EX1005, Welch, 2-3, Fig. 1 (annotated below).



EX1005, Welch, Fig. 1 (annotated).

195. Welch’s resource manager “determines required allocation and reallocation actions.” Welch explains:

The logical architecture of the QoS management software is shown in Figure 1. It behaves as follows. The application programs of real-time control paths send timestamped events to the metrics calculation component, which calculates path-level QoS metrics and sends them to the QoS monitor. The monitor checks for conformance of observed QoS to required QoS, and notifies the QoS diagnosis component when a QoS violation occurs. **QoS diagnosis notifies the action selection component of the cause(s) of poor QoS and recommends actions (e.g., move a program to a different host or LAN, shed a program, or replicate a program) to improve QoS.** Action selection ranks the recommended actions, identifies redundant actions, and forwards the results to the allocation analysis component; this component consults resource discovery for host and LAN load index metrics, determines a

good way to allocate the hardware resources in order to perform the actions, **and requests the actions be performed by the allocation enactment component.**

EX1005, Welch, 2.

196. Welch's resource manager uses collected information to "*determine[] required allocation and reallocation actions.*" Welch explains:

The physical QoS management architecture is shown in Figure 2. The core component of the middleware is the resource manager. It is activated when programs die and when time-constrained control paths miss their deadlines. In response to these events, it takes appropriate measures to improve the quality of service delivered to the applications.

The reallocations made by the resource manager make use of information provided by the hardware and software monitors, as well as from a specification file that describes QoS requirements and the structures of the software system and the hardware system.

The system data repository component is responsible for collecting and maintaining all system information. The program control (PC) component consists of a central control program and a set of startup daemons. When resource manager needs to start a program on a particular host, it informs the control program, which notifies the startup daemon on that host. The HCI provides information to the user regarding the system configuration, application status, and reallocation decisions. It also allows the operator to dynamically modify the behavioral characteristics of the resource discovery and the resource manager components.

EX1005, Welch, 2-3.

197. Next, Welch's QoS management software determines the actions "*needed to maintain a plurality of Quality of Service (QoS) requirements*" because it includes a QoS diagnosis component. EX1005, Welch, 2. "The application programs of real-time control paths send timestamped events to the metrics calculation component, which calculates path-level QoS metrics and sends them to the **QoS monitor**." EX1005, Welch, 2. "**The monitor** checks for conformance of observed **QoS to required QoS**, and notifies the QoS diagnosis component when a QoS violation occurs." EX1005, Welch, 2; EX1005, Welch, 1 ("dynamic QoS-based resource managers strive to meet the **QoS requirements** of each application they manage.").

198. These QoS diagnoses are used by the action selection component. EX1005, Welch, 2-3. "QoS diagnosis notifies the action selection component of the cause(s) of poor QoS and recommends actions (e.g., move a program to a different host or LAN, shed a program, or replicate a program) to improve QoS." EX1005, Welch, 2. "Action selection ranks the recommended actions, identifies redundant actions, and forwards the results to the allocation analysis component." EX1005, Welch, 2. A POSITA would have clearly understood that Welch's actions apply to a plurality of programs and characteristics. Accordingly, allocation and reallocation actions are done to maintain QoS "*for the M managed characteristic application*

computer programs” by performing the allocation-reallocation actions in accordance with those requirements.

199. Second, Welch discloses that the “*QoS requirements dictat[e] parameters regarding service quality of the M management characteristic application programs.*” EX1005, Welch, 2-3. Welch describes using QoS objectives related to timeliness and dependability (“*parameters*”). Welch explains that “real-time systems have rigorous, multidimensional Quality-of-Service (QoS) objectives. They must behave in a dependable manner, respond to threats in a timely fashion and provide continuous availability within hostile environments.” EX1005, Welch, 1. Thus, dependability, timeliness, and availability are examples of a plurality of QoS requirements regarding the quality of the RTCS applications.

200. Again, Welch discloses that “[t]he reallocations made by the resource manager make use of information provided by the hardware and software monitors, as well as from a specification file that describes QoS requirements and the structures of the software system and the hardware system.” EX1005, Welch, 2. “If a path becomes unhealthy (misses its deadline), the QoS Monitors detect such a condition, diagnose the cause of poor health, and suggest an action (such as moving or replicating an application program) to the Resource Manager (step 8 of Figure 3).” EX1005, Welch, 3. Accordingly, the functions of Welch’s software that allocate and reallocate resources to match QoS requirements are thus a “*second*

function” as claimed.

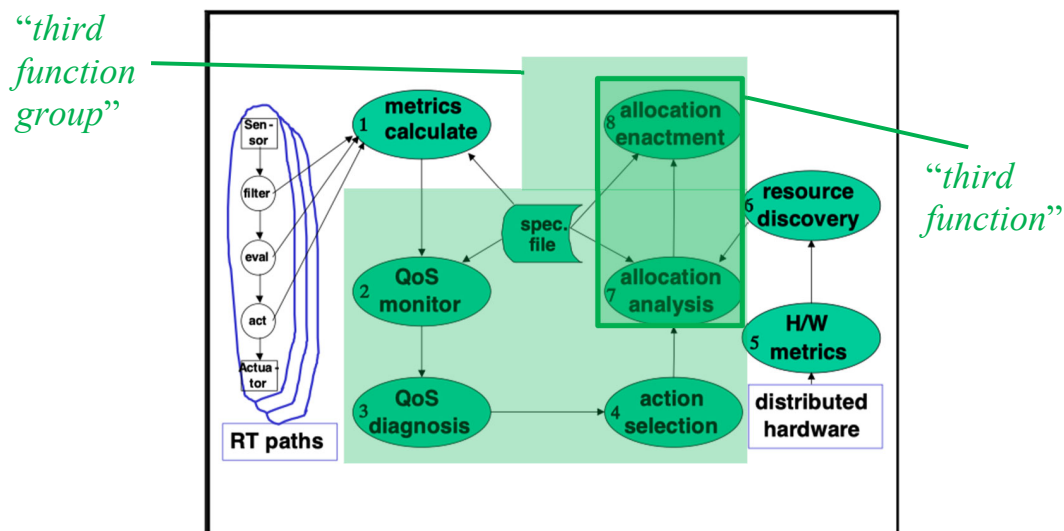
201. Thus, because Welch’s resource manager includes a QoS diagnosis and action selection components that make allocation and reallocation decisions to maintain QoS requirements, Welch renders obvious the limitation “*a second function which determines required allocation and reallocation actions needed to maintain a plurality of Quality of Service (QoS) requirements established for the M managed characteristic application computer programs, the QoS requirements dictating parameters regarding service quality of the M management characteristic application programs.*”

202. In sum, Welch discloses “a second function [*e.g., QoS diagnosis and action selection components*] which determines required allocation and reallocation actions [*e.g., Welch’s allocations and reallocations*] needed to maintain a plurality of Quality of Service (QoS) requirements [*e.g., Welch’s QoS requirements*] established for the M managed characteristic application computer programs [*e.g., RTCS applications*], the QoS requirements dictating parameters regarding service quality [*e.g., timeliness and dependability*] of the M management characteristic application programs [*e.g., RTCS applications*].”

(g) [10.6] a third function which generates automatic control signal requests corresponding to the actions dictated by the QoS requirements, such that the managed characteristic application computer programs are moved, shutdown, and started in accordance with satisfaction of the QoS requirements; and

203. Welch discloses or at least renders obvious limitation [10.6].

204. First, as shown in Fig. 1, annotated below, Welch's resource manager ("*third function group*") includes "allocation analysis" and "allocation enactment" components ("*third function*") that automatically notify a Program Control to make changes to the applications in response to QoS violations ("*generates automatic control signal requests corresponding to the actions dictated by the QoS requirements.*").



EX1005, Welch, Fig. 1 (annotated).

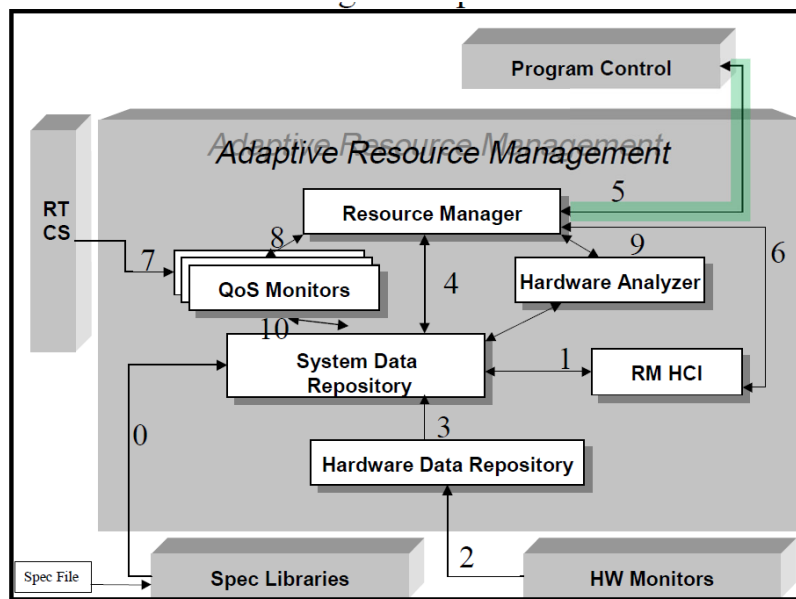
205. Welch's allocation enactment component "consults resource discovery for host and LAN load index metrics, determines a good way to allocate the hardware

resources in order to perform the actions, and **requests the actions be performed by the allocation enactment component.**” EX1005, Welch, 2. For example:

The logical architecture of the QoS management software is shown in Figure 1. It behaves as follows. The application programs of real-time control paths send timestamped events to the metrics calculation component, which calculates path-level QoS metrics and sends them to the QoS monitor. The monitor checks for conformance of observed QoS to required QoS, and notifies the QoS diagnosis component when a QoS violation occurs. QoS diagnosis notifies the action selection component of the cause(s) of poor QoS and recommends actions (e.g., move a program to a different host or LAN, shed a program, or replicate a program) to improve QoS. **Action selection ranks the recommended actions, identifies redundant actions, and forwards the results to the allocation analysis component; this component consults resource discovery for host and LAN load index metrics, determines a good way to allocate the hardware resources in order to perform the actions, and requests the actions be performed by the allocation enactment component.**

EX1005, Welch, 2.

206. “Once a host is selected, the Resource Manager notifies the Program Control to make the change (step 5 of Figure 3) and updates the RM HCI accordingly (step 6 of Figure 3).” EX1005, Welch, 3, Fig. 3 (annotated below).



