(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
6 September 2002 (06.09.2002)

PCT

(10) International Publication Number
WO 02/069604 A2

(51) International Patent Classification[7]: H04L 29/06, 12/26

(21) International Application Number: PCT/US01/45696

(22) International Filing Date:
2 November 2001 (02.11.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/246,335 7 November 2000 (07.11.2000) US
09/797,411 1 March 2001 (01.03.2001) US

(71) Applicant: SURGIENT NETWORKS, INC. [US/US]; 8303 Mopac, Suite C300, Austin, TX 78746 (US).

(72) Inventors: CANION, Rodney, S.; 3204 Lating Stream Lane, West Lake Hills, TX 78746 (US). BAILEY, Brian, W.; 8804B Clearbrook Trail, Austin, TX 78729 (US). GARVENS, Thomas, E.; 5124 Concho Creek Bend, Austin, TX 78735 (US). JOHNSON, Scott, C.; 3612 Galena Hills Loop, Round Rock, TX 78681-1032 (US). RICHTER, Roger, K.; 15248 Faubion Trail, Leander, TX 78641 (US). WANG, Ho; 2317 Waterway Bend, Austin, TX 78728 (US).

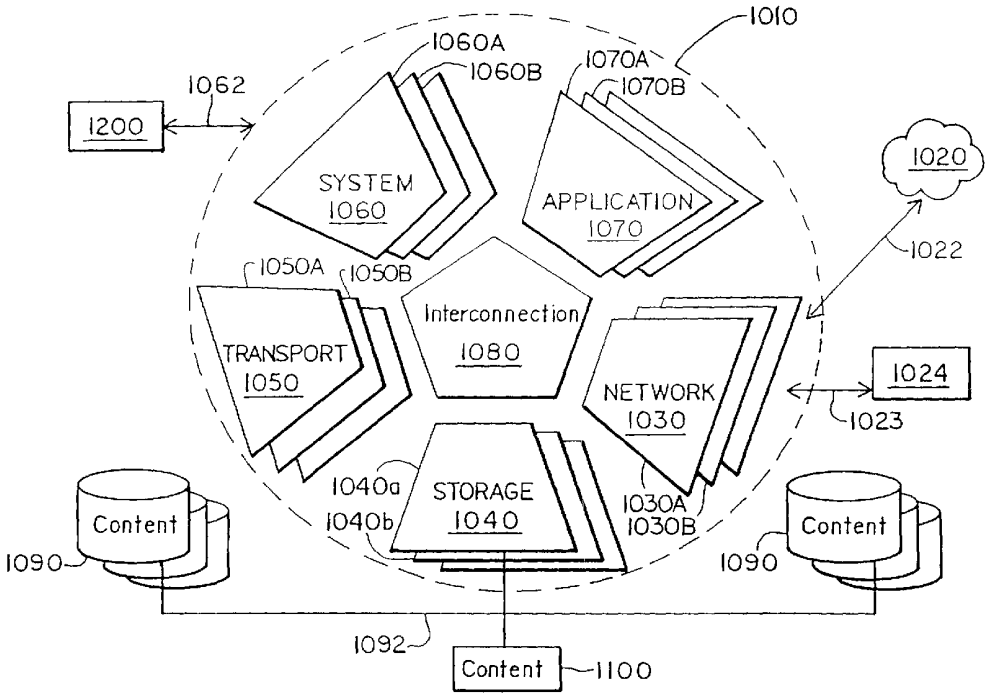
(74) Agent: ENDERS, William, W.; O'keefe, Egan & Peterman, LLP, 1101 Capital of Texas Highway South, Building C, Suite 200, Austin, TX 78746 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF,

[Continued on next page]

(54) Title: NETWORK SECURITY ACCELERATOR



(57) Abstract: A network processing system uses intelligent security hardware is a security accelerator at its front end. The security hardware performs initial processing of incoming data, such as security detection tasks. The security hardware is directly connected to one or more processing units, via a bus or switch fabric, which execute appropriate applications and/or storage programming.

WO 02/069604 A2

CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

NETWORK SECURITY ACCELERATOR

BACKGROUND OF THE INVENTION

The present invention relates generally to network connected computing systems, and more particularly, to security techniques for network connected computing systems.

Network connected computing systems often incorporate a variety of security measures in order to protect against a wide variety of unauthorized intrusions, access, or attacks against the computing system. An important task of a network connected computing system, such as a network endpoint system, may be to permit authorized system access since unauthorized access to the system can result in degraded performance, loss of service to authorized clients, loss of content on the system, etc. Network endpoint systems may include a wide variety of computing devices, including but not limited to, classic general purpose servers, network appliances (specialized servers), content delivery systems, home or laptop computers, clients, any other device that operates as an endpoint network connection, etc.

In computer network parlance, "security" is a catchall term that refers to the task of ensuring that only authorized clients are allowed access to a computing system and of preventing unauthorized intrusions or attacks. The security counter measures implemented with network connected computing systems are often denoted as firewalls. Network endpoint systems may be implemented with a stand alone hardware based firewall located between the endpoint system and the external network to which the endpoint system is coupled. A hardware based firewall often is expensive, inflexible and a performance bottleneck. Alternatively, a software based firewall often has more flexibility but is even slower than hardware based solutions.

There are many different types of security attacks and different types of attacks require different security counter measures. Many software programming techniques have been developed to detect security attacks and to prevent them from having adverse effects on an endpoint computer system. These software techniques are referred to as "security tools" and may be implemented in the hardware or software based firewall.

A common example of a network security tool is one that detects "denial of service" (DoS) attacks. A DoS attack is an attack on a system with the intent of preventing clients from using the attacked server. Generally, a DoS attack is not an attempt to gain unauthorized access to a system; rather, its objective is to prevent legitimate clients from gaining authorized access by overwhelming the system with connection requests.

DoS attacks make use of weaknesses in network protocols, such as the TCP/IP protocol. For example, if a sufficient number of new TCP connection requests are sent to an endpoint system (such as a server or content delivery system) at the same time, the endpoint system will attempt to establish a connection for each request. Eventually, the system's connection table fills up or other resources become depleted, which can result in loss of system functionality.

Many other types of attacks on network connected computing systems are also known. For example in "ping" attacks, a ping request is received with a broadcast destination request resulting in massive amounts of replies to the ping. A "Trojan horse" attack may execute a program on the endpoint computer system and cause an unauthorized transfer of data to the external network. In a "syn" attack, syn requests may cause the TCP/IP stack to overfill. Other attacks may rely on the use of "bogus" source addresses. Security tools often address these types of attacks (and others) and may further include other security functions. Such additional security functions commonly required of a network security system include authentication verification, packet filtering, and access control list (ACL) enforcement.

Detection of attempted security violations often requires the endpoint computing system to examine packets to distinguish real requests from attack based requests. The packet processing may also require comparisons among packets in a series, to detect various attack "signatures". All of this processing requires use of processor and memory resources. For this reason security measures are often implemented as a standalone network device.

SUMMARY OF THE INVENTION

One aspect of the invention is a network processing system connected to a network that carries data in packet format. Intelligent security hardware is placed at the network interface of the endpoint system. The intelligent security hardware (or a security accelerator) is programmed to receive packets from the network and to examine each packet to determine

whether data in the packet represents a potential security violation. One or more processing units are programmed to execute some form of application and/or storage programming, and to thereby respond to requests contained within the packets. An interconnection medium is used to directly connect the security hardware to the processing units.

An advantage of the invention is that the security hardware at the front end of the endpoint eliminates the need for a firewall. Security tools are offloaded from other endpoint resources to the security hardware, so that the other resources can be devoted to server application tasks. The security hardware has the capability of performing "look ahead" processing, thus unburdening the processing units that must perform the basic tasks appropriate for that particular network node.

The security hardware may be implemented with either a network processor or a CPU type general processor or a combination thereof. When a network processor is used, it performs "pass through" type processing that is especially suitable for many types of security algorithms. This type of processing can be more suited for these algorithms than the state-modification intensive and memory-access intensive processing of a CPU type processor.

The security hardware can detect attempted security breaches very quickly. It can take immediate action, such as discarding the packet or notifying the network administrator. Its programming can be easily updated by means of a simple download, making it easy to upgrade the security hardware to detect new types of attacks.

DESCRIPTION OF THE FIGURES

FIG. 1A is a representation of components of a content delivery system according to one embodiment of the disclosed content delivery system.

FIG. 1B is a representation of data flow between modules of a content delivery system of FIGURE 1A according to one embodiment of the disclosed content delivery system.

FIG. 1C is a simplified schematic diagram showing one possible network content delivery system hardware configuration.

FIG. 1D is a simplified schematic diagram showing a network content delivery engine configuration possible with the network content delivery system hardware configuration of FIG. 1C.

FIG. 1E is a simplified schematic diagram showing an alternate network content delivery engine configuration possible with the network content delivery system hardware configuration of FIG. 1C.

FIG. 1F is a simplified schematic diagram showing another alternate network content delivery engine configuration possible with the network content delivery system hardware configuration of FIG. 1C.

FIGS. 1G-1J illustrate exemplary clusters of network content delivery systems.

FIG. 2 is a simplified schematic diagram showing another possible network content delivery system configuration.

FIG. 2A is a simplified schematic diagram showing a network endpoint computing system.

FIG. 2B is a simplified schematic diagram showing a network endpoint computing system.

FIG. 3 is a functional block diagram of an exemplary network processor.

FIG. 4 is a functional block diagram of an exemplary interface between a switch fabric and a processor.

FIGS. 5 - 8 illustrate various systems having a network security accelerator in accordance with the invention.

DETAILED DESCRIPTION

Disclosed herein are systems and methods for operating network connected computing systems. The network connected computing systems disclosed provide a more

efficient use of computing system resources and provide improved performance as compared to traditional network connected computing systems. Network connected computing systems may include network endpoint systems. The systems and methods disclosed herein may be particularly beneficial for use in network endpoint systems. Network endpoint systems may include a wide variety of computing devices, including but not limited to, classic general purpose servers, specialized servers, network appliances, storage area networks or other storage medium, content delivery systems, corporate data centers, application service providers, home or laptop computers, clients, any other device that operates as an endpoint network connection, etc.

Other network connected systems may be considered a network intermediate node system. Such systems are generally connected to some node of a network that may operate in some other fashion than an endpoint. Typical examples include network switches or network routers. Network intermediate node systems may also include any other devices coupled to intermediate nodes of a network.

Further, some devices may be considered both a network intermediate node system and a network endpoint system. Such hybrid systems may perform both endpoint functionality and intermediate node functionality in the same device. For example, a network switch that also performs some endpoint functionality may be considered a hybrid system. As used herein such hybrid devices are considered to be a network endpoint system and are also considered to be a network intermediate node system.

For ease of understanding, the systems and methods disclosed herein are described with regards to an illustrative network connected computing system. In the illustrative example the system is a network endpoint system optimized for a content delivery application. Thus a content delivery system is provided as an illustrative example that demonstrates the structures, methods, advantages and benefits of the network computing system and methods disclosed herein. Content delivery systems (such as systems for serving streaming content, HTTP content, cached content, etc.) generally have intensive input/output demands.

It will be recognized that the hardware and methods discussed below may be incorporated into other hardware or applied to other applications. For example with respect

to hardware, the disclosed system and methods may be utilized in network switches. Such switches may be considered to be intelligent or smart switches with expanded functionality beyond a traditional switch. Referring to the content delivery application described in more detail herein, a network switch may be configured to also deliver at least some content in addition to traditional switching functionality. Thus, though the system may be considered primarily a network switch (or some other network intermediate node device), the system may incorporate the hardware and methods disclosed herein. Likewise a network switch performing applications other than content delivery may utilize the systems and methods disclosed herein. The nomenclature used for devices utilizing the concepts of the present invention may vary. The network switch or router that includes the content delivery system disclosed herein may be called a network content switch or a network content router or the like. Independent of the nomenclature assigned to a device, it will be recognized that the network device may incorporate some or all of the concepts disclosed herein.

The disclosed hardware and methods also may be utilized in storage area networks, network attached storage, channel attached storage systems, disk arrays, tape storage systems, direct storage devices or other storage systems. In this case, a storage system having the traditional storage system functionality may also include additional functionality utilizing the hardware and methods shown herein. Thus, although the system may primarily be considered a storage system, the system may still include the hardware and methods disclosed herein. The disclosed hardware and methods of the present invention also may be utilized in traditional personal computers, portable computers, servers, workstations, mainframe computer systems, or other computer systems. In this case, a computer system having the traditional computer system functionality associated with the particular type of computer system may also include additional functionality utilizing the hardware and methods shown herein. Thus, although the system may primarily be considered to be a particular type of computer system, the system may still include the hardware and methods disclosed herein.

As mentioned above, the benefits of the present invention are not limited to any specific tasks or applications. The content delivery applications described herein are thus illustrative only. Other tasks and applications that may incorporate the principles of the present invention include, but are not limited to, database management systems, application service providers, corporate data centers, modeling and simulation systems, graphics rendering systems, other complex computational analysis systems, etc. Although the

principles of the present invention may be described with respect to a specific application, it will be recognized that many other tasks or applications performed with the hardware and methods.

Disclosed herein are systems and methods for delivery of content to computer-based networks that employ functional multi-processing using a "staged pipeline" content delivery environment to optimize bandwidth utilization and accelerate content delivery while allowing greater determination in the data traffic management. The disclosed systems may employ individual modular processing engines that are optimized for different layers of a software stack. Each individual processing engine may be provided with one or more discrete subsystem modules configured to run on their own optimized platform and/or to function in parallel with one or more other subsystem modules across a high speed distributive interconnect, such as a switch fabric, that allows peer-to-peer communication between individual subsystem modules. The use of discrete subsystem modules that are distributively interconnected in this manner advantageously allows individual resources (*e.g.*, processing resources, memory resources) to be deployed by sharing or reassignment in order to maximize acceleration of content delivery by the content delivery system. The use of a scalable packet-based interconnect, such as a switch fabric, advantageously allows the installation of additional subsystem modules without significant degradation of system performance. Furthermore, policy enhancement/enforcement may be optimized by placing intelligence in each individual modular processing engine.

The network systems disclosed herein may operate as network endpoint systems. Examples of network endpoints include, but are not limited to, servers, content delivery systems, storage systems, application service providers, database management systems, corporate data center servers, etc. A client system is also a network endpoint, and its resources may typically range from those of a general purpose computer to the simpler resources of a network appliance. The various processing units of the network endpoint system may be programmed to achieve the desired type of endpoint.

Some embodiments of the network endpoint systems disclosed herein are network endpoint content delivery systems. The network endpoint content delivery systems may be utilized in replacement of or in conjunction with traditional network servers. A "server" can be any device that delivers content, services, or both. For example, a content delivery server

receives requests for content from remote browser clients via the network, accesses a file system to retrieve the requested content, and delivers the content to the client. As another example, an applications server may be programmed to execute applications software on behalf of a remote client, thereby creating data for use by the client. Various server appliances are being developed and often perform specialized tasks.

As will be described more fully below, the network endpoint system disclosed herein may include the use of network processors. Though network processors conventionally are designed and utilized at intermediate network nodes, the network endpoint system disclosed herein adapts this type of processor for endpoint use.

The network endpoint system disclosed may be construed as a switch based computing system. The system may further be characterized as an asymmetric multi-processor system configured in a staged pipeline manner.