EX1005, Welch, Fig. 3 (annotated).

207. **Second**, Welch teaches that the actions performed at the direction of the allocation enactment component include replicating, moving, terminating, and starting programs (“*such that the managed characteristic application computer programs are moved, shutdown, and started in accordance with satisfaction of the QoS requirements.*”).

208. For example, “QoS diagnosis notifies the action selection component of the cause(s) of poor QoS and recommends actions (e.g., move a program to a different host or LAN, shed a program, or replicate a program) to improve QoS.” EX1005, Welch, 2. “The Resource Manager needs to decide on which host(s) and LAN(s) the unhealthy sub-path needs to be replicated or moved. EX1005, Welch, 3. “Once a host is selected, the Resource Manager notifies the Program Control to make the change (step 5 of Figure 3).” EX1005, Welch, 3.

209. Additionally, “[t]he Resource Manager informs the Program Control to start the Real-Time Control System (RTCS) application programs on the specified hosts (step 5 of Figure 3).” EX1005, Welch, 3. “The Program Control starts the programs and informs the Resource Manager accordingly (step 5 of Figure 3).” EX1005, Welch, 3. “When resource manager needs to start a program on a particular host, it informs the control program, which notifies the startup daemon on that host.” EX1005, Welch, 3.

210. A POSITA would have recognized that moving, shedding, replicating, and starting programs is the same as the claimed moving, shutting down, and starting programs.

211. Accordingly, Welch’s allocation enactment component of the resource manager moves, replicates, shed, and starts programs (“*the managed characteristic application computer programs are moved, shutdown, and started*”). Moreover, these actions are performed “to improve QoS.” (“*in accordance with satisfaction of the QoS requirements*”). EX1005, Welch, 2.

212. Thus, because Welch’s resource manager uses a program control to move, shed, or startup copies of the RTCS applications to maintain QoS requirements, Welch renders obvious the limitation “*a third function which generates automatic control signal requests corresponding to the actions dictated by the QoS requirements, such that the managed characteristic application*

computer programs are moved, shutdown, and started in accordance with satisfaction of the QoS requirements.”

213. In sum, Welch discloses “a third function [*e.g.*, allocation enactment and allocation analysis components] which generates automatic control signal requests [*e.g.*, Welch’s requests] corresponding to the actions dictated by the QoS requirements [*e.g.*, Welch’s QoS requirements], such that the managed characteristic application computer programs [*e.g.*, RTCS applications] are moved, shutdown, and started [*e.g.*, replicated, moved, shed, and started] in accordance with satisfaction of the QoS requirements [*e.g.*, Welch’s QoS requirements].”

(h) [10.7] a fourth function group which provides program control capabilities permitting starting, stopping, and configuring of selected ones of the M managed characteristic application computer programs on respective ones of the N hosts in the resource managed system,

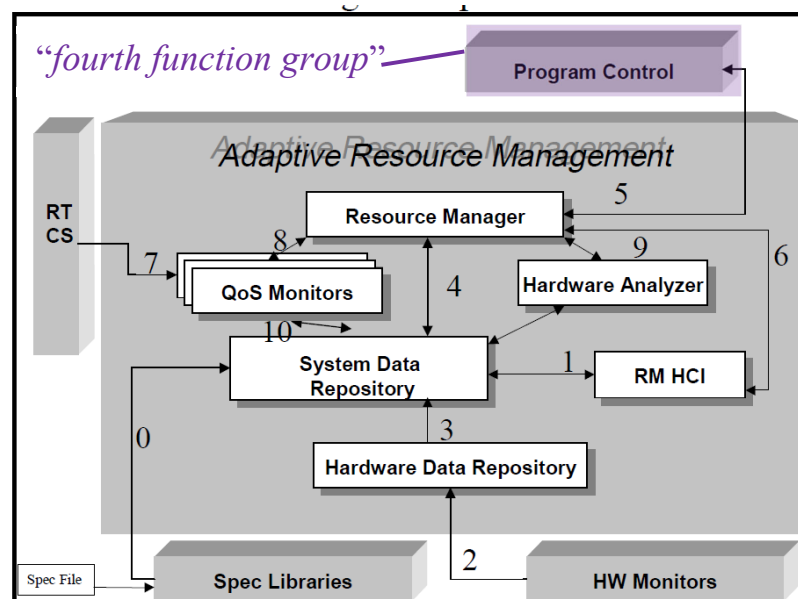
214. Welch discloses or at least renders obvious limitation [10.7].

215. As an initial matter, limitations [10.6] (§VIII.A.4(g) above) and [10.7] are fairly similar. Limitation [10.6] calls for generating requests for QoS-related actions (*e.g.*, move, shutdown, and start an application). Limitation [10.7] merely states the capability to perform these actions. A POSITA would have understood that if the system tracks all these QoS requirements and generates action requests (limitation [10.6]), it would also actually effectuate such requests in this limitation [10.7]. Therefore, much of the evidence I provide in limitation [10.6] (§VIII.A.4(g)

above) applies here as well.

216. Welch describes a program control component (“*fourth function group*”) that “*provides program control capabilities permitting starting, stopping, and configuring of*” RTCS applications as explained below. As described in §VIII.A.4(b) above for limitation [10.1], Welch describes a number of RTCS applications that run on distributed hardware of a resource managed system (“*M managed characteristic application computer programs on respective ones of the N hosts in the resource managed system.*”). EX1005, Welch, 3.

217. Welch’s program control is illustrated in annotated Fig. 3 below.



EX1005, Welch, Fig. 3 (annotated).

218. The program control can “start” and “configure” a program on a particular host. EX1005, Welch, 3. “The program control (PC) component consists

of a central control program and a set of startup daemons. When resource manager needs to **start** a program on a particular host, it informs the control program, which notifies the startup daemon on that host.” EX1005, Welch, 3. “The Resource Manager needs to decide on which host(s) and LAN(s) the unhealthy sub-path needs to be **replicated or moved**.” EX1005, Welch, 3. “The HCI provides information to the user regarding the system configuration, application status, and reallocation decisions.” EX1005, Welch, 3. The software can also “shed a program” (“*stop*”) an application. EX1005, Welch, 2.

219. Welch’s Program Control thus starts, stops, and configures “selected ones of the M managed characteristic application computer programs on respective ones of the N hosts in the resource managed system.” EX1005, Welch, 2-3. This is because “[t]he Resource Manager finds the host(s) and LAN(s) with the smallest load” and “then re-starts the terminated program on that host **by informing the Program Control** (step 5 of Figure 3) and RM HCI (step 6 of Figure 3) accordingly.” EX1005, Welch, 3. The applications are started, stopped, or configured on “*respective*” hosts in that they are started, stopped, or configured on the “specified hosts.” EX1005, Welch, 3.

220. Thus, because Welch’s program control is responsible for starting, stopping, moving, and configuring the programs that run on the plurality of hosts, Welch renders obvious the limitation “*a fourth function group which provides*

program control capabilities permitting starting, stopping, and configuring of selected ones of the M managed characteristic application computer programs on respective ones of the N hosts in the resource managed system.”

221. In sum, Welch discloses “a fourth function group [e.g., program control] which provides program control capabilities [e.g., Welch’s program control functionality] permitting starting, stopping, and configuring [e.g., starting, shedding, configured] of selected ones of the M managed characteristic application computer programs [e.g., RTCS] on respective ones of the N hosts [e.g., specified hosts] in the resource managed system [e.g., distributed system].”

(i) [10.8] where M and N are positive integers and where M may be equal to, greater than, or less than N.

222. Welch renders obvious limitation [10.8].

223. As an initial matter, a POSITA would have recognized that the language “where M may be equal to, greater than, or less than N” merely states that M and N are variables independent of each other. Additionally, the phrase “positive” is redundant because a negative number of hosts or applications is nonsensical.

224. A POSITA would have recognized that the N number of hosts (*see* §VIII.A.4(a) above for limitation [10.0]) and the M number of applications (*see* §VIII.A.4(a) above for limitation [10.0]) in Welch are independent variables. This is because Welch describes that applications can be shed, started, replicated and

moved, which indicates to a POSITA that there is not a 1:1 correspondence between hosts and applications. *See also* EX1005, Welch, 1 (“Traditional load balancing systems assume *independent* jobs.”). Welch describes distributed hardware (independent of the number of hosts) that includes a plurality of heterogeneous hosts. *See supra* §VIII.A.4(a) above for limitation [10.0].

225. Additionally, Welch describes increasing and decreasing the number of applications. EX1005, Welch, 2-3 (“QoS diagnosis notifies the action selection component of the cause(s) of poor QoS and recommends actions (e.g., move a program to a different host or LAN, shed a program, or replicate a program) to improve QoS.”, “The Resource Manager informs the Program Control to start the Real-Time Control System (RTCS) application programs on the specified hosts.”).

226. A POSITA would have found it obvious for these applications to scale independently of the number of heterogeneous hosts. This is because it would be impractical and cumbersome to install or remove a new physical (host) system each time a new application is started or shutdown. Additionally, a POSITA would have found it obvious for Welch’s RTCS applications to scale independently of the number of heterogeneous hosts for at least the reasons described with respect to Ground 2 (see §VIII.B below). Accordingly, because the number of applications is independent of the number of hosts, a POSITA would have understood that in Welch, “*M and N are positive integers and where M may be equal to, greater than,*

or less than N ” as claimed.

227. Thus, because Welch describe systems in which the number of applications is independent from the number of hosts, Welch renders obvious “*where M and N are positive integers and where M may be equal to, greater than, or less than N .*”

228. In sum, Welch discloses “where M [e.g., number of RTCS applications] and N [e.g., number of hosts] are positive integers and where M may be equal to, greater than, or less than N [e.g., M and N are independent variables].

5. Claim 11.

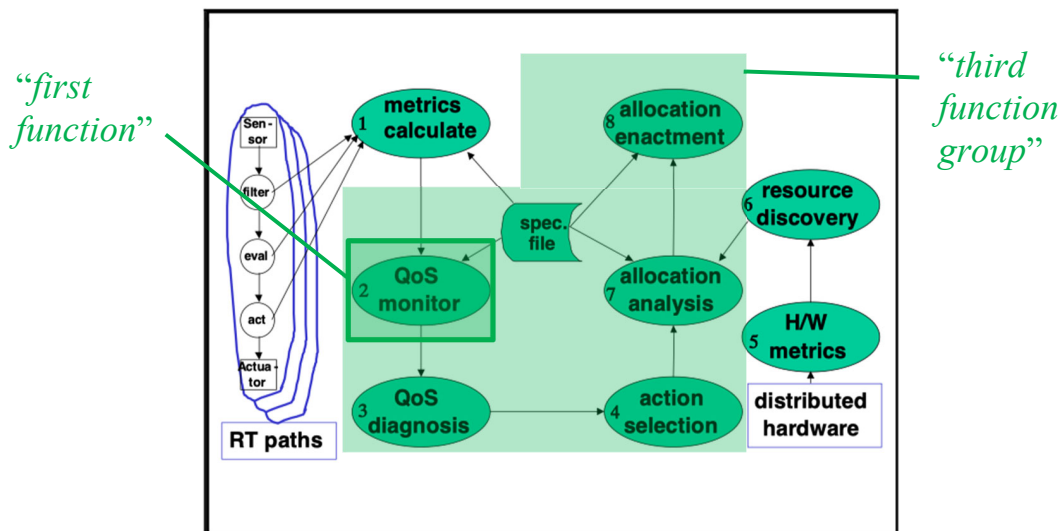
(a) [11] The software as recited in claim 10, wherein the first function receives system specification information comprising host configuration and capabilities.

229. Welch renders obvious limitation [11]. The combination of Welch and Ravindran further renders obvious claim 10 as discussed in §VIII.A.4 above.

230. First, as described in §VIII.A.4(e) above for limitation [10.4], Welch teaches a QoS monitor (“*first function*”).

231. Second, the QoS monitor (“*first function*”) “*receives specific information about host configuration and capabilities.*” EX1005, Welch, 2-3. With reference to Fig. 1, annotated below, Welch’s QoS monitor (“*first function*”) receives QoS metrics: “The application programs of real-time control paths send timestamped events to the metrics calculation component, which calculates path-

level QoS metrics and sends them to the QoS monitor. The monitor checks for conformance of observed QoS to required QoS, and notifies the QoS diagnosis component when a QoS violation occurs.” EX1005, Welch, 2.



EX1005, Welch, Fig. 1 (annotated).

232. The QoS metrics include “*specific information about host configuration and capabilities*” in the form of the status of the CPU or memory (“*configuration and capabilities*” of a host). A POSITA would have understood that the CPU and memory represent the configuration and capabilities of the host in that they represent the processing speed and available memory. Welch explains that the Resource Manager receives data from the hardware and software monitors. For example:

The physical QoS management architecture is shown in Figure 2. The core component of the middleware is the resource manager. It is activated when programs die and when time-constrained control paths

miss their deadlines. In response to these events, it takes appropriate measures to improve the quality of service delivered to the applications.

The reallocations made by the resource manager make use of information provided by the hardware and software monitors, as well as from a specification file that describes QoS requirements and the structures of the software system and the hardware system.

The system data repository component is responsible for collecting and maintaining all system information. The program control (PC) component consists of a central control program and a set of startup daemons. When resource manager needs to start a program on a particular host, it informs the control program, which notifies the startup daemon on that host. **The HCI provides information to the user regarding the system configuration, application status, and reallocation decisions.** It also allows the operator to dynamically modify the behavioral characteristics of the resource discovery and the resource manager components.

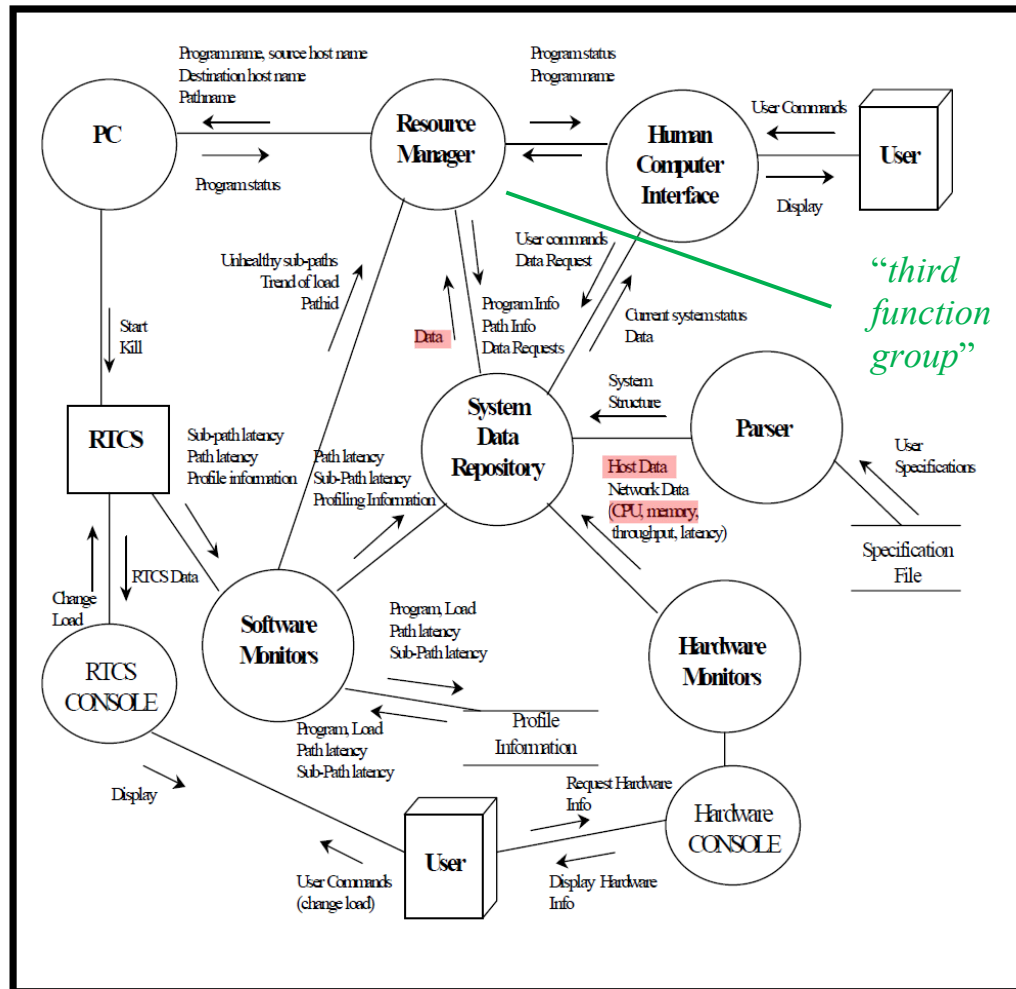
EX1005, Welch, 2-3.

233. Accordingly, Welch's system collects "system configuration" information because it displays it to a user. Moreover, it would have been obvious for the resource manager to receive host configuration and capabilities in order to decide which host can best handle a new application: "The Resource Manger finds the host(s) and LAN(s) with the smallest load indices by querying the Host analyzer." EX1005, Welch, 3.

234. "The reallocations made by **the resource manager make use of**

information provided by the hardware and software **monitors**.” EX1005, Welch,

3. As shown in annotated Fig. 2 below, the Resource Manager (“*third function group*”) receives data from the repository, which includes the CPU and memory data received from the hardware monitors. EX1005, Welch, Fig. 2.



EX1005, Welch, Fig. 2 (annotated).

235. Thus, because Welch’s QoS monitor receives QoS metrics such as the CPU or memory state of the hosts, Welch renders obvious “*wherein the first function receives system specification information comprising host configuration and*

capabilities.”

236. In sum, Welch discloses “[t]he software as recited in claim 10, wherein the first function [*e.g.*, QoS monitors] receives system specification information [*e.g.*, host data] comprising host configuration and capabilities [*e.g.*, CPU and memory configuration].”

6. Claim 14.

(a) [14] The software as recited in claim 10, wherein the first function receives program control information comprising application status and detected application faults for each of the M managed characteristic application computer programs, and detected failures regarding the N hosts.

237. Welch discloses or at least renders obvious limitation [14]. The combination of Welch and Ravindran further renders obvious claim 10 as discussed in §VIII.A.4 above.

238. First, as described in §VIII.A.4(e) above for limitation [10.4], Welch teaches a first function that “determines a state and health of the N hosts ... and the M managed characteristic application computer programs in the distributed environment.” EX1005, Welch, 2-3.

239. Second, the process of determining the state and health of the M applications includes “*receiv[ing] program control information comprising application status.*” EX1005, Welch, 2-3. Welch’s resource manager receives information from the software monitors as to the health of RTCS applications, which

are described as “paths.” EX1005, Welch, 2-3. From that data, the QoS monitor (“*first function*”) of the resource manager determines the state and health of the RTCS applications. For example:

The physical QoS management architecture is shown in Figure 2. The core component of the middleware is the resource manager. It is activated **when programs die and when time-constrained control paths miss their deadlines.** In response to these events, it takes appropriate measures to improve the quality of service delivered to the applications. **The reallocations made by the resource manager make use of information provided by the hardware and software monitors, as well as from a specification file that describes QoS requirements and the structures of the software system and the hardware system.** The system data repository component is responsible for collecting and maintaining all system information. The program control (PC) component consists of a central control program and a set of startup daemons. When resource manager needs to start a program on a particular host, it informs the control program, which notifies the startup daemon on that host. The HCI provides information to the user regarding the system configuration, application status, and reallocation decisions. It also allows the operator to dynamically modify the behavioral characteristics of the resource discovery and the resource manager components.

EX1005, Welch, 2-3.

240. The resource manager determines the state of the applications through time stamp and path latency information (“*application status*”): “The RTCS

continuously sends the application profile (time stamps or program/path latencies) information to QoS Monitors.” EX1005, Welch, 3.

241. Third, the process of determining the state and health of the M applications includes detecting application faults, which teaches “*receiv[ing] program control information comprising ... detected application faults for each of the M managed characteristic application computer programs.*” EX1005, Welch, 2-3. “If a sub-path (RTCS program) is terminated due to some hardware/**software failure**, Program Control detects such a condition and informs the Resource Manager accordingly (step 5 of Figure 3).” EX1005, Welch, 3. “If a path becomes **unhealthy (misses its deadline)**, the QoS Monitors detect such a condition, diagnose the cause of poor health, and suggest an action (such as moving or replicating an application program) to the Resource Manager (step 8 of Figure 3). EX1005, Welch, 3.

242. Welch’s resource manager also receives information as to application failures: “If a sub-path (RTCS program) is terminated due to some hardware/software **failure**, Program Control detects such a condition and informs the Resource Manager accordingly (step 5 of Figure 3).” EX1005, Welch, 3. “The Resource Manager finds the host(s) and LAN(s) with the smallest load indices by querying the Host analyzer (step 9 of Figure 3).” It then re-starts the terminated program on that host by informing the Program Control.” EX1005, Welch, 3.