EXEMPLARY SYSTEM OVERVIEW

FIG. 1A is a representation of one embodiment of a content delivery system 1010, for example as may be employed as a network endpoint system in connection with a network 1020. Network 1020 may be any type of computer network suitable for linking computing systems. Content delivery system 1010 may be coupled to one or more networks including, but not limited to, the public internet, a private intranet network (e.g., linking users and hosts such as employees of a corporation or institution), a wide area network (WAN), a local area network (LAN), a wireless network, any other client based network or any other network environment of connected computer systems or online users. Thus, the data provided from the network 1020 may be in any networking protocol. In one embodiment, network 1020 may be the public internet that serves to provide access to content delivery system 1010 by multiple online users that utilize internet web browsers on personal computers operating through an internet service provider. In this case the data is assumed to follow one or more of various Internet Protocols, such as TCP/IP, UDP, HTTP, RTSP, SSL, FTP, etc. However, the same concepts apply to networks using other existing or future protocols, such as IPX, SNMP, NetBios, Ipv6, etc. The concepts may also apply to file protocols such as network file system (NFS) or common internet file system (CIFS) file sharing protocol.

Examples of content that may be delivered by content delivery system 1010 include, but are not limited to, static content (*e.g.*, web pages, MP3 files, HTTP object files, audio stream files, video stream files, *etc.*), dynamic content, *etc.* In this regard, static content may be defined as content available to content delivery system 1010 via attached storage devices and as content that does not generally require any processing before delivery. Dynamic content, on the other hand, may be defined as content that either requires processing before delivery, or resides remotely from content delivery system 1010. As illustrated in FIG. 1A, content sources may include, but are not limited to, one or more storage devices 1090 (magnetic disks, optical disks, tapes, storage area networks (SAN's), *etc.*), other content sources 1100, third party remote content feeds, broadcast sources (live direct audio or video broadcast feeds, *etc.*), delivery of cached content, combinations thereof, *etc.* Broadcast or remote content may be advantageously received through second network connection 1023 and delivered to network 1020 via an accelerated flowpath through content delivery system 1010. As discussed below, second network connection 1023 may be connected to a second network 1024 (as shown). Alternatively, both network connections 1022 and 1023 may be connected to network 1020.

As shown in FIG. 1A, one embodiment of content delivery system 1010 includes multiple system engines 1030, 1040, 1050, 1060, and 1070 communicatively coupled via distributive interconnection 1080. In the exemplary embodiment provided, these system engines operate as content delivery engines. As used herein, "content delivery engine" generally includes any hardware, software or hardware/software combination capable of performing one or more dedicated tasks or sub-tasks associated with the delivery or transmittal of content from one or more content sources to one or more networks. In the embodiment illustrated in FIG. 1A content delivery processing engines (or "processing blades") include network interface processing engine 1030, storage processing engine 1040, network transport / protocol processing engine 1050 (referred to hereafter as a transport processing engine), system management processing engine 1060, and application processing engine 1070. Thus configured, content delivery system 1010 is capable of providing multiple dedicated and independent processing engines that are optimized for networking, storage and application protocols, each of which is substantially self-contained and therefore capable of functioning without consuming resources of the remaining processing engines.

It will be understood with benefit of this disclosure that the particular number and identity of content delivery engines illustrated in FIG. 1A are illustrative only, and that for any given content delivery system 1010 the number and/or identity of content delivery engines may be varied to fit particular needs of a given application or installation. Thus, the number of engines employed in a given content delivery system may be greater or fewer in number than illustrated in FIG. 1A, and/or the selected engines may include other types of content delivery engines and/or may not include all of the engine types illustrated in FIG. 1A. In one embodiment, the content delivery system 1010 may be implemented within a single chassis, such as for example, a 2U chassis.

Content delivery engines 1030, 1040, 1050, 1060 and 1070 are present to independently perform selected sub-tasks associated with content delivery from content sources 1090 and/or 1100, it being understood however that in other embodiments any one or more of such subtasks may be combined and performed by a single engine, or subdivided to be performed by more than one engine. In one embodiment, each of engines 1030, 1040, 1050, 1060 and 1070 may employ one or more independent processor modules (*e.g.*, CPU modules) having independent processor and memory subsystems and suitable for performance of a given function/s, allowing independent operation without interference from other engines or modules. Advantageously, this allows custom selection of particular processor-types based on the particular sub-task each is to perform, and in consideration of factors such as speed or efficiency in performance of a given subtask, cost of individual processor, *etc.* The processors utilized may be any processor suitable for adapting to endpoint processing. Any "PC on a board" type device may be used, such as the x86 and Pentium processors from Intel Corporation, the SPARC processor from Sun Microsystems, Inc., the PowerPC processor from Motorola, Inc. or any other microcontroller or microprocessor. In addition, network processors (discussed in more detail below) may also be utilized. The modular multi-task configuration of content delivery system 1010 allows the number and/or type of content delivery engines and processors to be selected or varied to fit the needs of a particular application.

The configuration of the content delivery system described above provides scalability without having to scale all the resources of a system. Thus, unlike the traditional rack and stack systems, such as server systems in which an entire server may be added just to expand one segment of system resources, the content delivery system allows the particular resources

needed to be the only expanded resources. For example, storage resources may be greatly expanded without having to expand all of the traditional server resources.

DISTRIBUTIVE INTERCONNECT

Still referring to FIG. 1A, distributive interconnection 1080 may be any multi-node I/O interconnection hardware or hardware/software system suitable for distributing functionality by selectively interconnecting two or more content delivery engines of a content delivery system including, but not limited to, high speed interchange systems such as a switch fabric or bus architecture. Examples of switch fabric architectures include cross-bar switch fabrics, Ethernet switch fabrics, ATM switch fabrics, etc. Examples of bus architectures include PCI, PCI-X, S-Bus, Microchannel, VME, etc. Generally, for purposes of this description, a "bus" is any system bus that carries data in a manner that is visible to all nodes on the bus. Generally, some sort of bus arbitration scheme is implemented and data may be carried in parallel, as n-bit words. As distinguished from a bus, a switch fabric establishes independent paths from node to node and data is specifically addressed to a particular node on the switch fabric. Other nodes do not see the data nor are they blocked from creating their own paths. The result is a simultaneous guaranteed bit rate in each direction for each of the switch fabric's ports.

The use of a distributed interconnect 1080 to connect the various processing engines in lieu of the network connections used with the switches of conventional multi-server endpoints is beneficial for several reasons. As compared to network connections, the distributed interconnect 1080 is less error prone, allows more deterministic content delivery, and provides higher bandwidth connections to the various processing engines. The distributed interconnect 1080 also has greatly improved data integrity and throughput rates as compared to network connections.

Use of the distributed interconnect 1080 allows latency between content delivery engines to be short, finite and follow a known path. Known maximum latency specifications are typically associated with the various bus architectures listed above. Thus, when the employed interconnect medium is a bus, latencies fall within a known range. In the case of a switch fabric, latencies are fixed. Further, the connections are "direct", rather than by some undetermined path. In general, the use of the distributed interconnect 1080 rather than

network connections, permits the switching and interconnect capacities of the content delivery system 1010 to be predictable and consistent.

One example interconnection system suitable for use as distributive interconnection 1080 is an 8/16 port 28.4 Gbps high speed PRIZMA-E non-blocking switch fabric switch available from IBM. It will be understood that other switch fabric configurations having greater or lesser numbers of ports, throughput, and capacity are also possible. Among the advantages offered by such a switch fabric interconnection in comparison to shared-bus interface interconnection technology are throughput, scalability and fast and efficient communication between individual discrete content delivery engines of content delivery system 1010. In the embodiment of FIG. 1A, distributive interconnection 1080 facilitates parallel and independent operation of each engine in its own optimized environment without bandwidth interference from other engines, while at the same time providing peer-to-peer communication between the engines on an as-needed basis (*e.g.*, allowing direct communication between any two content delivery engines 1030, 1040, 1050, 1060 and 1070). Moreover, the distributed interconnect may directly transfer inter-processor communications between the various engines of the system. Thus, communication, command and control information may be provided between the various peers via the distributed interconnect. In addition, communication from one peer to multiple peers may be implemented through a broadcast communication which is provided from one peer to all peers coupled to the interconnect. The interface for each peer may be standardized, thus providing ease of design and allowing for system scaling by providing standardized ports for adding additional peers.

NETWORK INTERFACE PROCESSING ENGINE

As illustrated in FIG. 1A, network interface processing engine 1030 interfaces with network 1020 by receiving and processing requests for content and delivering requested content to network 1020. Network interface processing engine 1030 may be any hardware or hardware/software subsystem suitable for connections utilizing TCP (Transmission Control Protocol) IP (Internet Protocol), UDP (User Datagram Protocol), RTP (Real-Time Transport Protocol), Internet Protocol (IP), Wireless Application Protocol (WAP) as well as other networking protocols. Thus the network interface processing engine 1030 may be suitable for handling queue management, buffer management, TCP connect sequence, checksum, IP address lookup, internal load balancing, packet switching, *etc.* Thus, network interface processing engine 1030 may be employed as illustrated to process or terminate one or more

layers of the network protocol stack and to perform look-up intensive operations, offloading these tasks from other content delivery processing engines of content delivery system 1010. Network interface processing engine 1030 may also be employed to load balance among other content delivery processing engines of content delivery system 1010. Both of these features serve to accelerate content delivery, and are enhanced by placement of distributive interchange and protocol termination processing functions on the same board. Examples of other functions that may be performed by network interface processing engine 1030 include, but are not limited to, security processing.

With regard to the network protocol stack, the stack in traditional systems may often be rather large. Processing the entire stack for every request across the distributed interconnect may significantly impact performance. As described herein, the protocol stack has been segmented or "split" between the network interface engine and the transport processing engine. An abbreviated version of the protocol stack is then provided across the interconnect. By utilizing this functionally split version of the protocol stack, increased bandwidth may be obtained. In this manner the communication and data flow through the content delivery system 1010 may be accelerated. The use of a distributed interconnect (for example a switch fabric) further enhances this acceleration as compared to traditional bus interconnects.

The network interface processing engine 1030 may be coupled to the network 1020 through a Gigabit (Gb) Ethernet fiber front end interface 1022. One or more additional Gb Ethernet interfaces 1023 may optionally be provided, for example, to form a second interface with network 1020, or to form an interface with a second network or application 1024 as shown (*e.g.*, to form an interface with one or more server/s for delivery of web cache content, *etc.*). Regardless of whether the network connection is via Ethernet, or some other means, the network connection could be of any type, with other examples being ATM, SONET, or wireless. The physical medium between the network and the network processor may be copper, optical fiber, wireless, etc.

In one embodiment, network interface processing engine 1030 may utilize a network processor, although it will be understood that in other embodiments a network processor may be supplemented with or replaced by a general purpose processor or an embedded microcontroller. The network processor may be one of the various types of specialized

processors that have been designed and marketed to switch network traffic at intermediate nodes. Consistent with this conventional application, these processors are designed to process high speed streams of network packets. In conventional operation, a network processor receives a packet from a port, verifies fields in the packet header, and decides on an outgoing port to which it forwards the packet. The processing of a network processor may be considered as "pass through" processing, as compared to the intensive state modification processing performed by general purpose processors. A typical network processor has a number of processing elements, some operating in parallel and some in pipeline. Often a characteristic of a network processor is that it may hide memory access latency needed to perform lookups and modifications of packet header fields. A network processor may also have one or more network interface controllers, such as a gigabit Ethernet controller, and are generally capable of handling data rates at "wire speeds".

Examples of network processors include the C-Port processor manufactured by Motorola, Inc., the IXP1200 processor manufactured by Intel Corporation, the Prism processor manufactured by SiTera Inc., and others manufactured by MMC Networks, Inc. and Agere, Inc. These processors are programmable, usually with a RISC or augmented RISC instruction set, and are typically fabricated on a single chip.

The processing cores of a network processor are typically accompanied by special purpose cores that perform specific tasks, such as fabric interfacing, table lookup, queue management, and buffer management. Network processors typically have their memory management optimized for data movement, and have multiple I/O and memory buses. The programming capability of network processors permit them to be programmed for a variety of tasks, such as load balancing, network protocol processing, network security policies, and QoS/CoS support. These tasks can be tasks that would otherwise be performed by another processor. For example, TCP/IP processing may be performed by a network processor at the front end of an endpoint system. Another type of processing that could be offloaded is execution of network security policies or protocols. A network processor could also be used for load balancing. Network processors used in this manner can be referred to as "network accelerators" because their front end "look ahead" processing can vastly increase network response speeds. Network processors perform look ahead processing by operating at the front end of the network endpoint to process network packets in order to reduce the workload placed upon the remaining endpoint resources. Various uses of network accelerators are

described in the following concurrently filed U.S. patent applications: no. 09/797,412entitled "Network Transport Accelerator," by Bailey et. al; no. 09/797,507 entitled "Single Chassis Network Endpoint System With Network Processor For Load Balancing," by Richter et. al; the disclosures of which are all incorporated herein by reference. When utilizing network processors in an endpoint environment it may be advantageous to utilize techniques for order serialization of information, such as for example, as disclosed in concurrently filed U.S. patent application no. 09/797,197 entitled "Methods and Systems For The Order Serialization Of Information In A Network Processing Environment," by Richter et. al, the disclosure of which is incorporated herein by reference.

FIG. 3 illustrates one possible general configuration of a network processor. As illustrated, a set of traffic processors 21 operate in parallel to handle transmission and receipt of network traffic. These processors may be general purpose microprocessors or state machines. Various core processors 22 - 24 handle special tasks. For example, the core processors 22 - 24 may handle lookups, checksums, and buffer management. A set of serial data processors 25 provide Layer 1 network support. Interface 26 provides the physical interface to the network 1020. A general purpose bus interface 27 is used for downloading code and configuration tasks. A specialized interface 28 may be specially programmed to optimize the path between network processor 12 and distributed interconnection 1080.

As mentioned above, the network processors utilized in the content delivery system 1010 are utilized for endpoint use, rather than conventional use at intermediate network nodes. In one embodiment, network interface processing engine 1030 may utilize a MOTOROLA C-Port C-5 network processor capable of handling two Gb Ethernet interfaces at wire speed, and optimized for cell and packet processing. This network processor may contain sixteen 200 MHz MIPS processors for cell/packet switching and thirty-two serial processing engines for bit/byte processing, checksum generation/verification, etc. Further processing capability may be provided by five co-processors that perform the following network specific tasks: supervisor/executive, switch fabric interface, optimized table lookup, queue management, and buffer management. The network processor may be coupled to the network 1020 by using a VITESSE GbE SERDES (serializer-deserializer) device (for example the VSC7123) and an SFP (small form factor pluggable) optical transceiver for LC fiber connection.

TRANSPORT / PROTOCOL PROCESSING ENGINE

Referring again to FIG. 1A, transport processing engine 1050 may be provided for performing network transport protocol sub-tasks, such as processing content requests received from network interface engine 1030. Although named a "transport" engine for discussion purposes, it will be recognized that the engine 1050 performs transport and protocol processing and the term transport processing engine is not meant to limit the functionality of the engine. In this regard transport processing engine 1050 may be any hardware or hardware/software subsystem suitable for TCP/UDP processing, other protocol processing, transport processing, *etc.* In one embodiment transport engine 1050 may be a dedicated TCP/UDP processing module based on an INTEL PENTIUM III or MOTOROLA POWERPC 7450 based processor running the Thread-X RTOS environment with protocol stack based on TCP/IP technology.

As compared to traditional server type computing systems, the transport processing engine 1050 may off-load other tasks that traditionally a main CPU may perform. For example, the performance of server CPUs significantly decreases when a large amount of network connections are made merely because the server CPU regularly checks each connection for time outs. The transport processing engine 1050 may perform time out checks for each network connection, session management, data reordering and retransmission, data queueing and flow control, packet header generation, etc. off-loading these tasks from the application processing engine or the network interface processing engine. The transport processing engine 1050 may also handle error checking, likewise freeing up the resources of other processing engines.

NETWORK INTERFACE / TRANSPORT SPLIT PROTOCOL

The embodiment of FIG. 1A contemplates that the protocol processing is shared between the transport processing engine 1050 and the network interface engine 1030. This sharing technique may be called "split protocol stack" processing. The division of tasks may be such that higher tasks in the protocol stack are assigned to the transport processor engine. For example, network interface engine 1030 may processes all or some of the TCP/IP protocol stack as well as all protocols lower on the network protocol stack. Another approach could be to assign state modification intensive tasks to the transport processing engine.

In one embodiment related to a content delivery system that receives packets, the network interface engine performs the MAC header identification and verification, IP header identification and verification, IP header checksum validation, TCP and UDP header identification and validation, and TCP or UDP checksum validation. It also may perform the lookup to determine the TCP connection or UDP socket (protocol session identifier) to which a received packet belongs. Thus, the network interface engine verifies packet lengths, checksums, and validity. For transmission of packets, the network interface engine performs TCP or UDP checksum generation, IP header generation, and MAC header generation, IP checksum generation, MAC FCS/CRC generation, etc.

Tasks such as those described above can all be performed rapidly by the parallel and pipeline processors within a network processor. The "fly by" processing style of a network processor permits it to look at each byte of a packet as it passes through, using registers and other alternatives to memory access. The network processor's "stateless forwarding" operation is best suited for tasks not involving complex calculations that require rapid updating of state information.

An appropriate internal protocol may be provided for exchanging information between the network interface engine 1030 and the transport engine 1050 when setting up or terminating a TCP and/or UDP connections and to transfer packets between the two engines. For example, where the distributive interconnection medium is a switch fabric, the internal protocol may be implemented as a set of messages exchanged across the switch fabric. These messages indicate the arrival of new inbound or outbound connections and contain inbound or outbound packets on existing connections, along with identifiers or tags for those connections. The internal protocol may also be used to transfer identifiers or tags between the transport engine 1050 and the application processing engine 1070 and/or the storage processing engine 1040. These identifiers or tags may be used to reduce or strip or accelerate a portion of the protocol stack.

For example, with a TCP/IP connection, the network interface engine 1030 may receive a request for a new connection. The header information associated with the initial request may be provided to the transport processing engine 1050 for processing. That result of this processing may be stored in the resources of the transport processing engine 1050 as state and management information for that particular network session. The transport

processing engine 1050 then informs the network interface engine 1030 as to the location of these results. Subsequent packets related to that connection that are processed by the network interface engine 1030 may have some of the header information stripped and replaced with an identifier or tag that is provided to the transport processing engine 1050. The identifier or tag may be a pointer, index or any other mechanism that provides for the identification of the location in the transport processing engine of the previously setup state and management information (or the corresponding network session). In this manner, the transport processing engine 1050 does not have to process the header information of every packet of a connection. Rather, the transport interface engine merely receives a contextually meaningful identifier or tag that identifies the previous processing results for that connection.

In one embodiment, the data link, network, transport and session layers (layers 2-5) of a packet may be replaced by identifier or tag information. For packets related to an established connection the transport processing engine does not have to perform intensive processing with regard to these layers such as hashing, scanning, look up, etc. operations. Rather, these layers have already been converted (or processed) once in the transport processing engine and the transport processing engine just receives the identifier or tag provided from the network interface engine that identifies the location of the conversion results.

In this manner an identifier or tag is provided for each packet of an established connection so that the more complex data computations of converting header information may be replaced with a more simplistic analysis of an identifier or tag. The delivery of content is thereby accelerated, as the time for packet processing and the amount of system resources for packet processing are both reduced. The functionality of network processors, which provide efficient parallel processing of packet headers, is well suited for enabling the acceleration described herein. In addition, acceleration is further provided as the physical size of the packets provided across the distributed interconnect may be reduced.

Though described herein with reference to messaging between the network interface engine and the transport processing engine, the use of identifiers or tags may be utilized amongst all the engines in the modular pipelined processing described herein. Thus, one engine may replace packet or data information with contextually meaningful information that may require less processing by the next engine in the data and communication flow path. In

addition, these techniques may be utilized for a wide variety of protocols and layers, not just the exemplary embodiments provided herein.

With the above-described tasks being performed by the network interface engine, the transport engine may perform TCP sequence number processing, acknowledgement and retransmission, segmentation and reassembly, and flow control tasks. These tasks generally call for storing and modifying connection state information on each TCP and UDP connection, and therefore are considered more appropriate for the processing capabilities of general purpose processors.

As will be discussed with references to alternative embodiments (such as FIGS. 2 and 2A), the transport engine 1050 and the network interface engine 1030 may be combined into a single engine. Such a combination may be advantageous as communication across the switch fabric is not necessary for protocol processing. However, limitations of many commercially available network processors make the split protocol stack processing described above desirable.

APPLICATION PROCESSING ENGINE

Application processing engine 1070 may be provided in content delivery system 1010 for application processing, and may be, for example, any hardware or hardware/software subsystem suitable for session layer protocol processing (*e.g.,* HTTP, RTSP streaming, *etc.*) of content requests received from network transport processing engine 1050. In one embodiment application processing engine 1070 may be a dedicated application processing module based on an INTEL PENTIUM III processor running, for example, on standard x86 OS systems (*e.g.,* Linux, Windows NT, FreeBSD, *etc.*). Application processing engine 1070 may be utilized for dedicated application-only processing by virtue of the off-loading of all network protocol and storage processing elsewhere in content delivery system 1010. In one embodiment, processor programming for application processing engine 1070 may be generally similar to that of a conventional server, but without the tasks off-loaded to network interface processing engine 1030, storage processing engine 1040, and transport processing engine 1050.

STORAGE MANAGEMENT ENGINE

Storage management engine 1040 may be any hardware or hardware/software subsystem suitable for effecting delivery of requested content from content sources (for example content sources 1090 and/or 1100) in response to processed requests received from application processing engine 1070. It will also be understood that in various embodiments a storage management engine 1040 may be employed with content sources other than disk drives (*e.g.*, solid state storage, the storage systems described above, or any other media suitable for storage of data) and may be programmed to request and receive data from these other types of storage.

In one embodiment, processor programming for storage management engine 1040 may be optimized for data retrieval using techniques such as caching, and may include and maintain a disk cache to reduce the relatively long time often required to retrieve data from content sources, such as disk drives. Requests received by storage management engine 1040 from application processing engine 1070 may contain information on how requested data is to be formatted and its destination, with this information being comprehensible to transport processing engine 1050 and/or network interface processing engine 1030. The storage management engine 1040 may utilize a disk cache to reduce the relatively long time it may take to retrieve data stored in a storage medium such as disk drives. Upon receiving a request, storage management engine 1040 may be programmed to first determine whether the requested data is cached, and then to send a request for data to the appropriate content source 1090 or 1100. Such a request may be in the form of a conventional read request. The designated content source 1090 or 1100 responds by sending the requested content to storage management engine 1040, which in turn sends the content to transport processing engine 1050 for forwarding to network interface processing engine 1030.

Based on the data contained in the request received from application processing engine 1070, storage processing engine 1040 sends the requested content in proper format with the proper destination data included. Direct communication between storage processing engine 1040 and transport processing engine 1050 enables application processing engine 1070 to be bypassed with the requested content. Storage processing engine 1040 may also be configured to write data to content sources 1090 and/or 1100 (*e.g.*, for storage of live or broadcast streaming content).

In one embodiment storage management engine 1040 may be a dedicated block-level cache processor capable of block level cache processing in support of thousands of concurrent multiple readers, and direct block data switching to network interface engine 1030. In this regard storage management engine 1040 may utilize a POWER PC 7450 processor in conjunction with ECC memory and a LSI SYMFC929 dual 2GBaud fibre channel controller for fibre channel interconnect to content sources 1090 and/or 1100 via dual fibre channel arbitrated loop 1092. It will be recognized, however, that other forms of interconnection to storage sources suitable for retrieving content are also possible. Storage management engine 1040 may include hardware and/or software for running the Fibre Channel (FC) protocol, the SCSI (Small Computer Systems Interface) protocol, iSCSI protocol as well as other storage networking protocols.

Storage management engine 1040 may employ any suitable method for caching data, including simple computational caching algorithms such as random removal (RR), first-in first-out (FIFO), predictive read-ahead, over buffering, etc. algorithms. Other suitable caching algorithms include those that consider one or more factors in the manipulation of content stored within the cache memory, or which employ multi-level ordering, key based ordering or function based calculation for replacement. In one embodiment, storage management engine may implement a layered multiple LRU (LMLRU) algorithm that uses an integrated block/buffer management structure including at least two layers of a configurable number of multiple LRU queues and a two-dimensional positioning algorithm for data blocks in the memory to reflect the relative priorities of a data block in the memory in terms of both recency and frequency. Such a caching algorithm is described in further detail in concurrently filed U.S. patent application no. 09/797,198 entitled "Systems and Methods for Management of Memory" by Qiu et. al, the disclosure of which is incorporated herein by reference.

For increasing delivery efficiency of continuous content, such as streaming multimedia content, storage management engine 1040 may employ caching algorithms that consider the dynamic characteristics of continuous content. Suitable examples include, but are not limited to, interval caching algorithms. In one embodiment, improved caching performance of continuous content may be achieved using an LMLRU caching algorithm that weighs ongoing viewer cache value versus the dynamic time-size cost of maintaining particular content in cache memory. Such a caching algorithm is described in further detail in

concurrently filed U.S. patent application no. 09/797,201 entitled "Systems and Methods for Management of Memory in Information Delivery Environments" by Qiu et. al, the disclosure of which is incorporated herein by reference.

SYSTEM MANAGEMENT ENGINE

System management (or host) engine 1060 may be present to perform system management functions related to the operation of content delivery system 1010. Examples of system management functions include, but are not limited to, content provisioning/updates, comprehensive statistical data gathering and logging for sub-system engines, collection of shared user bandwidth utilization and content utilization data that may be input into billing and accounting systems, "on the fly" ad insertion into delivered content, customer programmable sub-system level quality of service ("QoS") parameters, remote management (*e.g.*, SNMP, web-based, CLI), health monitoring, clustering controls, remote/local disaster recovery functions, predictive performance and capacity planning, *etc.* In one embodiment, content delivery bandwidth utilization by individual content suppliers or users (*e.g.*, individual supplier/user usage of distributive interchange and/or content delivery engines) may be tracked and logged by system management engine 1060, enabling an operator of the content delivery system 1010 to charge each content supplier or user on the basis of content volume delivered.

System management engine 1060 may be any hardware or hardware/software subsystem suitable for performance of one or more such system management engines and in one embodiment may be a dedicated application processing module based, for example, on an INTEL PENTIUM III processor running an x86 OS. Because system management engine 1060 is provided as a discrete modular engine, it may be employed to perform system management functions from within content delivery system 1010 without adversely affecting the performance of the system. Furthermore, the system management engine 1060 may maintain information on processing engine assignment and content delivery paths for various content delivery applications, substantially eliminating the need for an individual processing engine to have intimate knowledge of the hardware it intends to employ.

Under manual or scheduled direction by a user, system management processing engine 1060 may retrieve content from the network 1020 or from one or more external servers on a second network 1024 (*e.g.*, LAN) using, for example, network file system (NFS)

or common internet file system (CIFS) file sharing protocol. Once content is retrieved, the content delivery system may advantageously maintain an independent copy of the original content, and therefore is free to employ any file system structure that is beneficial, and need not understand low level disk formats of a large number of file systems.

Management interface 1062 may be provided for interconnecting system management engine 1060 with a network 1200 (*e.g.*, LAN), or connecting content delivery system 1010 to other network appliances such as other content delivery systems 1010, servers, computers, *etc.* Management interface 1062 may be by any suitable network interface, such as 10/100 Ethernet, and may support communications such as management and origin traffic. Provision for one or more terminal management interfaces (not shown) for may also be provided, such as by RS-232 port, *etc.* The management interface may be utilized as a secure port to provide system management and control information to the content delivery system 1010. For example, tasks which may be accomplished through the management interface 1062 include reconfiguration of the allocation of system hardware (as discussed below with reference to FIGS. 1C-1F), programming the application processing engine, diagnostic testing, and any other management or control tasks. Though generally content is not envisioned being provided through the management interface, the identification of or location of files or systems containing content may be received through the management interface 1062 so that the content delivery system may access the content through the other higher bandwidth interfaces.

MANAGEMENT PERFORMED BY THE NETWORK INTEFACE

Some of the system management functionality may also be performed directly within the network interface processing engine 1030. In this case some system policies and filters may be executed by the network interface engine 1030 in real-time at wirespeed. These polices and filters may manage some traffic / bandwidth management criteria and various service level guarantee policies. Examples of such system management functionality of are described below. It will be recognized that these functions may be performed by the system management engine 1060, the network interface engine 1030, or a combination thereof.

For example, a content delivery system may contain data for two web sites. An operator of the content delivery system may guarantee one web site ("the higher quality site") higher performance or bandwidth than the other web site ("the lower quality site"),

presumably in exchange for increased compensation from the higher quality site. The network interface processing engine 1030 may be utilized to determine if the bandwidth limits for the lower quality site have been exceeded and reject additional data requests related to the lower quality site. Alternatively, requests related to the lower quality site may be rejected to ensure the guaranteed performance of the higher quality site is achieved. In this manner the requests may be rejected immediately at the interface to the external network and additional resources of the content delivery system need not be utilized. In another example, storage service providers may use the content delivery system to charge content providers based on system bandwidth of downloads (as opposed to the traditional storage area based fees). For billing purposes, the network interface engine may monitor the bandwidth use related to a content provider. The network interface engine may also reject additional requests related to content from a content provider whose bandwidth limits have been exceeded. Again, in this manner the requests may be rejected immediately at the interface to the external network and additional resources of the content delivery system need not be utilized.

Additional system management functionality, such as quality of service (QoS) functionality, also may be performed by the network interface engine. A request from the external network to the content delivery system may seek a specific file and also may contain Quality of Service (QoS) parameters. In one example, the QoS parameter may indicate the priority of service that a client on the external network is to receive. The network interface engine may recognize the QoS data and the data may then be utilized when managing the data and communication flow through the content delivery system. The request may be transferred to the storage management engine to access this file via a read queue, e.g., [Destination IP][Filename][File Type (CoS)][Transport Priorities (QoS)]. All file read requests may be stored in a read queue. Based on CoS/QoS policy parameters as well as buffer status within the storage management engine (empty, full, near empty, block seq#, etc), the storage management engine may prioritize which blocks of which files to access from the disk next, and transfer this data into the buffer memory location that has been assigned to be transmitted to a specific IP address. Thus based upon QoS data in the request provided to the content delivery system, the data and communication traffic through the system may be prioritized. The QoS and other policy priorities may be applied to both incoming and outgoing traffic flow. Therefore a request having a higher QoS priority may

be received after a lower order priority request, yet the higher priority request may be served data before the lower priority request.