243. Fourth, the process of determining the state and health of the N hosts includes detecting hardware failures: *“receiv[ing] program control information comprising ... detected failures regarding the N hosts.”* If a sub-path (RTCS program) is terminated due to some **hardware**/software **failure**, Program Control detects such a condition and informs the Resource Manager accordingly (step 5 of Figure 3).” EX1005, Welch, 3.

244. Thus, because the QoS monitor of Welch’s resource manager receives information as to application and host statuses, faults (e.g., missing a deadline), and failures (e.g., terminations), Welch renders obvious *“wherein the first function receives program control information comprising application status and detected application faults for each of the M managed characteristic application computer programs, and detected failures regarding the N hosts.”*

245. In sum, Welch discloses “[t]he software as recited in claim 10, wherein the first function [e.g., QoS monitors] receives program control information [e.g., information from the software monitors] comprising application status [e.g., time stamps or program/path latencies] and detected application faults [e.g., missed deadlines] for each of the M managed characteristic application computer programs [e.g., RTCS applications], and detected failures regarding the N hosts [e.g., hardware/software failure].”

7. Claim 16.

(a) [16] The software as recited in claim 10, wherein the first function receives application performance data representing each one of the M managed characteristic application computer programs.

246. Welch discloses or at least renders obvious limitation [16]. The combination of Welch and Ravindran further renders obvious claim 10 as discussed in §VIII.A.4(a) above.

247. First, as described in §VIII.A.4(e) above for limitation [10.4], Welch teaches a first function that “determines a state and health of the N hosts, a network operatively coupling the N hosts to one another and the M managed characteristic application computer programs in the distributed environment.”

248. Second, the process of determining the state and health of the N hosts includes “*receiv[ing] application performance data representing each one of the M managed characteristic application computer programs.*” Welch explains that the “RTCS [applications] continuously sends the application profile (time stamps or program/path latencies) information to QoS Monitors (step 7 of Figure 3).” EX1005, Welch, 3. The program/path latencies are “*performance data*” as claimed. Welch further states that “dynamic QoS-based resource managers strive to meet the QoS requirements of **each** application they manage.” EX1005, Welch, 1.

249. Thus, because Welch’s QoS monitors receive performance data in the form of program/path latencies, Welch renders obvious “wherein the first function

receives application performance data representing each one of the M managed characteristic application computer programs.”

250. In sum, Welch discloses “[t]he software as recited in claim 10, wherein the first function [e.g., QoS monitors] receives application performance data [e.g., program path latencies] representing each one [e.g., each application] of the M managed characteristic application computer programs [e.g., RTCS applications].”

8. Claim 18.

(a) [18.0] The software as recited in claim 10, wherein the second function which determines the required allocation and reallocation actions established for the M managed characteristic application computer programs by:

251. The combination of Welch and Ravindran renders obvious claim 10 as discussed in §VIII.A.4 above.

252. For the reasons explained in §VIII.A.4(f) above for limitation [10.5], and to the extent the preamble is limiting, Welch teaches a “second function which determines the required allocation and reallocation actions established for the M managed characteristic application computer programs.”

(b) [18.1] responding to application and host failures by determining if and what recovery actions should be taken;

253. First, the action selection and QoS diagnosis components of Welch’s resource manager (“*second function*”) responds to a QoS violation (“*responding to application and host failures*”) by moving, shedding, or replicating a program (“*by*

determining if and what recovery actions should be taken”). EX1005, Welch, 2-3.

For example:

The logical architecture of the QoS management software is shown in Figure 1. It behaves as follows. The application programs of real-time control paths send timestamped events to the metrics calculation component, which calculates path-level QoS metrics and sends them to the QoS monitor. The monitor checks for conformance of observed QoS to required QoS, and notifies the QoS diagnosis component when a QoS violation occurs. QoS diagnosis notifies the action selection component of the cause(s) of poor QoS and recommends actions (e.g., move a program to a different host or LAN, shed a program, or replicate a program) to improve QoS. **Action selection ranks the recommended actions, identifies redundant actions, and forwards the results to the allocation analysis component; this component consults resource discovery for host and LAN load index metrics, determines a good way to allocate the hardware resources in order to perform the actions, and requests the actions be performed by the allocation enactment component.**

EX1005, Welch, 2.

254. QoS violations may include host or application failures: “If a path becomes unhealthy (misses its deadline), the QoS Monitors detect such a condition, diagnose the cause of poor health, and suggest an action (such as moving or replicating an application program) to the Resource Manager (step 8 of Figure 3).”

EX1005, Welch, Fig. 3. A POSITA would have understood that when a service

misses its deadline (*e.g.*, to respond to requests in a timely fashion), it could be because of temporary congestion or permanent failure such as hardware/software crashes. In either case, such QoS violations constitute application and/or host failures. *See also* §V.A.1 and §V.B above.

255. Thus, because the action selection and QoS diagnosis components of Welch's resource manager respond to a QoS violation by "mov[ing] a program to a different host or LAN, shed[ding] a program, or replicat[ing] a program," Welch renders obvious "responding to application and host failures by determining if and what recovery actions should be taken." EX1005, Welch, 2.

256. In sum, Welch discloses "responding to application [*e.g.*, software failures] and host [*e.g.*, hardware failures] failures by determining if and what recovery actions [*e.g.*, moving, shedding, or replicating a program] should be taken."

(c) [18.2] determining if and where to place new copies of managed characteristic application computer programs or which managed characteristic application computer programs should be shutdown when QoS Manager functions indicate that scale up or scale down actions should be taken based on measured application performance and QoS specifications established for the M managed characteristic application computer programs;

257. As an initial matter, I note that this limitation (and indeed all of claim 18) merely claims traditional load-balancing: measuring performance/uptime, then moving and/or re/starting services on one or more nodes. *See* §V.B.1 above. Developing and managing load-balanced systems was well known in the art at least

as early as the 1990s. (I personally helped develop such technologies as discussed in my qualifications §II above.)

258. Welch renders both alternatives in limitation [18.2] obvious. Limitation [18.2] recites “(i) determining if and where to place new copies of managed characteristic application computer programs ... **or** (ii) [determining] which managed characteristic application computer programs should be shutdown....”

259. First, Welch teaches the first option of “(i) *determining if and where to place new copies of managed characteristic application computer programs.*” EX1005, Welch, 2-3. “The Resource Manager informs the Program Control to **start** the Real-Time Control System (RTCS) application programs on the specified hosts (step 5 of Figure 3).” EX1005, Welch, 3. “The Resource Manager needs to decide on which host(s) and LAN(s) the unhealthy sub-path needs to be **replicated** or moved. This decision is made by choosing the host(s) and LAN(s) with the smallest load indices.” EX1005, Welch, 3.

260. Second, Welch teaches the alternative option: “(ii) *determining ... which managed characteristic application computer programs should be shutdown when QoS Manager functions indicate that scale up or scale down actions should be taken based on measured application performance and QoS specifications established for the M managed characteristic application computer programs.*”

EX1005, Welch, 2-3. “The monitor checks for conformance of observed QoS to required QoS, and notifies the QoS diagnosis component when a QoS violation occurs. QoS diagnosis notifies the action selection component of the cause(s) of poor QoS and recommends actions (e.g., move a program to a different host or LAN, shed a program, or replicate a program) to improve QoS.” EX1005, Welch, 2.

261. In this context, a POSITA would have understood that “shedding” a program is synonymous with shutting it down. Moreover, “moving” a program from host A to host B necessarily implies shutting it down on host A while starting it on host B.

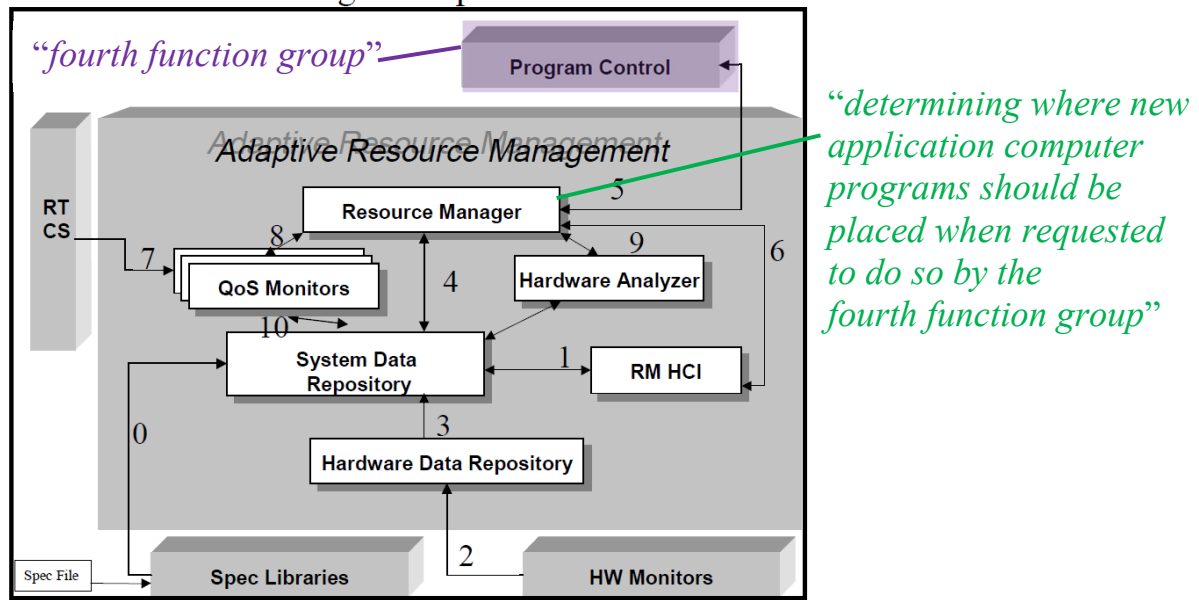
262. Thus, because Welch’s resource manager decides which applications to replicate, and where to place them, Welch renders obvious “*determining if and where to place new copies of managed characteristic application computer programs or which managed characteristic application computer programs should be shutdown when QoS Manager functions indicate that scale up or scale down actions should be taken based on measured application performance and QoS specifications established for the M managed characteristic application computer programs.*”

263. In sum, Welch discloses “determining if and where to place new copies of managed characteristic application computer programs [e.g., starting or moving RTCS applications] or which managed characteristic application computer programs

should be shutdown [*e.g.*, shed] when QoS Manager functions indicate that scale up or scale down actions [*e.g.*, starting or shedding] should be taken based on measured application performance [*e.g.*, observed QoS] and QoS specifications [*e.g.*, required QoS] established for the M managed characteristic application computer programs [*e.g.*, RTCS applications].”