The network interface engine may also be used to filter requests that are not supported by the content delivery system. For example, if a content delivery system is configured only to accept HTTP requests, then other requests such as FTP, telnet, etc. may be rejected or filtered. This filtering may be applied directly at the network interface engine, for example by programming a network processor with the appropriate system policies. Limiting undesirable traffic directly at the network interface offloads such functions from the other processing modules and improves system performance by limiting the consumption of system resources by the undesirable traffic. It will be recognized that the filtering example described herein is merely exemplary and many other filter criteria or policies may be provided.

MULTI-PROCESSOR MODULE DESIGN

As illustrated in FIG. 1A, any given processing engine of content delivery system 1010 may be optionally provided with multiple processing modules so as to enable parallel or redundant processing of data and/or communications. For example, two or more individual dedicated TCP/UDP processing modules 1050a and 1050b may be provided for transport processing engine 1050, two or more individual application processing modules 1070a and 1070b may be provided for network application processing engine 1070, two or more individual network interface processing modules 1030a and 1030b may be provided for network interface processing engine 1030 and two or more individual storage management processing modules 1040a and 1040b may be provided for storage management processing engine 1040. Using such a configuration, a first content request may be processed between a first TCP/UDP processing module and a first application processing module via a first switch fabric path, at the same time a second content request is processed between a second TCP/UDP processing module and a second application processing module via a second switch fabric path. Such parallel processing capability may be employed to accelerate content delivery.

Alternatively, or in combination with parallel processing capability, a first TCP/UDP processing module 1050a may be backed-up by a second TCP/UDP processing module 1050b that acts as an automatic failover spare to the first module 1050a. In those embodiments employing multiple-port switch fabrics, various combinations of multiple

modules may be selected for use as desired on an individual system-need basis (*e.g.*, as may be dictated by module failures and/or by anticipated or actual bottlenecks), limited only by the number of available ports in the fabric. This feature offers great flexibility in the operation of individual engines and discrete processing modules of a content delivery system, which may be translated into increased content delivery acceleration and reduction or substantial elimination of adverse effects resulting from system component failures.

In yet other embodiments, the processing modules may be specialized to specific applications, for example, for processing and delivering HTTP content, processing and delivering RTSP content, or other applications. For example, in such an embodiment an application processing module 1070a and storage processing module 1040a may be specially programmed for processing a first type of request received from a network. In the same system, application processing module 1070b and storage processing module 1040b may be specially programmed to handle a second type of request different from the first type. Routing of requests to the appropriate respective application and/or storage modules may be accomplished using a distributive interconnect and may be controlled by transport and/or interface processing modules as requests are received and processed by these modules using policies set by the system management engine.

Further, by employing processing modules capable of performing the function of more than one engine in a content delivery system, the assigned functionality of a given module may be changed on an as-needed basis, either manually or automatically by the system management engine upon the occurrence of given parameters or conditions. This feature may be achieved, for example, by using similar hardware modules for different content delivery engines (*e.g.*, by employing PENTIUM III based processors for both network transport processing modules and for application processing modules), or by using different hardware modules capable of performing the same task as another module through software programmability (*e.g.*, by employing a POWER PC processor based module for storage management modules that are also capable of functioning as network transport modules). In this regard, a content delivery system may be configured so that such functionality reassignments may occur during system operation, at system boot-up or in both cases. Such reassignments may be effected, for example, using software so that in a given content delivery system every content delivery engine (or at a lower level, every discrete content delivery processing module) is potentially dynamically reconfigurable using software

commands. Benefits of engine or module reassignment include maximizing use of hardware resources to deliver content while minimizing the need to add expensive hardware to a content delivery system.

Thus, the system disclosed herein allows various levels of load balancing to satisfy a work request. At a system hardware level, the functionality of the hardware may be assigned in a manner that optimizes the system performance for a given load. At the processing engine level, loads may be balanced between the multiple processing modules of a given processing engine to further optimize the system performance.

CLUSTERS OF SYSTEMS

The systems described herein may also be clustered together in groups of two or more to provide additional processing power, storage connections, bandwidth, etc. Communication between two individual systems each configured similar to content delivery system 1010 may be made through network interface 1022 and/or 1023. Thus, one content delivery system could communicate with another content delivery system through the network 1020 and/or 1024. For example, a storage unit in one content delivery system could send data to a network interface engine of another content delivery system. As an example, these communications could be via TCP/IP protocols. Alternatively, the distributed interconnects 1080 of two content delivery systems 1010 may communicate directly. For example, a connection may be made directly between two switch fabrics, each switch fabric being the distributed interconnect 1080 of separate content delivery systems 1010.

FIGS. 1G-1J illustrate four exemplary clusters of content delivery systems 1010. It will be recognized that many other cluster arrangements may be utilized including more or less content delivery systems. As shown in FIGS. 1G-1J, each content delivery system may be configured as described above and include a distributive interconnect 1080 and a network interface processing engine 1030. Interfaces 1022 may connect the systems to a network 1020. As shown in FIG. 1G, two content delivery systems may be coupled together through the interface 1023 that is connected to each system's network interface processing engine 1030. FIG. 1H shows three systems coupled together as in FIG. 1G. The interfaces 1023 of each system may be coupled directly together as shown, may be coupled together through a network or may be coupled through a distributed interconnect (for example a switch fabric).

FIG. 1I illustrates a cluster in which the distributed interconnects 1080 of two systems are directly coupled together through an interface 1500. Interface 1500 may be any communication connection, such as a copper connection, optical fiber, wireless connection, etc. Thus, the distributed interconnects of two or more systems may directly communicate without communication through the processor engines of the content delivery systems 1010. FIG. 1J illustrates the distributed interconnects of three systems directly communicating without first requiring communication through the processor engines of the content delivery systems 1010. As shown in FIG. 1J, the interfaces 1500 each communicate with each other through another distributed interconnect 1600. Distributed interconnect 1600 may be a switched fabric or any other distributed interconnect.

The clustering techniques described herein may also be implemented through the use of the management interface 1062. Thus, communication between multiple content delivery systems 1010 also may be achieved through the management interface 1062

EXEMPLARY DATA AND COMMUNICATION FLOW PATHS

FIG. 1B illustrates one exemplary data and communication flow path configuration among modules of one embodiment of content delivery system 1010. The flow paths shown in FIG. 1B are just one example given to illustrate the significant improvements in data processing capacity and content delivery acceleration that may be realized using multiple content delivery engines that are individually optimized for different layers of the software stack and that are distributively interconnected as disclosed herein. The illustrated embodiment of FIG. 1B employs two network application processing modules 1070a and 1070b, and two network transport processing modules 1050a and 1050b that are communicatively coupled with single storage management processing module 1040a and single network interface processing module 1030a. The storage management processing module 1040a is in turn coupled to content sources 1090 and 1100. In FIG. 1B, inter-processor command or control flow (i.e. incoming or received data request) is represented by dashed lines, and delivered content data flow is represented by solid lines. Command and data flow between modules may be accomplished through the distributive interconnection 1080 (not shown), for example a switch fabric.

As shown in FIG. 1B, a request for content is received and processed by network interface processing module 1030a and then passed on to either of network transport

processing modules 1050a or 1050b for TCP/UDP processing, and then on to respective application processing modules 1070a or 1070b, depending on the transport processing module initially selected. After processing by the appropriate network application processing module, the request is passed on to storage management processor 1040a for processing and retrieval of the requested content from appropriate content sources 1090 and/or 1100. Storage management processing module 1040a then forwards the requested content directly to one of network transport processing modules 1050a or 1050b, utilizing the capability of distributive interconnection 1080 to bypass network application processing modules 1070a and 1070b. The requested content may then be transferred via the network interface processing module 1030a to the external network 1020. Benefits of bypassing the application processing modules with the delivered content include accelerated delivery of the requested content and offloading of workload from the application processing modules, each of which translate into greater processing efficiency and content delivery throughput. In this regard, throughput is generally measured in sustained data rates passed through the system and may be measured in bits per second. Capacity may be measured in terms of the number of files that may be partially cached, the number of TCP/IP connections per second as well as the number of concurrent TCP/IP connections that may be maintained or the number of simultaneous streams of a certain bit rate. In an alternative embodiment, the content may be delivered from the storage management processing module to the application processing module rather than bypassing the application processing module. This data flow may be advantageous if additional processing of the data is desired. For example, it may be desirable to decode or encode the data prior to delivery to the network.

To implement the desired command and content flow paths between multiple modules, each module may be provided with means for identification, such as a component ID. Components may be affiliated with content requests and content delivery to effect a desired module routing. The data-request generated by the network interface engine may include pertinent information such as the component ID of the various modules to be utilized in processing the request. For example, included in the data request sent to the storage management engine may be the component ID of the transport engine that is designated to receive the requested content data. When the storage management engine retrieves the data from the storage device and is ready to send the data to the next engine, the storage management engine knows which component ID to send the data to.

As further illustrated in FIG. 1B, the use of two network transport modules in conjunction with two network application processing modules provides two parallel processing paths for network transport and network application processing, allowing simultaneous processing of separate content requests and simultaneous delivery of separate content through the parallel processing paths, further increasing throughput/capacity and accelerating content delivery. Any two modules of a given engine may communicate with separate modules of another engine or may communicate with the same module of another engine. This is illustrated in FIG. 1B where the transport modules are shown to communicate with separate application modules and the application modules are shown to communicate with the same storage management module.

FIG. 1B illustrates only one exemplary embodiment of module and processing flow path configurations that may be employed using the disclosed method and system. Besides the embodiment illustrated in FIG. 1B, it will be understood that multiple modules may be additionally or alternatively employed for one or more other network content delivery engines (*e.g.,* storage management processing engine, network interface processing engine, system management processing engine, *etc.*) to create other additional or alternative parallel processing flow paths, and that any number of modules (*e.g.,* greater than two) may be employed for a given processing engine or set of processing engines so as to achieve more than two parallel processing flow paths. For example, in other possible embodiments, two or more different network transport processing engines may pass content requests to the same application unit, or vice-versa.

Thus, in addition to the processing flow paths illustrated in FIG. 1B, it will be understood that the disclosed distributive interconnection system may be employed to create other custom or optimized processing flow paths (*e.g.,* by bypassing and/or interconnecting any given number of processing engines in desired sequence/s) to fit the requirements or desired operability of a given content delivery application. For example, the content flow path of FIG. 1B illustrates an exemplary application in which the content is contained in content sources 1090 and/or 1100 that are coupled to the storage processing engine 1040. However as discussed above with reference to FIG. 1A, remote and/or live broadcast content may be provided to the content delivery system from the networks 1020 and/or 1024 via the second network interface connection 1023. In such a situation the content may be received by the network interface engine 1030 over interface connection 1023 and immediately re-

broadcast over interface connection 1022 to the network 1020. Alternatively, content may be proceed through the network interface connection 1023 to the network transport engine 1050 prior to returning to the network interface engine 1030 for re-broadcast over interface connection 1022 to the network 1020 or 1024. In yet another alternative, if the content requires some manner of application processing (for example encoded content that may need to be decoded), the content may proceed all the way to the application engine 1070 for processing. After application processing the content may then be delivered through the network transport engine 1050, network interface engine 1030 to the network 1020 or 1024.

In yet another embodiment, at least two network interface modules 1030a and 1030b may be provided, as illustrated in FIG. 1A. In this embodiment, a first network interface engine 1030a may receive incoming data from a network and pass the data directly to the second network interface engine 1030b for transport back out to the same or different network. For example, in the remote or live broadcast application described above, first network interface engine 1030a may receive content, and second network interface engine 1030b provide the content to the network 1020 to fulfill requests from one or more clients for this content. Peer-to-peer level communication between the two network interface engines allows first network interface engine 1030a to send the content directly to second network interface engine 1030b via distributive interconnect 1080. If necessary, the content may also be routed through transport processing engine 1050, or through transport processing engine 1050 and application processing engine 1070, in a manner described above.

Still yet other applications may exist in which the content required to be delivered is contained both in the attached content sources 1090 or 1100 and at other remote content sources. For example in a web caching application, not all content may be cached in the attached content sources, but rather some data may also be cached remotely. In such an application, the data and communication flow may be a combination of the various flows described above for content provided from the content sources 1090 and 1100 and for content provided from remote sources on the networks 1020 and/or 1024.

The content delivery system 1010 described above is configured in a peer-to-peer manner that allows the various engines and modules to communicate with each other directly as peers through the distributed interconnect. This is contrasted with a traditional server architecture in which there is a main CPU. Furthermore unlike the arbitrated bus of

traditional servers, the distributed interconnect 1080 provides a switching means which is not arbitrated and allows multiple simultaneous communications between the various peers. The data and communication flow may by-pass unnecessary peers such as the return of data from the storage management processing engine 1040 directly to the network interface processing engine 1030 as described with reference to FIG. 1B.

Communications between the various processor engines may be made through the use of a standardized internal protocol. Thus, a standardized method is provided for routing through the switch fabric and communicating between any two of the processor engines which operate as peers in the peer to peer environment. The standardized internal protocol provides a mechanism upon which the external network protocols may "ride" upon or be incorporated within. In this manner additional internal protocol layers relating to internal communication and data exchange may be added to the external protocol layers. The additional internal layers may be provided in addition to the external layers or may replace some of the external protocol layers (for example as described above portions of the external headers may be replaced by identifiers or tags by the network interface engine).

The standardized internal protocol may consist of a system of message classes, or types, where the different classes can independently include fields or layers that are utilized to identify the destination processor engine or processor module for communication, control, or data messages provided to the switch fabric along with information pertinent to the corresponding message class. The standardized internal protocol may also include fields or layers that identify the priority that a data packet has within the content delivery system. These priority levels may be set by each processing engine based upon system-wide policies. Thus, some traffic within the content delivery system may be prioritized over other traffic and this priority level may be directly indicated within the internal protocol call scheme utilized to enable communications within the system. The prioritization helps enable the predictive traffic flow between engines and end-to-end through the system such that service level guarantees may be supported.

Other internally added fields or layers may include processor engine state, system timestamps, specific message class identifiers for message routing across the switch fabric and at the receiving processor engine(s), system keys for secure control message exchange, flow control information to regulate control and data traffic flow and prevent congestion, and

specific address tag fields that allow hardware at the receiving processor engines to move specific types of data directly into system memory.

In one embodiment, the internal protocol may be structured as a set, or system of messages with common system defined headers that allows all processor engines and, potentially, processor engine switch fabric attached hardware, to interpret and process messages efficiently and intelligently. This type of design allows each processing engine, and specific functional entities within the processor engines, to have their own specific message classes optimized functionally for the exchanging their specific types control and data information. Some message classes that may be employed are: System Control messages for system management, Network Interface to Network Transport messages, Network Transport to Application Interface messages, File System to Storage engine messages, Storage engine to Network Transport messages, etc. Some of the fields of the standardized message header may include message priority, message class, message class identifier (subtype), message size, message options and qualifier fields, message context identifiers or tags, etc. In addition, the system statistics gathering, management and control of the various engines may be performed across the switch fabric connected system using the messaging capabilities.

By providing a standardized internal protocol, overall system performance may be improved. In particular, communication speed between the processor engines across the switch fabric may be increased. Further, communications between any two processor engines may be enabled. The standardized protocol may also be utilized to reduce the processing loads of a given engine by reducing the amount of data that may need to be processed by a given engine.

The internal protocol may also be optimized for a particular system application, providing further performance improvements. However, the standardized internal communication protocol may be general enough to support encapsulation of a wide range of networking and storage protocols. Further, while internal protocol may run on PCI, PCI-X, ATM, IB, Lightening I/O, the internal protocol is a protocol above these transport-level standards and is optimal for use in a switched (non-bus) environment such as a switch fabric. In addition, the internal protocol may be utilized to communicate devices (or peers) connected to the system in addition to those described herein. For example, a peer need not be a processing engine. In one example, a peer may be an ASIC protocol converter that is

coupled to the distributed interconnect as a peer but operates as a slave device to other master devices within the system. The internal protocol may also be as a protocol communicated between systems such as used in the clusters described above.

Thus a system has been provided in which the networking / server clustering / storage networking has been collapsed into a single system utilizing a common low-overhead internal communication protocol / transport system.

CONTENT DELIVERY ACCELERATION

As described above, a wide range of techniques have been provided for accelerating content delivery from the content delivery system 1010 to a network. By accelerating the speed at which content may be delivered, a more cost effective and higher performance system may be provided. These techniques may be utilized separately or in various combinations.

One content acceleration technique involves the use of a multi-engine system with dedicated engines for varying processor tasks. Each engine can perform operations independently and in parallel with the other engines without the other engines needing to freeze or halt operations. The engines do not have to compete for resources such as memory, I/O, processor time, etc. but are provided with their own resources. Each engine may also be tailored in hardware and/or software to perform specific content delivery task, thereby providing increasing content delivery speeds while requiring less system resources. Further, all data, regardless of the flow path, gets processed in a staged pipeline fashion such that each engine continues to process its layer of functionality after forwarding data to the next engine / layer.

Content acceleration is also obtained from the use of multiple processor modules within an engine. In this manner, parallelism may be achieved within a specific processing engine. Thus, multiple processors responding to different content requests may be operating in parallel within one engine.

Content acceleration is also provided by utilizing the multi-engine design in a peer to peer environment in which each engine may communicate as a peer. Thus, the communications and data paths may skip unnecessary engines. For example, data may be

communicated directly from the storage processing engine to the transport processing engine without have to utilize resources of the application processing engine.

Acceleration of content delivery is also achieved by removing or stripping the contents of some protocol layers in one processing engine and replacing those layers with identifiers or tags for use with the next processor engine in the data or communications flow path. Thus, the processing burden placed on the subsequent engine may be reduced. In addition, the packet size transmitted across the distributed interconnect may be reduced. Moreover, protocol processing may be off-loaded from the storage and/or application processors, thus freeing those resources to focus on storage or application processing.

Content acceleration is also provided by using network processors in a network endpoint system. Network processors generally are specialized to perform packet analysis functions at intermediate network nodes, but in the content delivery system disclosed the network processors have been adapted for endpoint functions. Furthermore, the parallel processor configurations within a network processor allow these endpoint functions to be performed efficiently.

In addition, content acceleration has been provided through the use of a distributed interconnection such as a switch fabric. A switch fabric allows for parallel communications between the various engines and helps to efficiently implement some of the acceleration techniques described herein.

It will be recognized that other aspects of the content delivery system 1010 also provide for accelerated delivery of content to a network connection. Further, it will be recognized that the techniques disclosed herein may be equally applicable to other network endpoint systems and even non-endpoint systems.

EXEMPLARY HARDWARE EMBODIMENTS

FIGS. 1C-1F illustrate just a few of the many multiple network content delivery engine configurations possible with one exemplary hardware embodiment of content delivery system 1010. In each illustrated configuration of this hardware embodiment, content delivery system 1010 includes processing modules that may be configured to operate as content delivery engines 1030, 1040, 1050, 1060, and 1070 communicatively coupled via distributive

interconnection 1080. As shown in FIG. 1C, a single processor module may operate as the network interface processing engine 1030 and a single processor module may operate as the system management processing engine 1060. Four processor modules 1001 may be configured to operate as either the transport processing engine 1050 or the application processing engine 1070. Two processor modules 1003 may operate as either the storage processing engine 1040 or the transport processing engine 1050. The Gigabit (Gb) Ethernet front end interface 1022, system management interface 1062 and dual fibre channel arbitrated loop 1092 are also shown.

As mentioned above, the distributive interconnect 1080 may be a switch fabric based interconnect. As shown in FIG. 1C, the interconnect may be an IBM PRIZMA-E eight/sixteen port switch fabric 1081. In an eight port mode, this switch fabric is an 8 x 3.54 Gbps fabric and in a sixteen port mode, this switch fabric is a 16 x 1.77 Gbps fabric. The eight/sixteen port switch fabric may be utilized in an eight port mode for performance optimization. The switch fabric 1081 may be coupled to the individual processor modules through interface converter circuits 1082, such as IBM UDASL switch interface circuits. The interface converter circuits 1082 convert the data aligned serial link interface (DASL) to a UTOPIA (Universal Test and Operations PHY Interface for ATM) parallel interface. FPGAs (field programmable gate array) may be utilized in the processor modules as a fabric interface on the processor modules as shown in FIG. 1C. These fabric interfaces provide a 64/66Mhz PCI interface to the interface converter circuits 1082. FIG. 4 illustrates a functional block diagram of such a fabric interface 34. As explained below, the interface 34 provides an interface between the processor module bus and the UDASL switch interface converter circuit 1082. As shown in FIG. 4, at the switch fabric side, a physical connection interface 41 provides connectivity at the physical level to the switch fabric. An example of interface 41 is a parallel bus interface complying with the UTOPIA standard. In the example of FIG. 4, interface 41 is a UTOPIA 3 interface providing a 32-bit 110 Mhz connection. However, the concepts disclosed herein are not protocol dependent and the switch fabric need not comply with any particular ATM or non ATM standard.

Still referring to FIG. 4, SAR (segmentation and reassembly) unit 42 has appropriate SAR logic 42a for performing segmentation and reassembly tasks for converting messages to fabric cells and vice-versa as well as message classification and message class-to-queue routing, using memory 42b and 42c for transmit and receive queues. This permits different

classes of messages and permits the classes to have different priority. For example, control messages can be classified separately from data messages, and given a different priority. All fabric cells and the associated messages may be self routing, and no out of band signaling is required.

A special memory modification scheme permits one processor module to write directly into memory of another. This feature is facilitated by switch fabric interface 34 and in particular by its message classification capability. Commands and messages follow the same path through switch fabric interface 34, but can be differentiated from other control and data messages. In this manner, processes executing on processor modules can communicate directly using their own memory spaces.

Bus interface 43 permits switch fabric interface 34 to communicate with the processor of the processor module via the module device or I/O bus. An example of a suitable bus architecture is a PCI architecture, but other architectures could be used. Bus interface 43 is a master/target device, permitting interface 43 to write and be written to and providing appropriate bus control. The logic circuitry within interface 43 implements a state machine that provides the communications protocol, as well as logic for configuration and parity.

Referring again to FIG. 1C, network processor 1032 (for example a MOTOROLA C-Port C-5 network processor) of the network interface processing engine 1030 may be coupled directly to an interface converter circuit 1082 as shown. As mentioned above and further shown in FIG. 1C, the network processor 1032 also may be coupled to the network 1020 by using a VITESSE GbE SERDES (serializer-deserializer) device (for example the VSC7123) and an SFP (small form factor pluggable) optical transceiver for LC fibre connection.

The processor modules 1003 include a fibre channel (FC) controller as mentioned above and further shown in FIG. 1C. For example, the fibre channel controller may be the LSI SYMFC929 dual 2GBaud fibre channel controller. The fibre channel controller enables communication with the fibre channel 1092 when the processor module 1003 is utilized as a storage processing engine 1040. Also illustrated in FIGS. 1C-1F is optional adjunct processing unit 1300 that employs a POWER PC processor with SDRAM. The adjunct processing unit is shown coupled to network processor 1032 of network interface processing

engine 1030 by a PCI interface. Adjunct processing unit 1300 may be employed for monitoring system parameters such as temperature, fan operation, system health, etc.

As shown in FIGS. 1C-1F, each processor module of content delivery engines 1030, 1040, 1050, 1060, and 1070 is provided with its own synchronous dynamic random access memory ("SDRAM") resources, enhancing the independent operating capabilities of each module. The memory resources may be operated as ECC (error correcting code) memory. Network interface processing engine 1030 is also provided with static random access memory ("SRAM"). Additional memory circuits may also be utilized as will be recognized by those skilled in the art. For example, additional memory resources (such as synchronous SRAM and non-volatile FLASH and EEPROM) may be provided in conjunction with the fibre channel controllers. In addition, boot FLASH memory may also be provided on the of the processor modules.

The processor modules 1001 and 1003 of FIG. 1C may be configured in alternative manners to implement the content delivery processing engines such as the network interface processing engine 1030, storage processing engine 1040, transport processing engine 1050, system management processing engine 1060, and application processing engine 1070. Exemplary configurations are shown in FIGS. 1D-1F, however, it will be recognized that other configurations may be utilized.

As shown in FIG. 1D, two Pentium III based processing modules may be utilized as network application processing modules 1070a and 1070b of network application processing engine 1070. The remaining two Pentium III-based processing modules are shown in FIG. 1D configured as network transport / protocol processing modules 1050a and 1050b of network transport / protocol processing engine 1050. The embodiment of FIG. 1D also includes two POWER PC-based processor modules, configured as storage management processing modules 1040a and 1040b of storage management processing engine 1040. A single MOTOROLA C-Port C-5 based network processor is shown employed as network interface processing engine 1030, and a single Pentium III-based processing module is shown employed as system management processing engine 1060.

In FIG. 1E, the same hardware embodiment of FIG. 1C is shown alternatively configured so that three Pentium III-based processing modules function as network

application processing modules 1070a, 1070b and 1070c of network application processing engine 1070, and so that the sole remaining Pentium III-based processing module is configured as a network transport processing module 1050a of network transport processing engine 1050. As shown, the remaining processing modules are configured as in FIG. 1D.

In FIG. 1F, the same hardware embodiment of FIG. 1C is shown in yet another alternate configuration so that three Pentium III-based processing modules function as application processing modules 1070a, 1070b and 1070c of network application processing engine 1070. In addition, the network transport processing engine 1050 includes one Pentium III-based processing module that is configured as network transport processing module 1050a, and one POWER PC-based processing module that is configured as network transport processing module 1050b. The remaining POWER PC-based processor module is configured as storage management processing module 1040a of storage management processing engine 1040.

It will be understood with benefit of this disclosure that the hardware embodiment and multiple engine configurations thereof illustrated in FIGS. 1C-1F are exemplary only, and that other hardware embodiments and engine configurations thereof are also possible. It will further be understood that in addition to changing the assignments of individual processing modules to particular processing engines, distributive interconnect 1080 enables the various processing flow paths between individual modules employed in a particular engine configuration in a manner as described in relation to FIG. 1B. Thus, for any given hardware embodiment and processing engine configuration, a number of different processing flow paths may be employed so as to optimize system performance to suit the needs of particular system applications.

SINGLE CHASSIS DESIGN

As mentioned above, the content delivery system 1010 may be implemented within a single chassis, such as for example, a 2U chassis. The system may be expanded further while still remaining a single chassis system. In particular, utilizing a multiple processor module or blade arrangement connected through a distributive interconnect (for example a switch fabric) provides a system that is easily scalable. The chassis and interconnect may be configured with expansion slots provided for adding additional processor modules. Additional processor modules may be provided to implement additional applications within

the same chassis. Alternatively, additional processor modules may be provided to scale the bandwidth of the network connection. Thus, though describe with respect to a 1Gbps Ethernet connection to the external network, a 10 Gbps, 40 Gbps or more connection may be established by the system through the use of more network interface modules. Further, additional processor modules may be added to address a system's particular bottlenecks without having to expand all engines of the system. The additional modules may be added during a systems initial configuration, as an upgrade during system maintenance or even hot plugged during system operation.

ALTERNATIVE SYSTEMS CONFIGURATIONS

Further, the network endpoint system techniques disclosed herein may be implemented in a variety of alternative configurations that incorporate some, but not necessarily all, of the concepts disclosed herein. For example, FIGS. 2 and 2A disclose two exemplary alternative configurations. It will be recognized, however, that many other alternative configurations may be utilized while still gaining the benefits of the inventions disclosed herein.

FIG. 2 is a more generalized and functional representation of a content delivery system showing how such a system may be alternately configured to have one or more of the features of the content delivery system embodiments illustrated in FIGS. 1A-1F. FIG. 2 shows content delivery system 200 coupled to network 260 from which content requests are received and to which content is delivered. Content sources 265 are shown coupled to content delivery system 200 via a content delivery flow path 263 that may be, for example, a storage area network that links multiple content sources 265. A flow path 203 may be provided to network connection 272, for example, to couple content delivery system 200 with other network appliances, in this case one or more servers 201 as illustrated in FIG. 2.

In FIG. 2 content delivery system 200 is configured with multiple processing and memory modules that are distributively interconnected by inter-process communications path 230 and inter-process data movement path 235. Inter-process communications path 230 is provided for receiving and distributing inter-processor command communications between the modules and network 260, and interprocess data movement path 235 is provided for receiving and distributing inter-processor data among the separate modules. As illustrated in FIGS. 1A-1F, the functions of inter-process communications path 230 and inter-process data

movement path 235 may be together handled by a single distributive interconnect 1080 (such as a switch fabric, for example), however, it is also possible to separate the communications and data paths as illustrated in FIG. 2, for example using other interconnect technology.

FIG. 2 illustrates a single networking subsystem processor module 205 that is provided to perform the combined functions of network interface processing engine 1030 and transport processing engine 1050 of FIG. 1A. Communication and content delivery between network 260 and networking subsystem processor module 205 are made through network connection 270. For certain applications, the functions of network interface processing engine 1030 and transport processing engine 1050 of FIG. 1A may be so combined into a single module 205 of FIG. 2 in order to reduce the level of communication and data traffic handled by communications path 230 and data movement path 235 (or single switch fabric), without adversely impacting the resources of application processing engine or subsystem module. If such a modification were made to the system of FIG. 1A, content requests may be passed directly from the combined interface/transport engine to network application processing engine 1070 via distributive interconnect 1080. Thus, as previously described the functions of two or more separate content delivery system engines may be combined as desired (*e.g.*, in a single module or in multiple modules of a single processing blade), for example, to achieve advantages in efficiency or cost.

In the embodiment of FIG. 2, the function of network application processing engine 1070 of FIG. 1A is performed by application processing subsystem module 225 of FIG. 2 in conjunction with application RAM subsystem module 220 of FIG. 2. System monitor module 240 communicates with server/s 201 through flow path 203 and Gb Ethernet network interface connection 272 as also shown in FIG. 2. The system monitor module 240 may provide the function of the system management engine 1060 of FIG. 1A and/or other system policy / filter functions such as may also be implemented in the network interface processing engine 1030 as described above with reference to FIG. 1A.

Similarly, the function of network storage management engine 1040 is performed by storage subsystem module 210 in conjunction with file system cache subsystem module 215. Communication and content delivery between content sources 265 and storage subsystem module 210 are shown made directly through content delivery flowpath 263 through fibre channel interface connection 212. Shared resources subsystem module 255 is shown

provided for access by each of the other subsystem modules and may include, for example, additional processing resources, additional memory resources such as RAM, *etc.*

Additional processing engine capability (*e.g.*, additional system management processing capability, additional application processing capability, additional storage processing capability, encryption / decryption processing capability, compression / decompression processing capability, encoding / decoding capability, other processing capability, *etc.*) may be provided as desired and is represented by other subsystem module 275. Thus, as previously described the functions of a single network processing engine may be sub-divided between separate modules that are distributively interconnected. The sub-division of network processing engine tasks may also be made for reasons of efficiency or cost, and/or may be taken advantage of to allow resources (*e.g.*, memory or processing) to be shared among separate modules. Further, additional shared resources may be made available to one or more separate modules as desired.