(d) [18.3] determining where new application computer programs should be placed when requested to do so by the fourth function group; and

264. Welch’s resource manager determines where to place an application (“*determining where new application computer programs should be placed*”). EX1005, Welch, 3. “The Resource Manager finds the host(s) and LAN(s) with the smallest load indices by querying the Host analyzer (step 9 of Figure 3).” EX1005, Welch, 3. “It then re-starts the terminated program on that host by informing the Program Control (step 5 of Figure 3) and RM HCI (step 6 of Figure 3) accordingly.” EX1005, Welch, 3. This is done in response to being informed of a failure by the program control (“*when requested to do so by the fourth function group*”). “If a sub-path (RTCS program) is terminated due to some hardware/software failure, Program Control detects such a condition and informs the Resource Manager accordingly (step 5 of Figure 3).” EX1005, Welch, 2-3.



EX1005, Welch, Fig. 3 (annotated).

265. Thus, because Welch’s program control detects a failure and the resource manager responsively determines where to restart the terminated program, Welch renders obvious “*determining where new application computer programs should be placed when requested to do so by the fourth function group.*”

266. In sum, Welch discloses “determining where new application computer programs [*e.g.*, newly started RTCS applications] should be placed [*e.g.*, on which host they are executed] when requested to do so by the fourth function group [*e.g.*, program control group].”

(e) [18.4] determining which and how many application computer programs should run based on application system priorities.

267. Welch's resource manager decides which and how many applications

should run based in part on a specification file that describes QoS specifications (“*application system priorities*”). “The System Data Repository loads the user spec file “via the Spec Libraries, which consist of a compiler and data structures to store the compiled real-time system and **application QoS specifications** (step 0 of Figure 3).” EX1005, Welch, 3. “The System Data Repository sends the system information to Resource Management Human Computer Interface (RM HCI) for display purposes (step 1 of Figure 3).” EX1005, Welch, 3.

268. “Hardware (HW) Monitors continuously observe a resource’s load index and pass this information to the Hardware Data Repository (step 2 of Figure 3), which in turn passes such information to the System Data Repository (step 3 of Figure 3).” EX1005, Welch, 3. “The Resource Manager receives the initial startup information from the System Data Repository, as specified by the spec file (step 4 of Figure 3).” EX1005, Welch, 3. “The Resource Manager informs the Program Control to **start the Real-Time Control System (RTCS) application programs on the specified hosts** (step 5 of Figure 3).” EX1005, Welch, 3. “The Program Control starts the programs and informs the Resource Manager accordingly (step 5 of Figure 3).” EX1005, Welch, 3.

269. Welch’s system also determines the placement of applications “*based on application system priorities*” in that it considers the load index. EX1005, Welch, 3. “The Resource Manager needs to decide on which host(s) and LAN(s) the

unhealthy sub-path needs to be replicated or moved. This decision is made by choosing the host(s) and LAN(s) **with the smallest load indices.**” EX1005, Welch,

3. “The Hardware Analyzer ranks the hosts and LANs in ascending order of their load index and passes this information to the Resource Manager (step 9 of Figure 3).” EX1005, Welch, 3.

270. Thus, because Welch’s resource manager places applications based on a spec file and load indices, Welch renders obvious “*determining which and how many application computer programs should run based on application system priorities.*”

271. In sum, Welch discloses “determining which and how many application computer programs [*e.g.*, RTCS applications] should run [*e.g.*, executed on a host] based on application system priorities [*e.g.*, consideration of load indices].”

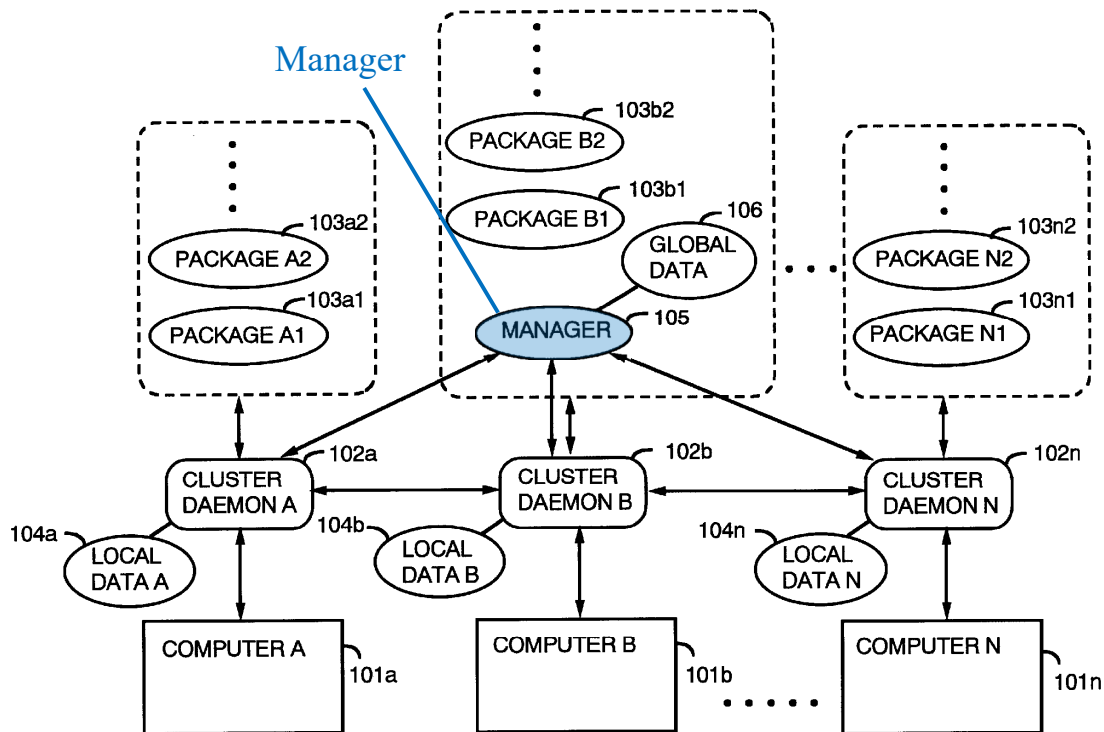
B. Ground 2: Claims 10, 11, 14, 16, and 18 are obvious under 35 U.S.C. § 103 over Welch, Ravindran, and Hosokawa.

272. To the extent Patent Owner argues that Welch does not explicitly provide an example in which $M > N$, $M = N$, or $M < N$, it would have been obvious for Welch’s distributed system to work in that manner as evidenced by Hosokawa. As I discussed in §VIII.A.4(i) above in relation to claim [10.8] (Ground 1), that claim language merely implies that, mathematically, M and N are independent positive integers.

273. In addition, to the extent that the “*at least one host*” on which the “*Software*” is stored is one of the “*N networked hosts*” managed by the software, it would have been obvious for Welch’s QoS management software to be stored on one of the “*N networked hosts*” as also taught by Hosokawa.

1. Hosokawa

274. Like Welch and the ’743 patent, Hosokawa relates to a “cluster controlling system [that] transfers packages which have been operating on one computer to another computer when a fault or failure has occurred by monitoring and controlling the packages in the entire system.” EX1007, Hosokawa, Abstract. Hosokawa describes a system in which the manager resides on one of the hosts being managed. “The manager 105 is one of the packages installed in the cluster system.” EX1007, Hosokawa, 7:16-17. “If a fault or failure occurs in the manager or in the computer running the manager, the manager is re-started on another computer by a cluster daemon.” EX1007, Hosokawa, Abstract. Hosokawa’s manager is illustrated as being run on one of the computer nodes in annotated Fig. 1 below.

**FIG. 1****EX1007, Hosokawa, Fig. 26a (annotated).**

275. Hosokawa refers to programs using several interchangeable terms. “In this specification, package program is a general term referring to application programs and to service programs.” EX1007, Hosokawa, 7:3-5.

276. Hosokawa also provides an explicit example in which the number of hosts (N) is equal to or independent from the number of applications (M) running on those hosts.

277. For example, Hosokawa illustrates an example in which the number of applications (referred to as software packages in Hosokawa) is the same as the number of hosts (referred to as nodes in Hosokawa). In other words, Hosokawa provides an example in which M is equal to N, as shown in annotated Fig. 26A

below.

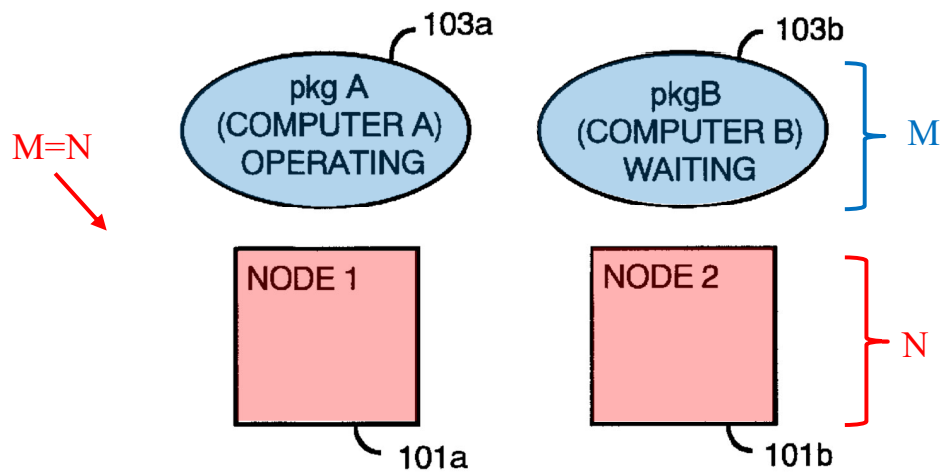


FIG. 26A

EX1007, Fig. 26a (annotated).

278. Additionally, Hosokawa illustrates an example where there are fewer applications M than hosts N ($M < N$). Such example is shown in annotated Fig. 37A below, where $M=2$ and $N=n$ ($n > 2$ and hence $N > 2$).