Also illustrated in FIG. 2 are optional monitoring agents 245 and resources 250. In the embodiment of FIG. 2, each monitoring agent 245 may be provided to monitor the resources 250 of its respective processing subsystem module, and may track utilization of these resources both within the overall system 200 and within its respective processing subsystem module. Examples of resources that may be so monitored and tracked include, but are not limited to, processing engine bandwidth, Fibre Channel bandwidth, number of available drives, IOPS (input/output operations per second) per drive and RAID (redundant array of inexpensive discs) levels of storage devices, memory available for caching blocks of data, table lookup engine bandwidth, availability of RAM for connection control structures and outbound network bandwidth availability, shared resources (such as RAM) used by streaming application on a per-stream basis as well as for use with connection control structures and buffers, bandwidth available for message passing between subsystems, bandwidth available for passing data between the various subsystems, *etc.*

Information gathered by monitoring agents 245 may be employed for a wide variety of purposes including for billing of individual content suppliers and/or users for pro-rata use of one or more resources, resource use analysis and optimization, resource health alarms, *etc.* In addition, monitoring agents may be employed to enable the deterministic delivery of content by system 200 as described in concurrently filed, co-pending United States patent

application number 09/797,200 entitled "Systems and Methods for the Deterministic Management of Information," which is incorporated herein by reference.

In operation, content delivery system 200 of FIG. 2 may be configured to wait for a request for content or services prior to initiating content delivery or performing a service. A request for content, such as a request for access to data, may include, for example, a request to start a video stream, a request for stored data, *etc.* A request for services may include, for example, a request for to run an application, to store a file, *etc.* A request for content or services may be received from a variety of sources. For example, if content delivery system 200 is employed as a stream server, a request for content may be received from a client system attached to a computer network or communication network such as the Internet. In a larger system environment, e.g., a data center, a request for content or services may be received from a separate subcomponent or a system management processing engine, that is responsible for performance of the overall system or from a sub-component that is unable to process the current request. Similarly, a request for content or services may be received by a variety of components of the receiving system. For example, if the receiving system is a stream server, networking subsystem processor module 205 might receive a content request. Alternatively, if the receiving system is a component of a larger system, e.g., a data center, system management processing engine may be employed to receive the request.

Upon receipt of a request for content or services, the request may be filtered by system monitor 240. Such filtering may serve as a screening agent to filter out requests that the receiving system is not capable of processing (*e.g.,* requests for file writes from read-only system embodiments, unsupported protocols, content/services unavailable on system 200, *etc.*). Such requests may be rejected outright and the requestor notified, may be re-directed to a server 201 or other content delivery system 200 capable of handling the request, or may be disposed of any other desired manner.

Referring now in more detail to one embodiment of FIG. 2 as may be employed in a stream server configuration, networking processing subsystem module 205 may include the hardware and/or software used to run TCP/IP (Transmission Control Protocol/Internet Protocol), UDP/IP (User Datagram Protocol/Internet Protocol), RTP (Real-Time Transport Protocol), Internet Protocol (IP), Wireless Application Protocol (WAP) as well as other networking protocols. Network interface connections 270 and 272 may be considered part of

networking subsystem processing module 205 or as separate components. Storage subsystem module 210 may include hardware and/or software for running the Fibre Channel (FC) protocol, the SCSI (Small Computer Systems Interface) protocol, iSCSI protocol as well as other storage networking protocols. FC interface 212 to content delivery flowpath 263 may be considered part of storage subsystem module 210 or as a separate component. File system cache subsystem module 215 may include, in addition to cache hardware, one or more cache management algorithms as well as other software routines.

Application RAM subsystem module 220 may function as a memory allocation subsystem and application processing subsystem module 225 may function as a stream-serving application processor bandwidth subsystem. Among other services, application RAM subsystem module 220 and application processing subsystem module 225 may be used to facilitate such services as the pulling of content from storage and/or cache, the formatting of content into RTSP (Real-Time Streaming Protocol) or another streaming protocol as well the passing of the formatted content to networking subsystem 205.

As previously described, system monitor module 240 may be included in content delivery system 200 to manage one or more of the subsystem processing modules, and may also be used to facilitate communication between the modules.

In part to allow communications between the various subsystem modules of content delivery system 200, inter-process communication path 230 may be included in content delivery system 200, and may be provided with its own monitoring agent 245. Inter-process communications path 230 may be a reliable protocol path employing a reliable IPC (Inter-process Communications) protocol. To allow data or information to be passed between the various subsystem modules of content delivery system 200, inter-process data movement path 235 may also be included in content delivery system 200, and may be provided with its own monitoring agent 245. As previously described, the functions of inter-process communications path 230 and inter-process data movement path 235 may be together handled by a single distributive interconnect 1080, that may be a switch fabric configured to support the bandwidth of content being served.

In one embodiment, access to content source 265 may be provided via a content delivery flow path 263 that is a fibre channel storage area network (SAN), a switched

technology. In addition, network connectivity may be provided at network connection 270 (*e.g.*, to a front end network) and/or at network connection 272 (*e.g.*, to a back end network) via switched gigabit Ethernet in conjunction with the switch fabric internal communication system of content delivery system 200. As such, that the architecture illustrated in FIGURE 2 may be generally characterized as equivalent to a networking system.

One or more shared resources subsystem modules 255 may also be included in a stream server embodiment of content delivery system 200, for sharing by one or more of the other subsystem modules. Shared resources subsystem module 255 may be monitored by the monitoring agents 245 of each subsystem sharing the resources. The monitoring agents 245 of each subsystem module may also be capable of tracking usage of shared resources 255. As previously described, shared resources may include RAM (Random Access Memory) as well as other types of shared resources.

Each monitoring agent 245 may be present to monitor one or more of the resources 250 of its subsystem processing module as well as the utilization of those resources both within the overall system and within the respective subsystem processing module. For example, monitoring agent 245 of storage subsystem module 210 may be configured to monitor and track usage of such resources as processing engine bandwidth, Fibre Channel bandwidth to content delivery flow path 263, number of storage drives attached, number of input/output operations per second (IOPS) per drive and RAID levels of storage devices that may be employed as content sources 265. Monitoring agent 245 of file system cache subsystem module 215 may be employed monitor and track usage of such resources as processing engine bandwidth and memory employed for caching blocks of data. Monitoring agent 245 of networking subsystem processing module 205 may be employed to monitor and track usage of such resources as processing engine bandwidth, table lookup engine bandwidth, RAM employed for connection control structures and outbound network bandwidth availability. Monitoring agent 245 of application processing subsystem module 225 may be employed to monitor and track usage of processing engine bandwidth. Monitoring agent 245 of application RAM subsystem module 220 may be employed to monitor and track usage of shared resource 255, such as RAM, which may be employed by a streaming application on a per-stream basis as well as for use with connection control structures and buffers. Monitoring agent 245 of inter-process communication path 230 may be employed to monitor and track usage of such resources as the bandwidth used for message

passing between subsystems while monitoring agent 245 of inter-process data movement path 235 may be employed to monitor and track usage of bandwidth employed for passing data between the various subsystem modules.

The discussion concerning FIG. 2 above has generally been oriented towards a system designed to deliver streaming content to a network such as the Internet using, for example, Real Networks, Quick Time or Microsoft Windows Media streaming formats. However, the disclosed systems and methods may be deployed in any other type of system operable to deliver content, for example, in web serving or file serving system environments. In such environments, the principles may generally remain the same. However for application processing embodiments, some differences may exist in the protocols used to communicate and the method by which data delivery is metered (via streaming protocol, versus TCP/IP windowing).

FIG. 2A illustrates an even more generalized network endpoint computing system that may incorporate at least some of the concepts disclosed herein. As shown in Figure 2A, a network endpoint system 10 may be coupled to an external network 11. The external network 11 may include a network switch or router coupled to the front end of the endpoint system 10. The endpoint system 10 may be alternatively coupled to some other intermediate network node of the external network. The system 10 may further include a network engine 9 coupled to an interconnect medium 14. The network engine 9 may include one or more network processors. The interconnect medium 14 may be coupled to a plurality of processor units 13 through interfaces 13a. Each processor unit 13 may optionally be couple to data storage (in the exemplary embodiment shown each unit is couple to data storage). More or less processor units 13 may be utilized than shown in FIG. 2A.

The network engine 9 may be a processor engine that performs all protocol stack processing in a single processor module or alternatively may be two processor modules (such as the network interface engine 1030 and transport engine 1050 described above) in which split protocol stack processing techniques are utilized. Thus, the functionality and benefits of the content delivery system 1010 described above may be obtained with the system 10. The interconnect medium 14 may be a distributive interconnection (for example a switch fabric) as described with reference to FIG. 1A. All of the various computing, processing, communication, and control techniques described above with reference to FIGS. 1A-1F and 2

may be implemented within the system 10. It will therefore be recognized that these techniques may be utilized with a wide variety of hardware and computing systems and the techniques are not limited to the particular embodiments disclosed herein.

The system 10 may consist of a variety of hardware configurations. In one configuration the network engine 9 may be a stand-alone device and each processing unit 13 may be a separate server. In another configuration the network engine 9 may be configured within the same chassis as the processing units 13 and each processing unit 13 may be a separate server card or other computing system. Thus, a network engine (for example an engine containing a network processor) may provide transport acceleration and be combined with multi-server functionality within the system 10. The system 10 may also include shared management and interface components. Alternatively, each processing unit 13 may be a processing engine such as the transport processing engine, application engine, storage engine, or system management engine of FIG. 1A. In yet another alternative, each processing unit may be a processor module (or processing blade) of the processor engines shown in the system of FIG. 1A.

FIG. 2B illustrates yet another use of a network engine 9. As shown in FIG. 2B, a network engine 9 may be added to a network interface card 35. The network interface card 35 may further include the interconnect medium 14 which may be similar to the distributed interconnect 1080 described above. The network interface card may be part of a larger computing system such as a server. The network interface card may couple to the larger system through the interconnect medium 14. In addition to the functions described above, the network engine 9 may perform all traditional functions of a network interface card.

It will be recognized that all the systems described above (FIGS. 1A, 2, 2A, and 2B) utilize a network engine between the external network and the other processor units that are appropriate for the function of the particular network node. The network engine may therefore offload tasks from the other processors. The network engine also may perform "look ahead processing" by performing processing on a request before the request reaches whatever processor is to perform whatever processing is appropriate for the network node. In this manner, the system operations may be accelerated and resources utilized more efficiently.

NETWORK SECURITY

The computing systems disclosed herein may incorporate the use of security measures to protect against network security attacks, including but not limited to, DoS attacks, syn attacks, ping attacks, Trojan horse attacks, unauthorized access attacks, etc. The security measures disclosed herein may be incorporated within a network endpoint computing system. More particularly, the security measures may be implemented with intelligent security hardware that is placed at the network interface of the endpoint system. The intelligent security hardware may be coupled on one side to the external network and on the other side to the rest of the endpoint computing system through an interconnect medium. The interconnection medium may be in one example a switch fabric. As used herein intelligent security hardware may also be called a security accelerator.

Preferably the intelligent security hardware is programmable and has the capability of performing "look ahead processing" so that processing may be performed on network packets in order to reduce the workload placed upon the endpoint systems resources. For example, the intelligent security hardware may include the capability of decoding incoming network packet headers and deciding where to forward the packet. One such type of intelligent hardware is a network processor or a network processor coupled with other processors or hardware. Any hardware capable of performing the functions described herein, however, may be suitable. The intelligent security hardware may receive incoming packets, decode headers, and determine if the packets are authentic or if the packets are part of a security attack. The intelligent security hardware may be programmable to implement any of a wide variety of security algorithms utilized to detect an attack or unauthorized access. Thus although the security functions are hardware based, the intelligent security hardware offers flexibility in being configured for serving a variety of security functions and for addressing a variety of changing security threats.

By providing intelligent security hardware (or a security accelerator) that can perform security functions at the front end of the endpoint computing system, the often complex and CPU intensive security functions may be off-loaded from the rest of the endpoint computing system. Security protections are, therefore, implemented without impacting the performance of the rest of the endpoint computing system. Further, unlike a conventional network front end, such as the network controllers of conventional servers, and unlike many firewalls, the intelligent security hardware need not DMA (direct memory access) packets into RAM.

Instead, it processes the packets as they arrive. This processing is sometimes referred to as "pass through processing". Thus, attacks may be detected quickly without inundating resources at the endpoint. In this manner, the security functions may be accelerated. The intelligent security hardware described herein may be considered a security accelerator.

Thus, the intelligent security hardware (or security accelerator) may be utilized in network endpoint system connected to a network that carries data in packet format. A security accelerator is programmed to receive packets from the network and to examine each packet to determine whether data in the packet represents a potential security violation. The network endpoint system may further include one or more processing units that are programmed to execute some form of server or client application programming or content delivery and to thereby respond to requests contained within the packets. An interconnection medium may be used to directly connect the security accelerator to the processing units.

The security accelerator may be utilized at the front end of the endpoint, and thus, eliminate the need for a firewall. Security tools are offloaded from other endpoint resources to the security accelerator, so that the other resources can be devoted to the other tasks that the endpoint system is to perform. The security accelerator's "look ahead" processing unburdens the processing units that must perform the basic tasks appropriate for that particular network endpoint.

The security accelerator may be implemented with either a network processor or a CPU type general processor. When a network processor is used, it performs "pass through" type processing that especially suitable for many types of security algorithms. This type of processing can be more suited for these algorithms than the state-modification intensive and memory-access intensive processing of a CPU type processor.

The security accelerator can detect attempted security breaches very quickly. It can take immediate action, such as discarding the packet or notifying the network administrator. By utilizing a programmable security accelerator, the security accelerator may be easily updated by means of a simple download, making it easy to upgrade the security accelerator to detect new types of attacks. Thus, the security accelerator provides flexibility for providing counter measures to new security attacks as the new types of attacks become known.

As mentioned above, the intelligent security hardware or security accelerator may be implemented through the use of a network processor. The use of a network processor for implementing the security functions will initially be discussed for illustrative purposes with regards to the content delivery systems of FIGS. 1A-1F and 2 and the network endpoint computing system of FIG. 2A, all of which are discussed above. It will be recognized that the security techniques described may be utilized in other systems or applications as will be described in more detail below.

As described above, the content delivery system 1010 of FIGS. 1A-1F may include a network interface processing engine 1030 which may include one or more network interface modules The network interface modules may include a network processor. The network interface processing engine operates as an interface between the external network and the distributed interconnection 1080 (for example a switch fabric). Similarly, the networking subsystem module of FIG. 2 may include a network processor and the system of FIG. 2A may include a network processor 12. As described above, a network processor is generally used in network switching devices and is specialized in decoding incoming network packet headers and deciding where to forward the packet. The network processor in all of theses systems may be programmed to provide the security measures described herein to achieve the benefits mentioned above. the specialized abilities of a network processor are therefore being applied security measures, a non-standard function for network processors.

EXEMPLARY SECURITY MEASURES

A variety of exemplary security measures are discussed herein with reference to utilizing a network processor as the intelligent security hardware / security accelerator. It will be recognized that other hardware may be utilized to achieve the benefits described herein.

As mentioned above, a network processor can be programmed to implement a full array of security or firewall tools. Though some security measures are described below, it will be recognized that the programmability of a network processor will allow any number of security measures to be implemented.

A common security attack is a denial of service (DoS) attack. To protect against DoS attacks, the network processor is programmed to receive incoming packets, decode headers,

and determine whether the packet is authentic. If the packet is authenticated, the network processor forwards it to the appropriate processor module or unit or other sub-system of the computing system. The techniques programmed into the network processor that determines whether a packet is part of a DoS attack may be implemented with various new and/or known techniques. For example, the receipt of a large number of new TCP open connection requests over a very short time period from the same source is indicative of a DoS attack. As is known in the field of network security, various tables, counters, and logic may be used to determine if a DoS attack is occurring.

Unlike a conventional network front end, such as the network controllers of conventional servers, and unlike many firewalls, the network processor does not DMA (direct memory access) packets into RAM. Instead, it processes the packets as they arrive. This processing is sometimes referred to as "pass through processing". Thus, DoS attacks may be detected quickly without inundating resources at the endpoint.

A "syn" attack is another potential network attack. In a syn attack, syn requests cause the TCP/IP stack to overfill. To counter this type of attack, the network processor may be programmed to monitor the frequency and same-source syn requests. If a threshold number of either of these counts is exceeded, the network processor is programmed to take action, such as throttling down requests or alerting a network administrator.

Other types of attacks make use of "bogus" source addresses. To thwart such attacks, the network processor can be programmed with various algorithms directed to authenticating source IP addresses. Yet another attack is known as “ping” attacks. In ping attacks, a ping request is received at one system or network (the “amplifier”) with a broadcast destination request. The node that is the victim of the attack is falsely listed as the address to which ping responses will be directed. The amplifier that received the ping request will broadcast the request to many endpoints within its network, thus generating massive amounts of ping replies directed at the victim of the attack. To thwart this attack at the victim endpoint, a network processor at the victim’s endpoint system may be programmed to detect a massive amount of replies to a ping. To thwart this attack at the amplifier receiving the ping request, a network processor in the amplifier may be programmed to ignore broadcast ping requests or to limit the number of ping responses provided from the network.

The network processor may be further programmed to implement authentication verification and access control lists (ACL). This type of security task is thereby offloaded from the other processing or CPU resources. For example, an authentication handshake that verifies the identity of a user could be offloaded from the other resources of the network endpoint system. Furthermore, once the user is identified, the network processor may be programmed to determine what content is accessible by that user and discard unauthorized content access requests.

The network processor can be further programmed to implement packet filtering. In packet filtering, the header information of a packet may be analyzed and compared to large filter lists. A common limitation of filtering in traditional security systems is that large filter lists are required, leading to excessive memory and processor bandwidth usage to search for a filter match for every incoming packet. By providing a network processor separate from the other processor resources, the filter lists are not limited by other system resources such as memory and CPU bandwidth. Further complex filters used to analyze many different header types may be implemented without impacting the other processing resources.

A Trojan horse attack is another security attack. In a Trojan horse attack a program may execute to cause an unauthorized data transfer out to the external network. A firewall may assume that the data being transferred out was from an authorized user, whereas the network processor could be programmed to permit outgoing data to be sent only to authorized users. Alternatively, the network processor may be programmed to allow outgoing data to only be initiated by trusted connections. Thus, the network processor may not only provide security related to incoming events, but also provide security related to outgoing events. In this sense, network processor provides "bi-directional" security.

The use of intelligent security hardware such as the programmable network processor further allows security provisions to be especially tailored to the type of network computing system that incorporates the hardware. For example, the network endpoint systems of FIGS. 1A-1F and 2 are content delivery systems. In these systems the network processor permits the security tools to be programmed especially for content delivery systems. The network processor could be programmed to allow access only for types of requests that are supported by the content delivery system. Thus, if the content delivery system only serves HTTP requests, the security tools can be directed specifically to that protocol and known security

weaknesses in that protocols. The network processor could also be further programmed to discard all non HTTP requests. Alternatively, if a system is known not to support a specific protocol, then the network processor may be programmed to disregard all connection requests for that type of protocol. A system that does not serve FTP connections may be, for example, programmed to discard all FTP connection requests.

Therefore, the security measures described herein may provide full firewall functionality and eliminate the need for a separate firewall device. Although many of the exemplary security measures have been described with reference to a network processor, it will be recognized that other programmable hardware may implement this same functionality.

OTHER SYSTEMS FOR USING THE SECURTY HARDWARE

The exemplary security techniques described above have been primarily described with reference to a network endpoint system such as a content delivery system configured in a multi-module (or multi-blade) manner. However, the security techniques are not limited to use in such systems and many alternative network endpoint systems may utilize these techniques. Further, although generally described with relation to a server type endpoint, a client endpoint system could also have intelligent security hardware (such as a network processor) programmed to implement security tools for attacks that might occur at a client system.

In the examples of FIGS. 5-8 described below, the network processing systems are network endpoint systems. The security techniques described are not, however, limited to only network endpoints but may also be utilized within hardware located at various intermediate network nodes. The hardware located at various intermediate nodes that incorporates the security techniques described herein may include routers, switches or other network hardware. In general, the intelligent security hardware (or security accelerator) could be at any network node. Although the various systems differ in their overall architectures, in each system, the security accelerator executes security tools of the type described above.

No matter what type of system is utilized, a common characteristic of each system is that the security accelerator resides between a network and other processing units that are appropriate for a given network node or between a network and other network nodes. The

security accelerator may be, but is not necessarily implemented with a network processor. A CPU type general purpose processor may be used instead of, or in addition to a network processor. Or, either a network processor or a CPU could be used with additional hardware logic. Regardless of what security hardware is used to implement the security accelerator, the security hardware performs "look ahead" processing on data as it is received. This processing, specifically directed to executing network security tools, is performed on data before the data reaches whatever device is to perform whatever basic processing is appropriate for the network node or before the data is routed to the next network node. The security accelerator thereby offloads the security processing from those processing units or network nodes.

FIGURE 5 illustrates a system 50 in which security accelerator 53 is a stand alone unit that is a separate physical entity from servers 51. The security accelerator 53 and the servers 51 are connected by an interconnection medium 52. The security accelerator 53 has an Ethernet connection on the front end and an interconnection medium connection on the back end. The interconnection medium 52 could be any message passing medium, including those described above, e.g., switch fabric, bus, or shared memory. Alternatively, a network connection such as a LAN, could be used.

FIGURE 6 illustrates a multi-slot chassis or fixed configuration chassis system 60. In system 60, the security accelerator 63 and servers 61 are implemented as cards within the same physical chassis, connected by an interconnection medium 62. Interconnection medium 62 may be any of the various interconnection media described above.

FIGURE 7 illustrates a security accelerator 75 in a system 70 that is similar to the system 60 except that the functionality of the server cards 71 - 73 has been split out; and thus, system 70 is an asymmetric multi-processing model. Interconnection medium 74 is implemented in a manner similar to interconnection medium 62 of system 60.

Both system 60 and system 70 integrate network transport acceleration and server functionality within a common chassis. This provides cost reduction in terms of shared power supplies and physical structural components. A number of serving units may be placed in a rack, and may share the same management and interface components. Higher

interconnection speeds occur within a single chassis as compared to connections between physically separate devices.

FIGURE 8 illustrates a system 80 in which security accelerator 81 is embedded on a network interface card (NIC) 80. The NIC may be a stand alone card or incorporated with a server. The security accelerator may be used to replace the server network interface card (NIC) and perform all the responsibilities of the server NIC in addition to the security measures described herein. All network interface and security processing elements may be integrated as a single component, which may be a chip as well as a card. In system 80, security accelerator 81 is connected to one or more processing units, such as servers, via the interconnection medium 82. Interconnection medium 82 may be any of the various interconnection mediums described above and may include a bus type connection such as a PCI or PCI-X bus, a switch fabric, or other interconnects.

Alternatively, the security accelerator may be placed between a server NIC and the rest of the server. In this case the server NIC would pass packets to the security accelerator and the security accelerator would perform the security measures. This embodiment may be desirable for servers with more than one server NIC. It may be particularly desirable to utilized the security accelerator to manage the access control lists of a server. Servers often utilized access control lists to allow or disallow access to all files in the file systems and every file request may require a look up in the control list. Offloading the access control list management to the security accelerator will free up the other processing and memory resources of the server.

Another network endpoint system that the security techniques described herein may be utilized in is a network filer appliance. A network filer is a device that serves only a specific kind of request, such as a file requests. A filer appliance is essentially a server that has been narrowed in features in order to improve performance. It is key for a filer appliance to maximize performance. The security techniques described herein can further improve the performance of a filer appliance by offloading the access control list and ACL lookups from the other resources of the filer appliance. Freeing up CPU bandwidth and memory usage on the filer appliance results in improved performance.

As mentioned above, the intelligent security hardware / security accelerator described herein may be incorporated into a network switch or router. The switch may include a network processor that is programmed to switch packets within the network switch. This network processor or an additional network processor or other hardware may be programmed to operate as the security accelerator. Thus, an external network may be connected on one side of the switch or router and security may be provided to network nodes or endpoints connected to the other side of the switch or router. In one embodiment, the network switch may be connected to multiple clients and provide security for those clients. Programmability of the intelligent security hardware may allow the security protection to be tuned specifically to the types of attacks that may be directed at those clients. In addition, although network switches may support packet filtering, the filter size and complexity may be limited so as not to add latency to a switched packet. The techniques described herein would allow the network switch to support larger, more complex filters. Thus, full firewall functionality may be provided within the network switch.

It will be understood with benefit of this disclosure that although specific exemplary embodiments of hardware and software have been described herein, other combinations of hardware and/or software may be employed to achieve one or more features of the disclosed systems and methods. Furthermore, it will be understood that operating environment and application code may be modified as necessary to implement one or more aspects of the disclosed technology, and that the disclosed systems and methods may be implemented using other hardware models as well as in environments where the application and operating system code may be controlled.

WHAT IS CLAIMED IS:

1. A network processing system connected to a network that carries data in packet format, comprising:

a security accelerator having a processor programmed to receive packets from the network and to examine each packet to determine whether data in the packet represents a potential security violation;

at least one processing unit programmed to respond to requests contained within the packets; and

an interconnection medium for directly connecting the security accelerator to the processing units.

2. The system of Claim 1, wherein the processor is a network processor.

3. The system of Claim 1, wherein the processor is a CPU processor.

4. The system of Claim 1, wherein the security accelerator further has hardware logic operable in conjunction with the processor.

5. The system of Claim 1, wherein the interconnection medium is a bus.

6. The system of Claim 1, wherein the interconnection medium is a switch fabric.

7. The system of Claim 1, wherein the interconnection medium is shared memory.

8. The system of Claim 1, wherein the network is the Internet.

9. The system of Claim 1, wherein the network is a private network.

10. The system of Claim 1, wherein the security accelerator determines a potential security violation by determining whether a packet is part of a denial of service attack.

11. The system of Claim 1, wherein the security accelerator determines a potential security violation by determining whether a packet is part of a syn attack.

12. The system of Claim 1, wherein the security accelerator is further programmed to determine whether outgoing data is authorized to be sent.

13. The system of Claim 1, wherein the network processing system is an endpoint system.

14. The system of Claim 13, wherein the endpoint system is a content delivery system.

15. The system of Claim 1, wherein the network processing system is a client system.

16. The system of Claim 1, wherein the network processing system is a single chassis system.

17. A method for processing network data at a network processing system that receives packet data via a network, comprising the steps of:

using a security accelerator having a processor to receive packets from the network and to examine each packet to determine whether data in the packet represents a potential security violation;

using at least one processing unit to respond to requests contained within the packets; and

directly connecting the security accelerator to the processing unit via an interconnection medium.

18. The method of Claim 17, wherein the processor is a network processor.

19. The method of Claim 17, wherein the processor is a CPU processor.

20. The method of Claim 17, wherein the security accelerator further has hardware logic operable in conjunction with the processor.

21. The method of Claim 17, wherein the interconnection medium is a bus.

22. The method of Claim 17, wherein the interconnection medium is a switch fabric.

23. The method of Claim 17, wherein the interconnection medium is shared memory.

24. The method of Claim 17, wherein the network is the Internet.

25. The method of Claim 17, wherein the network is a private network.

26. The method of Claim 17, wherein the security accelerator determines a potential security violation by determining whether a packet is part of a denial of service attack.

27. The method of Claim 17, wherein the security accelerator determines a potential security violation by determining whether a packet is part of a syn attack.

28. The method of Claim 17, wherein the security accelerator is further programmed to determine whether outgoing data is authorized to be sent.

29. The method of Claim 17, wherein the network processing system is an endpoint system.

30. The method of Claim 17, wherein the endpoint system is a content delivery system.

31. The method of Claim 17, wherein the network processing system is a client system.

32. A security accelerator device for use at a network node, comprising:

at least one processor programmed to receive packets from the network and to examine each packet to determine whether data in the packet represents a potential security violation;

an front end interface for connecting the security accelerator to a network; and

a back end interface for connecting the security accelerator to an interconnection medium.

33. The device of Claim 32, wherein the processor is a network processor.

34. The device of Claim 32, wherein the processor is a CPU processor.

35. The device of Claim 32, wherein the security accelerator further has hardware logic operable in conjunction with the processor.

36. The device of Claim 32, wherein the interconnection medium is a bus.

37. The device of Claim 32, wherein the interconnection medium is a switch fabric.

38. The device of Claim 32, wherein the interconnection medium is shared memory.

39. The device of Claim 32, wherein the security accelerator, the front end interface, and the back end interface are fabricated as a single circuit component.

40. A network connectable computing system providing at least some security functions in addition to system functionality, the system being configured to be connected on at least one end to a network, the system comprising:

at least one network connection configured to be coupled to the network;

at least one system processor for performing system functionality;

security hardware located in a data path between the network connection and the at least one processor; and

an interconnection between the at least one processor and the security hardware,

wherein the security hardware off-loads at least some security functions from other system resources by analyzes data packets entering the network connectable computing system to perform security functions prior to forwarding the data packets to the remainder of the system.

41. The system of claim 40 wherein the security hardware comprises a network processor.

42. The system of claim 41 wherein the security functions are programmable.

43. The system of claim 41, wherein the analysis of data packets comprises analyzing data packet headers.

44. The system of claim 43, wherein the at least one system processor and the network processor communicate in a peer to peer environment across a distributed interconnect.

45. The system of claim 44, wherein the at least one system processor comprises at least one storage processor and at least one application processor.

46. The system of claim 40, wherein the network connectable computing system is a network endpoint system and the at least one system processor comprises at least one storage processor and at least one application processor.

47. The system of claim 46, wherein the interconnection is a switch fabric.

48. A method of operating a network connectable computing system, comprising:
receiving data from a network;
analyzing the data with programmable security hardware to decode incoming data packet headers;
performing at least one security function based upon the analysis of the data packet header; and
forwarding the data packet to at least one system processor through a system interconnection after performing the at least one function.

49. The method of claim 48 wherein the security function is to determine if the data is part of a security attack.

50. The method of claim 48 wherein the security function is a filtering operation.

51. The method of claim 48 wherein the security function is an authentication verification or access control list function.

52. The method of claim 48, further comprising performing security functions on outgoing data packets to provide bi-directional security functionality.

53. The method of claim 48 wherein the security function is performed by a network processor.

54. The method of claim 53 wherein the network connectable computing system is a network endpoint system.

55. The method of claim 54 wherein the network processor is programmable to allow the implementation of different security algorithms.

56. A network endpoint system for performing endpoint functionality, the endpoint system comprising:

at least one system processor, the system processor performing endpoint processing functionality;

a distributed interconnect coupled to the at least one system processor; and

security hardware coupled to the distributed interconnect,

wherein the system is configured such that a data packet from a network may be processed by the security hardware prior to being processed by the at least one system processor, and

wherein the security hardware is configured to process at least a portion of the data packet to perform a security function prior to the security hardware forwarding the data packet to the distributed interconnect.