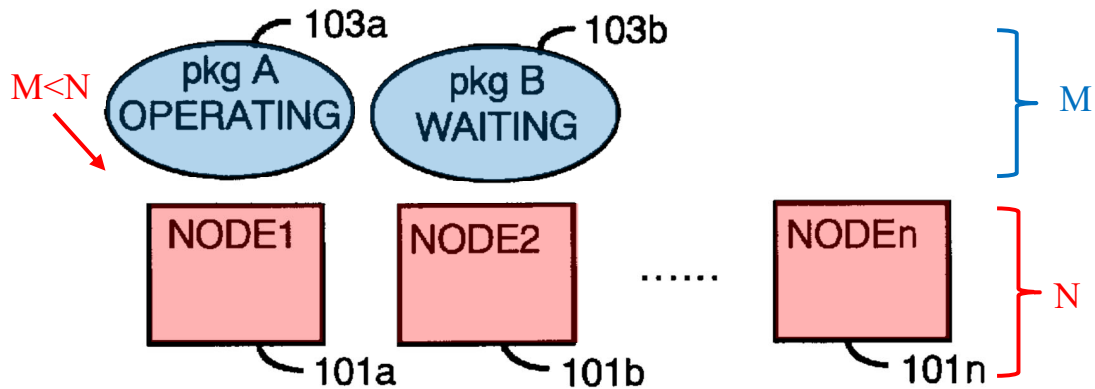


FIG. 37A

EX1007, Fig. 37a (annotated).

279. Finally, Hosokawa further illustrates an example where there are more applications M than hosts N ($M > N$). Such example is shown in annotated Fig. 46A below, where M is shown as at least double that of N (each host in N runs at least two applications M).

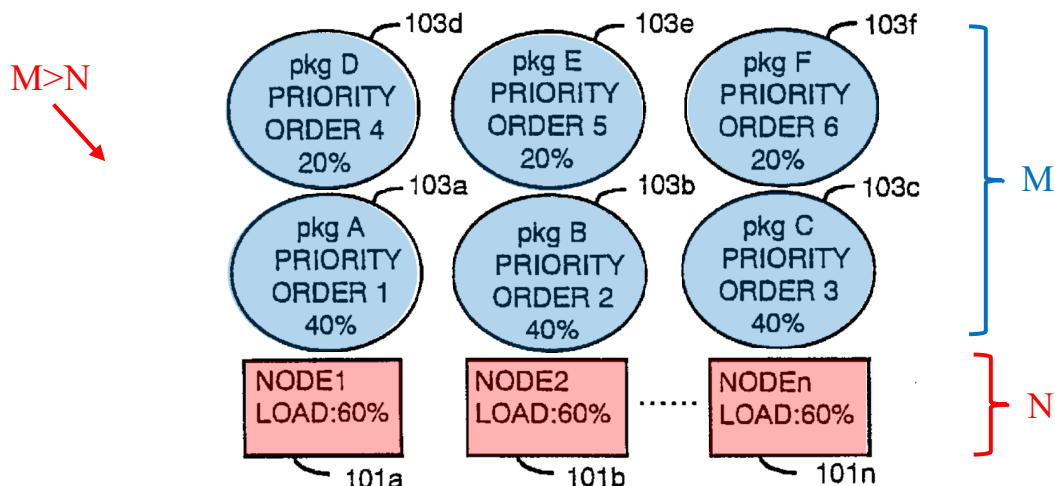


FIG. 46A

EX1007, Fig. 26a (annotated).

280. Accordingly, Hosokawa provides the general teaching that it was known that in distributed systems, the number of applications can be independent from the number of hosts.

281. Hosokawa is analogous art to the '743 patent because it is directed to the same field of endeavor (distributed computing) as the '743 patent. *Compare* EX1001, '743 patent, Abstract (“A resource manager for a distributed environment”) *with* EX1007, Hosokawa, Title (“Cluster Controlling System Operating on a Plurality of Computers in a Cluster System.”).

282. In addition, Hosokawa describes the same application survivability techniques that are reasonably pertinent to the problem purportedly solved by the '743 patent. *See* EX1007, Hosokawa, Abstract (The system “transfers packages which have been operating on one computer to another computer when a fault or failure has occurred.”). The '743 patent likewise purports to address survivability. *See* EX1001, '743 patent, 2:15-20 (seeking to “provide a system that is both flexible and adaptive (or at least easily adapted) to changes in system configuration, performance bottlenecks, survivability requirements, scalability, etc.”).

2. Motivation to combine Welch, Ravindran, and Hosokawa

283. A POSITA would have been motivated to combine Welch and Ravindran for the reasons described in §VIII.A.3 above.

284. A POSITA would have been motivated to implement Welch's

hardware monitors according to the conventional implementation details as described in Ravindran, which is expressly referenced in Welch, as described in §VIII.A.4(b) above for Ground 1, limitation [10.1].

285. A POSITA would have found it obvious for Welch’s number of applications to be independent of the number of hosts, as explicitly shown by Hosokawa. A POSITA also would have found it obvious for Welch’s QoS management software to run on one of the hosts that it manages, as taught by Hosokawa. These are conventional implementation details that were well known in the art, as shown by Hosokawa. *See* §V.B.1 above.

(a) The number of applications are independent from the number of hosts.

286. Welch describes a distributed system that includes a plurality of hosts and LANs. EX1005, Welch, 2-3. Indeed, the QoS management software is to “recommend actions (e.g., move a program to a **different host or LAN**,” which indicates that the “distributed hardware” includes multiple hosts. EX1005, Welch, 2. These hosts support a plurality of RTCS applications. EX1005, Welch, 2-3. “The Resource Manager informs the Program Control to start the Real-Time Control System (RTCS) application programs on the specified hosts (step 5 of Figure 3).” EX1005, Welch, 3.

287. These applications are scaled up or down to meet QoS requirements: Welch’s system “recommends actions (e.g., move a program to a different host or

LAN, shed a program, or replicate a program) to improve QoS.” EX1005, Welch, 2. A POSITA would have understood that when scaling up or down the number of applications (M), while the number of hosts (N) remain largely fixed, this implies that M and N are independent (*i.e.*, can be larger, smaller, or equal to each other).

288. Hosokawa describes a similar distributed system that involves multiple hosts. EX1007, Hosokawa, Abstract. Hosokawa relates to a “cluster controlling system [that] transfers packages which have been operating on one computer to another computer when a fault or failure has occurred by monitoring and controlling the packages in the entire system.” EX1007, Hosokawa, Abstract. Hosokawa’s system includes “a plurality of computers A-N ... included in a cluster.” EX1007, Hosokawa, 6:65-66. “**The respective computers are executing** cluster daemon A-N (102a-102n), and the respective cluster daemons start-up respective **package programs** A1-N2.” EX1007, Hosokawa, 6:66-7:2. Hosokawa’s system further includes situations where there are more applications than hosts (*e.g.*, Fig. 46A), fewer applications than hosts (*e.g.*, Fig. 37A), and the same number of applications and hosts (*e.g.*, Fig. 26A). *See* §VIII.B.1 above.

289. A POSITA would have found it beneficial for Welch’s scalable system to have the number of applications be independent from the number of hosts. That way, as applications are replicated, or shed, host systems do not have to be correspondingly commissioned or decommissioned. And when application demand

is low, the applications can be shed so as to conserve computing resources, and conversely then scale up as demand grows. *See* §V.B.1 above. The number of applications being independent from the number of hosts thus allows Welch to meet its design goals: “Furthermore, resources should be utilized in an efficient manner, and scalability must be provided to address the ever-increasing complexity of scenarios that confront such systems, even though the worst case scenarios of the environment may be unknown.” EX1005, Welch, 1.

290. Such resource management would have been well known to a POSITA. Consider a computer with 10MB of RAM memory. Such a host could run a single application that requires all 10MB of memory. But it could also run an application that requires only 6MB of memory. In that case, the remaining 4MB of memory remain unused. A POSITA would have known that unused resources are wasted and may still consume power even when not in use (*e.g.*, DRAM is volatile and must be constantly refreshed with power signals so as not to corrupt its state or lose coherency). If, say, another application needs to run that consumes 3MB of memory, one alternative would be to place it on an entirely separate host: but that would be wasteful (*e.g.*, bring up a whole new computer for a “small” application). Instead, a more cost/energy efficient way would be to run the 3MB application on the same host running the 6MB application. Both can run easily on modern computers and multi-process operating systems. In fact, that host would still have 1MB left (10-6-

3=1) to run a 3rd application that might fit in 1MB. *See also* §V.B.1 above.

291. Accordingly, the combination of Welch and Hosokawa represents combining prior art elements (Welch and Hosokawa's distributed system) according to known methods (independently scaling applications) to yield predictable results (allowing for scalability to quickly respond to demand for resources). Additionally, the combination is merely applying a known technique (number of applications independent from number of hosts) to a known device (distributed computing systems) ready for improvement to yield predictable results (efficient utilization of resources).

292. A POSITA would have had a reasonable expectation of success in combining the teachings of Hosokawa with those of Welch because both Welch and Hosokawa describe similar distributed computing systems suitable for scalability. *Compare* EX1005, Welch, Title, 1 ("scalability must be provided.") *with* EX1007, Hosokawa, abstract (describing a "cluster controlling system [that] transfers packages which have been operating on one computer to another computer when a fault or failure has occurred by monitoring and controlling the packages in the entire system.").

293. Accordingly, a POSITA would have expected that Hosokawa's independent number of applications to work just as well in Welch's distributed computing system. In addition, independently scaling in distributed systems was

known. *See e.g.*, EX1008, Shah, 3:54-56 (describing a “software system designed to support the development and execution of distributed, scalable, fault resilient telecommunications applications 12.”). *See also* §V.B.1 above.

(b) Management software executing on a host

294. Welch describes using QoS management software (sometimes referred to as middleware) that manages the scaling and moving of applications across multiple hosts. EX1005, Welch, 2-3. As Welch explains, a “core component of the middleware is the resource manager. It is activated when programs die and when time-constrained control paths miss their deadlines. In response to these events, it takes appropriate measures to improve the quality of service delivered to the applications.” EX1005, Welch, 2. For example, “[i]f a path becomes unhealthy (misses its deadline), the QoS Monitors detect such a condition, diagnose the cause of poor health, and suggest an action (such as moving or replicating an application program).” EX1005, Welch, 3. “This decision is made by choosing the host(s) and LAN(s) with the smallest load indices.” EX1005, Welch, 3.

295. Hosokawa describes a similar management software that “mov[es] package programs to other computers when a fault or failure occurs on one computer in the cluster.” EX1007, Hosokawa, 2:23-26. In particular, the “manager 105 communicates with cluster daemons on the respective computers to monitor and control the entire cluster system.” EX1007, Hosokawa, 7:10-12. The manager then

“transfers packages which have been operating on one computer to another computer when a fault or failure has occurred by monitoring and controlling the packages in the entire system.” EX1007, Hosokawa, Abstract.

296. Hosokawa provides an example of a known implementation technique in which the software manager executes on one of the managed hosts. In particular, “[t]he manager 105 is one of the packages installed in the cluster system.” EX1007, Hosokawa, 7:16-17. This provides the benefit of recovering the manager when there is a fault or failure. “The manager 105 is restarted in another computer by a cluster daemon when a fault or failure occurs in the manager, or in the computer operating the manager.” EX1007, Hosokawa, 7:17-20. In an example, the manager is illustrated as being executed on Computer B in annotated Fig. 1, shown below.

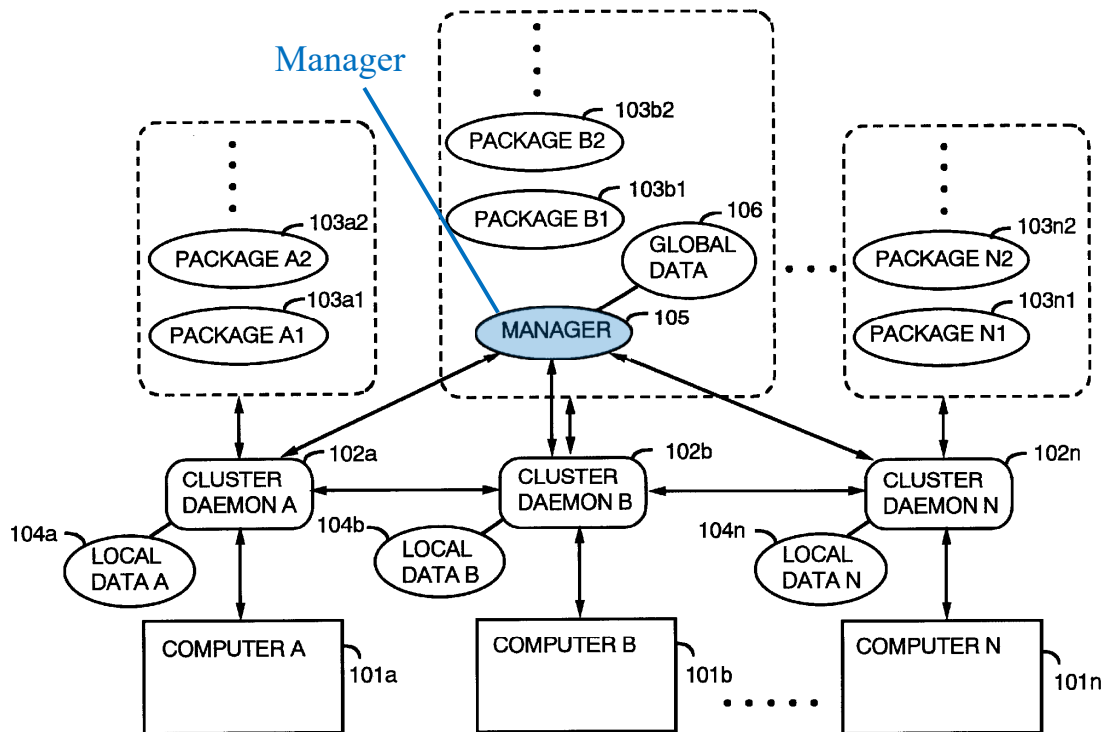


FIG. 1

EX1007, Fig. 26a (annotated).

297. A POSITA would have thus found it similarly beneficial for Welch's QoS management software to be executed one of several managed hosts. That way, if the host running the QoS management software fails, the QoS management software can be restarted on another host, as taught by Hosokawa. EX1007, Hosokawa, 7:17-20. This is consistent with Welch's goals of "QoS and resource management middleware for survivability and scalability services." EX1005, Welch, 4. Moreover, Welch's system is already configured to restart faulty applications on other hosts. See EX1005, Welch, 3 ("The Resource Manger needs to decide on which host(s) and LAN(s) the unhealthy sub-path needs to be replicated

or moved.”). Indeed, the combination is nothing more than the predictable use of prior art elements according to their established functions.

298. Accordingly, the combination of Welch and Hosokawa represents combining prior art elements (Welch and Hosokawa’s distributed software managers) according to known methods (with the software manager executing on one of the managed hosts) to yield predictable results (allowing for survivability of the software manager). Additionally, the combination is merely applying a known technique (Hosokawa’s management software stored on the managed hosts) to a known device (Welch’s distributed computing system) ready for improvement to yield predictable results (survivability of the management software).

299. A POSITA would have had a reasonable expectation of success in combining the teachings of Hosokawa with those of Welch because both Welch and Hosokawa describe similar distributed computing systems designed for survivability. *Compare* EX1005, Welch, 2-3 (“QoS and resource management middleware for survivability and scalability services”) *with* EX1007, Hosokawa, Fig.1, Abstract (“If a fault or failure occurs in the manager or in the computer running the manager, the manager is re-started on another computer by a cluster daemon.”). Accordingly, a POSITA would have expected that Hosokawa’s survivable management software technique to work just as well in Welch’s distributed computing system.

3. Claim 10

(a) [10.0] Software stored on at least one host for converting N networked hosts into a resource managed system instantiating M managed characteristic application computer programs, each managed characteristic application computer program managed by one of the N networked hosts, the software comprising:

300. To the extent the preamble is limiting, Welch and Hosokawa render this limitation obvious.

301. First, for the reasons described in §VIII.A.4(a) above for limitation [10.0] in Ground 1, Welch teaches “software ... for converting N networked hosts into a resource managed system instantiating M managed characteristic application computer programs, each managed characteristic application computer program managed by one of the N networked hosts.”

302. Second, it would have been obvious for Welch’s software to be “stored on at least one host” where that at least one host is one of the managed hosts, as taught by Hosokawa. In particular, Welch explains that its “**software runs on** a heterogeneous network configuration that includes Myrinet, ATM, FDDI, and ethernet, on **multiple heterogeneous host platforms**.” EX1005, Welch, 4. The “multiple heterogeneous host platforms” are the “N networked hosts” managed by the QoS management software.

303. As explained in §VIII.B.1 above, Hosokawa describes a distributed system in which “[t]he manager 105 is one of the packages installed in the cluster

system.” EX1007, Hosokawa, 7:16-17. “If a fault or failure occurs in the manager or in the computer running the manager, the manager is re-started on another computer by a cluster daemon.” EX1007, Hosokawa, Abstract. Hosokawa’s manager is illustrated as being run on one of the computer nodes in annotated Fig. 1 below.

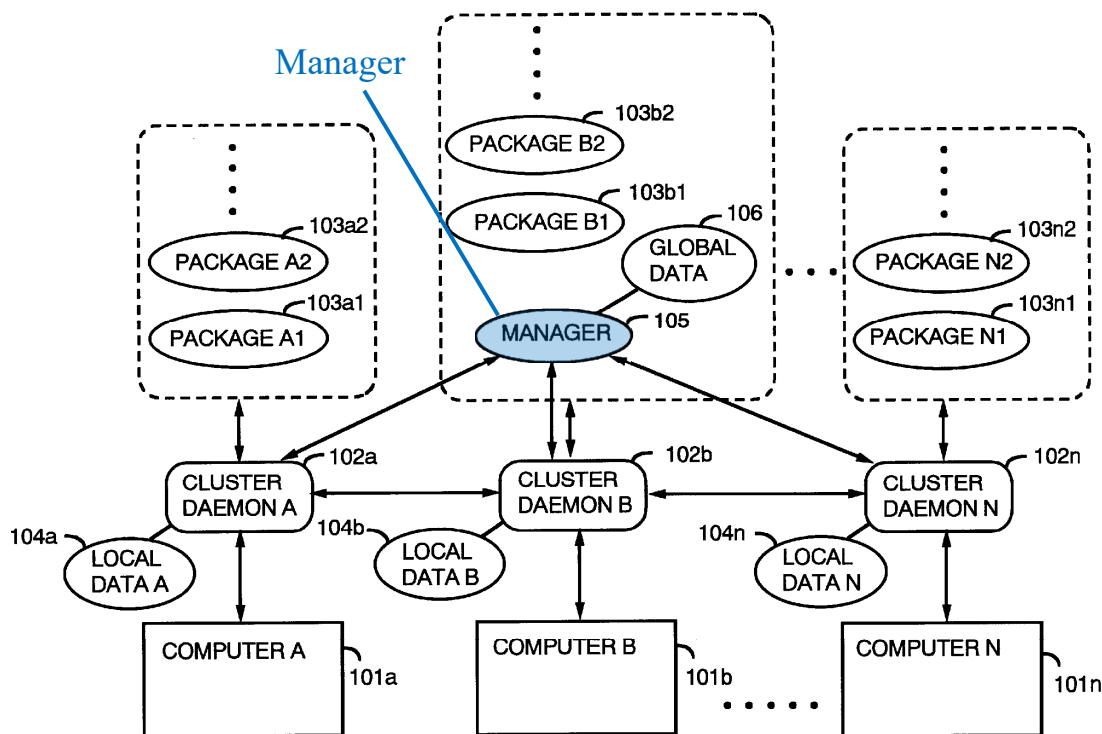


FIG. 1

EX1007, Fig. 26a (annotated).

304. Accordingly, to the extent limitation [10.0] requires that the “at least one host” be one of the “N networked hosts,” it would have been obvious for Welch’s QoS management software to run on one of the managed hosts as taught by Hosokawa. See §VIII.B.2 above.

305. Thus, Welch describes QoS management software for managing real-

time control system applications on distributed hardware to provide startup, replicating, and other actions related to providing for scalability and survivability, which may run on one of the managed hosts as taught by Hosokawa. Welch and Hosokawa thus render obvious “*Software stored on at least one host for converting N networked hosts into a resource managed system instantiating M managed characteristic application computer programs, each managed characteristic application computer program managed by one of the N networked hosts.*”

306. In sum, the combination of Welch and Hosokawa discloses “Software [e.g., QoS management software] stored on at least one host [e.g., one of Welch’s heterogeneous hosts as evidenced by Hosokawa] for converting N networked hosts [e.g., distributed hardware] into a resource managed system [e.g., distributed system] instantiating M managed characteristic application computer programs [e.g., RTCS programs], each managed characteristic application computer program managed by one of the N networked hosts [e.g., distributed hardware].”

(b) Limitations [10.1]-[10.7]

307. See §§VIII.A.4(b)VIII.A.4(h) above for respective limitations [10.1]-[10.7], in Ground 1.

(c) [10.8] where M and N are positive integers and where M may be equal to, greater than, or less than N .

308. Welch and Hosokawa render this limitation obvious.

309. As an initial matter, a POSITA would have recognized that the

language “where M may be equal to, greater than, or less than N” merely states that M and N are variables independent of each other. Additionally, the phrase “positive” is redundant because a negative number of hosts or applications is nonsensical.

310. A POSITA would have found it obvious that the number of hosts N and the number of applications M in Welch are independent variables. As described in §VIII.A.4(i) above for limitation [10.8], in Ground 1, Welch describes distributed hardware that includes a plurality of heterogeneous hosts. Given that Welch describes increasing and decreasing the number of applications on a fixed number of hosts, a POSITA would have understood that in Welch, “*M and N are positive integers and where M may be equal to, greater than, or less than N*” as claimed.

311. Indeed, it would have been obvious for Welch’s applications to be equal to or independent from the number of hosts, as taught by Hosokawa. Hosokawa refers to programs using several interchangeable terms. “In this specification, package program is a general term referring to application programs and to service programs.” EX1007, Hosokawa, 7:3-5.

312. Hosokawa provides an explicit example in which the number of hosts (N) is independent from the number of applications (M) running on those hosts. For example, Hosokawa illustrates an example in which the number of applications (referred to as software packages in Hosokawa) is the same as the number of hosts

(referred to as nodes in Hosokawa). In other words, Hosokawa provides an example in which M is equal to N, as shown in annotated Fig. 26A below.

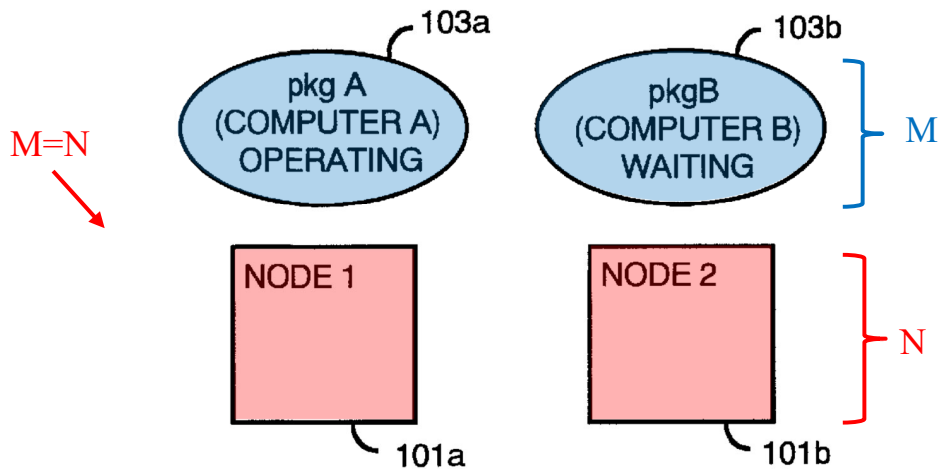


FIG. 26A

EX1007, Fig. 26a (annotated).

FIGS. 26A-26D illustrate one example of the operation according to the ninth embodiment. In FIGS. 26A-26D, the elements having the same numbers as those in FIG. 1 correspond to the same portions of FIG. 1.

First of all, as illustrated in FIG. 26A, a first computer A 101a is executing a first package A 103a, a second computer B 101b is waiting on a second package B 103b. When the computer A 101a stops due to a fault or failure, as illustrated in FIG. 6B, the manager 105 starts to operate in the above-explained manner, and the computer B 101b executes the package B 103b which has been waiting to operate as illustrated in FIG. 26C. At this time, package A 103a can be changed to package B 103b at high speed, which reduces the time necessary for

starting-up, since package B 103b has been waiting. Once computer A101 a recovers from the stop state and starts to operate normally, package A103a, which had been stopped, is started-up on computer A 101a, and assumes the “waiting” state as illustrated in FIG. 26D.

EX1007, Hosokawa, 16:3-31.

313. Additionally, Hosokawa illustrates an example where there are fewer applications M than hosts N ($M < N$). Such example is shown in annotated Fig. 37A below, where $M=2$ and $N=n$ ($n > 2$ and hence $N > 2$).

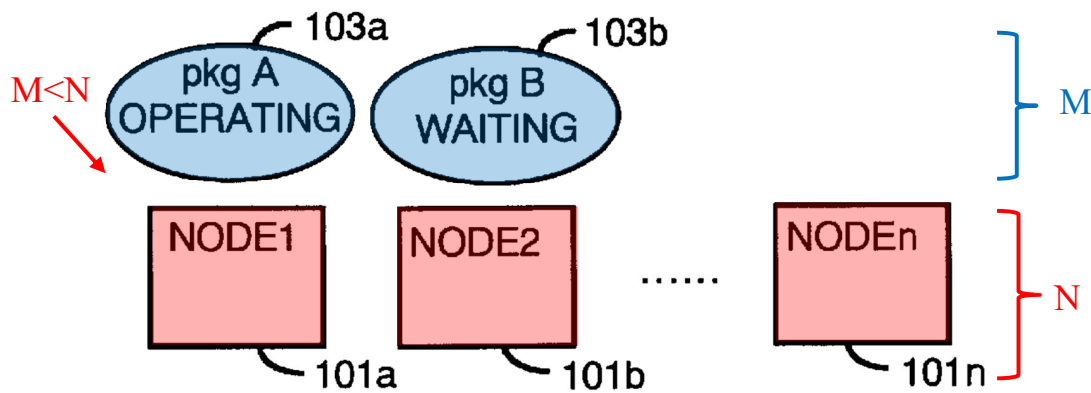


FIG. 37A

EX1007, Fig. 37a (annotated).

FIGS. 37A-37D explain an example of the operation according to the twelfth embodiment. In FIGS. 37A-37D, the elements having the same numbers as those in FIG. 1 correspond to the same portions of FIG. 1.

In the beginning, as shown in FIG. 37A, assuming that a first computer A 101a is executing a first package A103a, and a second computer B

101b is executing a second package B 101b. And assuming that the package A 103a and the package B 103b are different kinds of packages 103.

If the computer A 101 a stops due to a fault or failure in the manner illustrated in FIG. 37B, after the state illustrated in FIG. 34A, the manager 105 starts operating in the manner explained above, then the package. A 103a starts-up to operate on the computer A101n having being vacant until now.

EX1007, Hosokawa, 22:25-39.

314. Finally, Hosokawa further illustrates an example where there are more applications M than hosts N ($M > N$). Such example is shown in annotated Fig. 46A below, where M is shown as at least double that of N (each host in N runs at least two applications M).

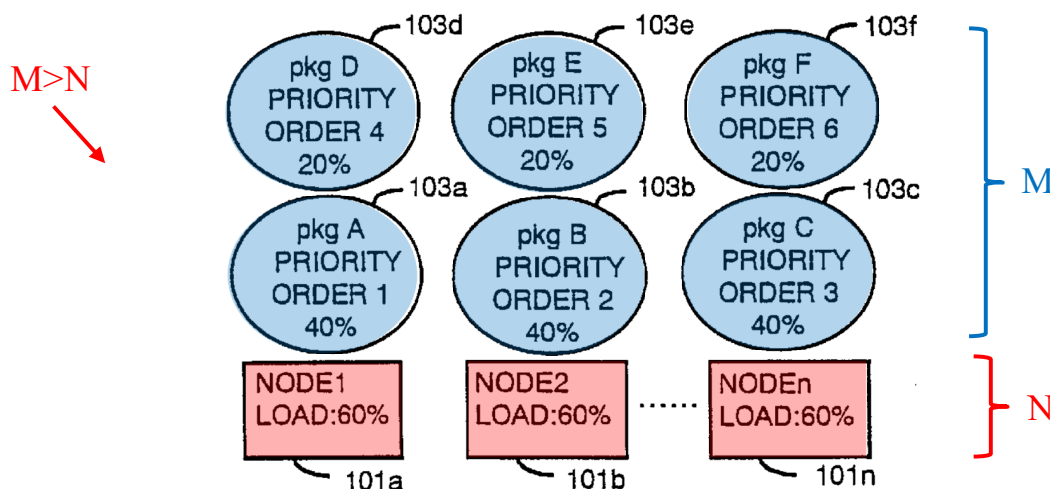


FIG. 46A

EX1007, Fig. 26a (annotated).

FIGS. 46A-46D explain another example of the operation according to the fifteenth embodiment. In FIGS. 46A-46D, the elements having the same numbers as those in FIGS. 45A-46C are the same portions as those in FIGS. 45A-46C.

If the packages A 103a and D 103d on the computer A 101 a stop due to a failure or fault as illustrated in FIG. 46B, after the State shown in FIG. 46A, the manager operates in the above explained manner and starts-up and executes the package. A 103a with the highest priority of all stopped packages on the computer N 101n, as shown in FIG. 46C.

The package F 103f with the lowest priority is stopped at this time.

EX1007, Hosokawa, 28:65-29:10.

315. Accordingly, Hosokawa provides both the general teaching that it was known that in distributed systems, the number of applications can be independent from the number of hosts, and the specific example recited in this limitation [10.8]. It would have been obvious for Welch's distributed system to operate similarly as taught by Hosokawa for the reasons described in §VIII.B.2 above.

316. Thus, because Welch alone and in combination with Hosokawa describe systems in which the number of applications is independent from the number of hosts, Welch and Hosokawa render obvious "*where M and N are positive integers and where M may be equal to, greater than, or less than N*" as claimed.

317. In sum, the combination of Welch and Hosokawa discloses "where M [e.g., number of RTCS applications] and N [e.g., number of hosts] are positive

integers and where M may be equal to, greater than, or less than N [*e.g.*, M and N are independent variables as evidenced by Hosokawa].”

4. Claims 11, 14, 16, and 18

318. See §§VIII.A.5-VIII.A.8 above for claims respective 11, 14, 16, and 18, in Ground 1.

IX. CONCLUSION

319. This declaration and my opinions herein are made to the best of my knowledge and understanding, and based on the material available to me, at the time of signing this declaration. I declare that all statements made herein on my own knowledge are true and that all statements made on information and belief are believed to be true, and further, that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 or Title 18 of the United States Code.

Date: December 22, 2024

A handwritten signature in black ink, appearing to read "Erez Zadok", written over a horizontal line.

Erez Zadok, PhD