57. The network endpoint system of claim 56, wherein the security hardware is programmable so that different security algorithms may be implemented in the security hardware.

58. The network endpoint system of claim 56, wherein the at least one system processor comprises at least one storage processor and at least one application processor.

59. The network endpoint system of claim 58, wherein the security hardware comprises at least one network processor.

60. The network endpoint system of claim 59, wherein the network processor, the storage processor and the application processor operate in a peer to peer environment across the distributed interconnect.

61. The network endpoint system of claim 60, wherein the distributed interconnect is a switch fabric.

62. The network endpoint system of claim 56, wherein the network endpoint system is a content delivery system.

63. The network endpoint system of claim 62 wherein:

the security hardware comprises at least one network processor;

the at least one system processor comprises at least one storage processor and at least one application processor, the storage processor being configured to interface with a storage system; and

the network processor, the storage processor and the application processor operate in a peer to peer environment across the distributed interconnect.

64. The network endpoint system of claim 63 wherein the distributed interconnect is a switch fabric.

65. The network endpoint system of claim 64, wherein the system is configured in a single chassis.

66. A method of operating a network endpoint system, comprising:

providing a network processor within the network endpoint system, the network processor being at an interface which couples the network endpoint system to a network;

processing data passing through the interface with the network processor;

performing security functions as part of the processing of the network processor; and

forwarding incoming network data from the network processor to a system processor which performs at least some endpoint functionality upon the data.

67. The method of claim 66 wherein the network processor rejects incoming network data which violates security algorithms and forwards incoming data which passes security algorithms to the system processor.

68. The method of claim 66 wherein the network processor analyzes headers of data packets to perform the security functions.

69. The method of claim 68 wherein the network processor is programmable to implement different security algorithms.

70. The method of claim 68 wherein the security functions include detecting a security
70.

71. The method of claim 70 wherein the security attack comprises a denial of service attack.

72. The method of claim 68 wherein the security function is a filtering operation.

73. The method of claim 68 wherein the security function is an authentication verification or access control list function.

74. The method of claim 68, wherein the security function is performed upon outgoing data.

75. The method of claim 74 wherein the security function comprises performing security functions upon both outgoing and incoming data.

76. A network connectable computing system, comprising:
a first connection to receive data packets from a network;
security hardware comprising at least one network processor, the security hardware coupled to the interface connection; and
a second connection to transmit data processed by the security hardware,
wherein the at least one network processor analyzes at least a portion of the data packets to perform at least one security function.

77. The system of claim 76, wherein the network processor analyzes headers of the data packets.

78. The system of claim 76, wherein the system is an intermediate network node system.

79. The system of claim 78, wherein the system is a network switch.

80. The system of claim 76, wherein the system is a network endpoint system.

81. The system of claim 76, wherein the system is a network endpoint system having at least one server or at least one server card coupled to the second connection.

82. The system of claim 76, wherein the system is incorporated into a network interface card.

83. The system of claim 81, wherein the second connection is a distributed interconnection.

84. The system of claim 83, wherein the distributed interconnection is a switch fabric.

85. The system of claim 76, wherein the second connection is coupled to an asymmetric multi-processing system.

86. The system of claim 85, wherein the second connection is a distributed interconnection and the asymmetric multi-processing system includes a plurality of task specific processors.

87. The system of claim 86, wherein the distributed interconnection is a switch fabric and the task specific processors include storage or application processors.

88. The system of claim 87, wherein the task specific processors include storage and application processors.

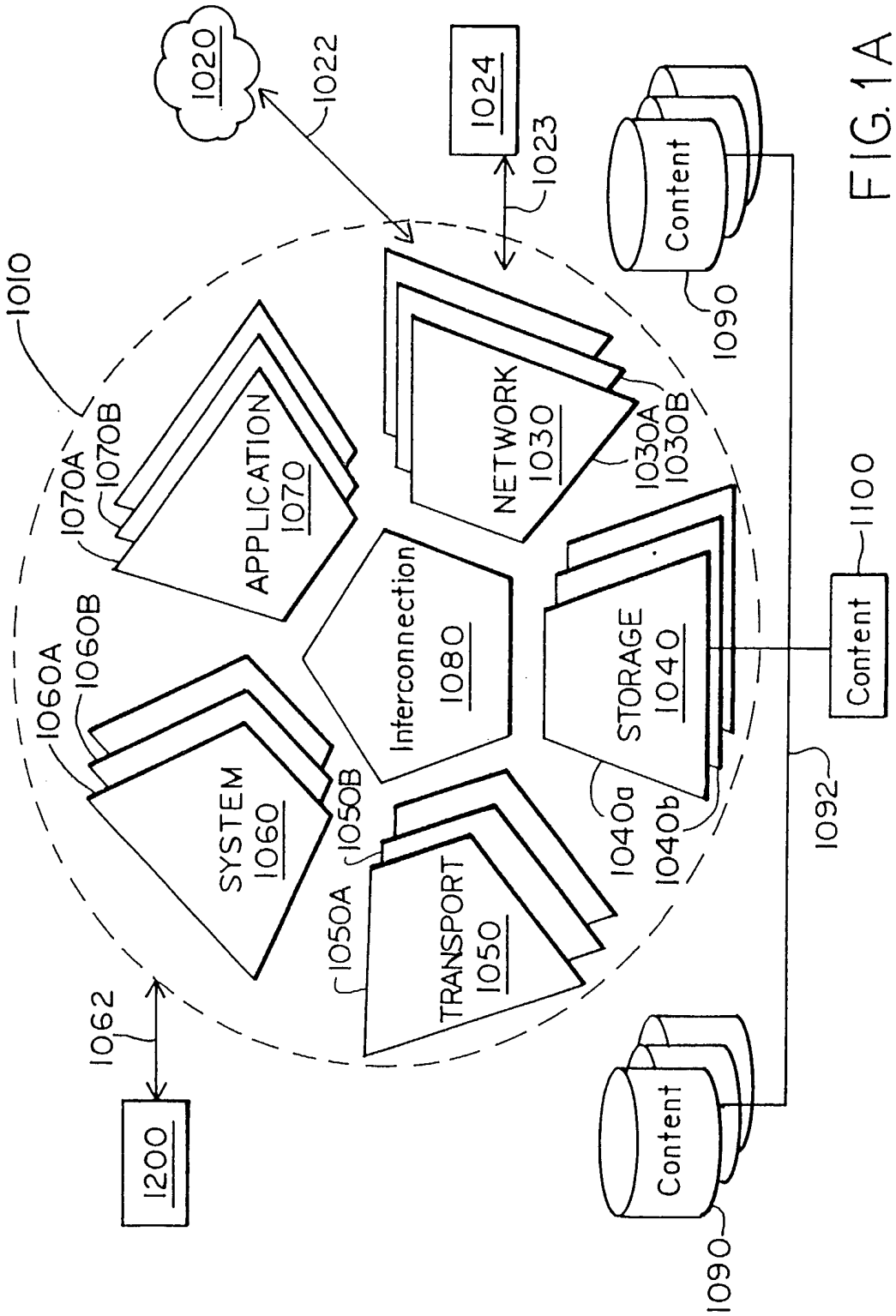

1020
1022
1024
1023
1010
1070A
1070B
APPLICATION
1070
NETWORK
1030
1030A
1030B
Content
1090
1060A
1060B
SYSTEM
1060
Interconnection
1080
STORAGE
1040
1100
Content
1050B
1050A
TRANSPORT
1050
1040a
1040b
1092
1062
1200
Content
1090

FIG. 1A

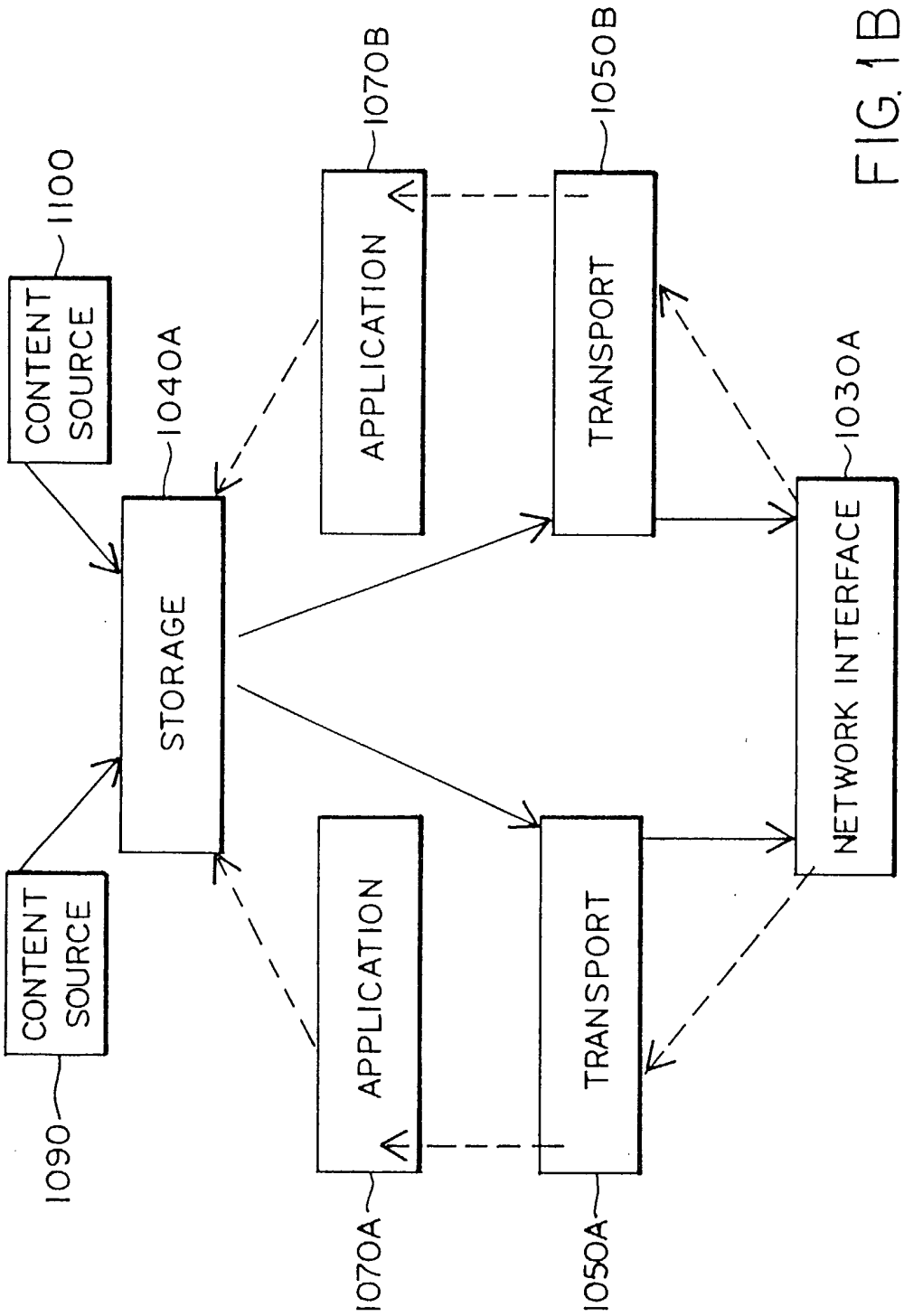

1100
CONTENT SOURCE
1040A
STORAGE
1090
CONTENT SOURCE
1070B
APPLICATION
1050B
TRANSPORT
1030A
NETWORK INTERFACE
1070A
APPLICATION
1050A
TRANSPORT

FIG. 1B

FIG.IC | FIG. IC' | FIG. IC''

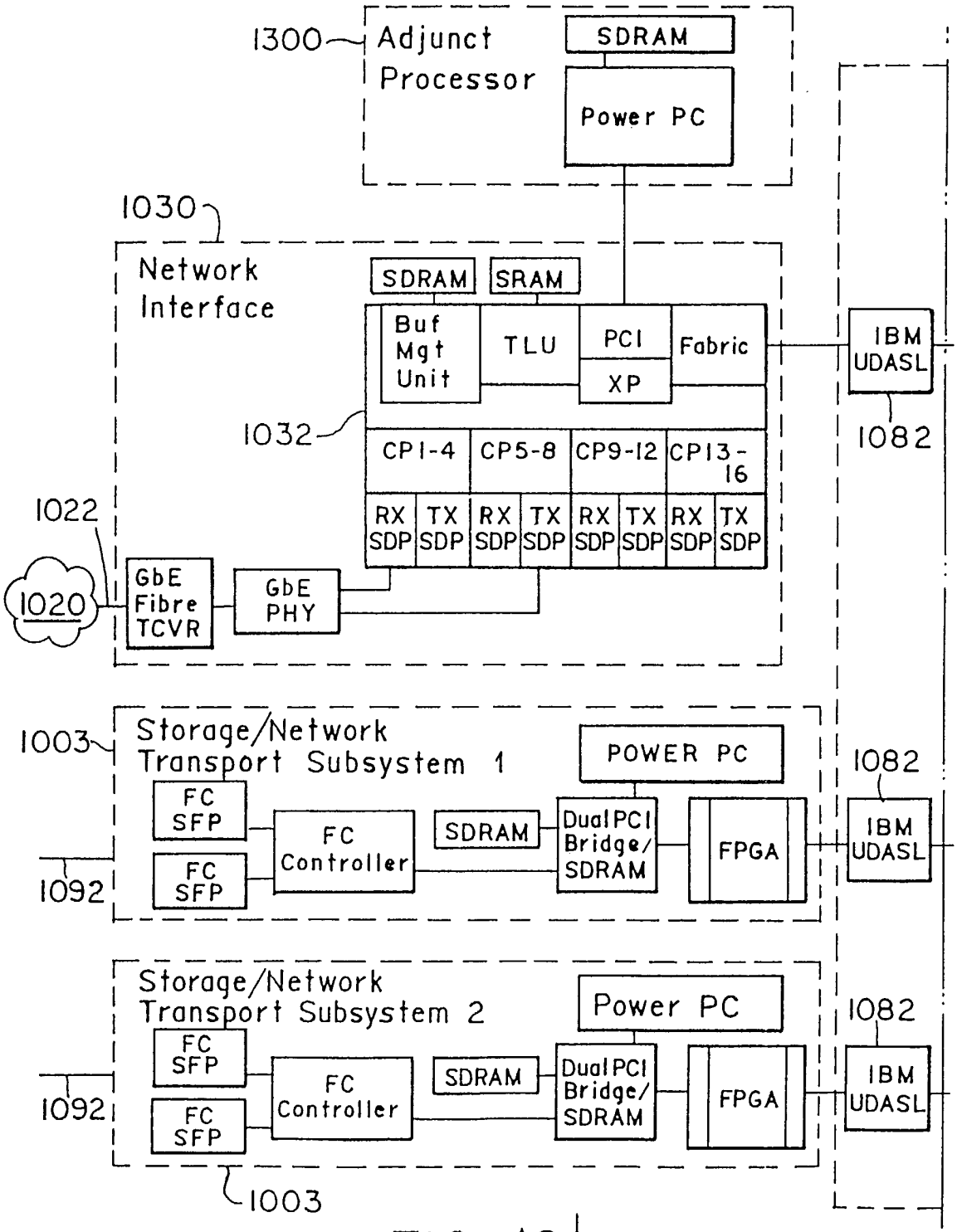


FIG. 1C'

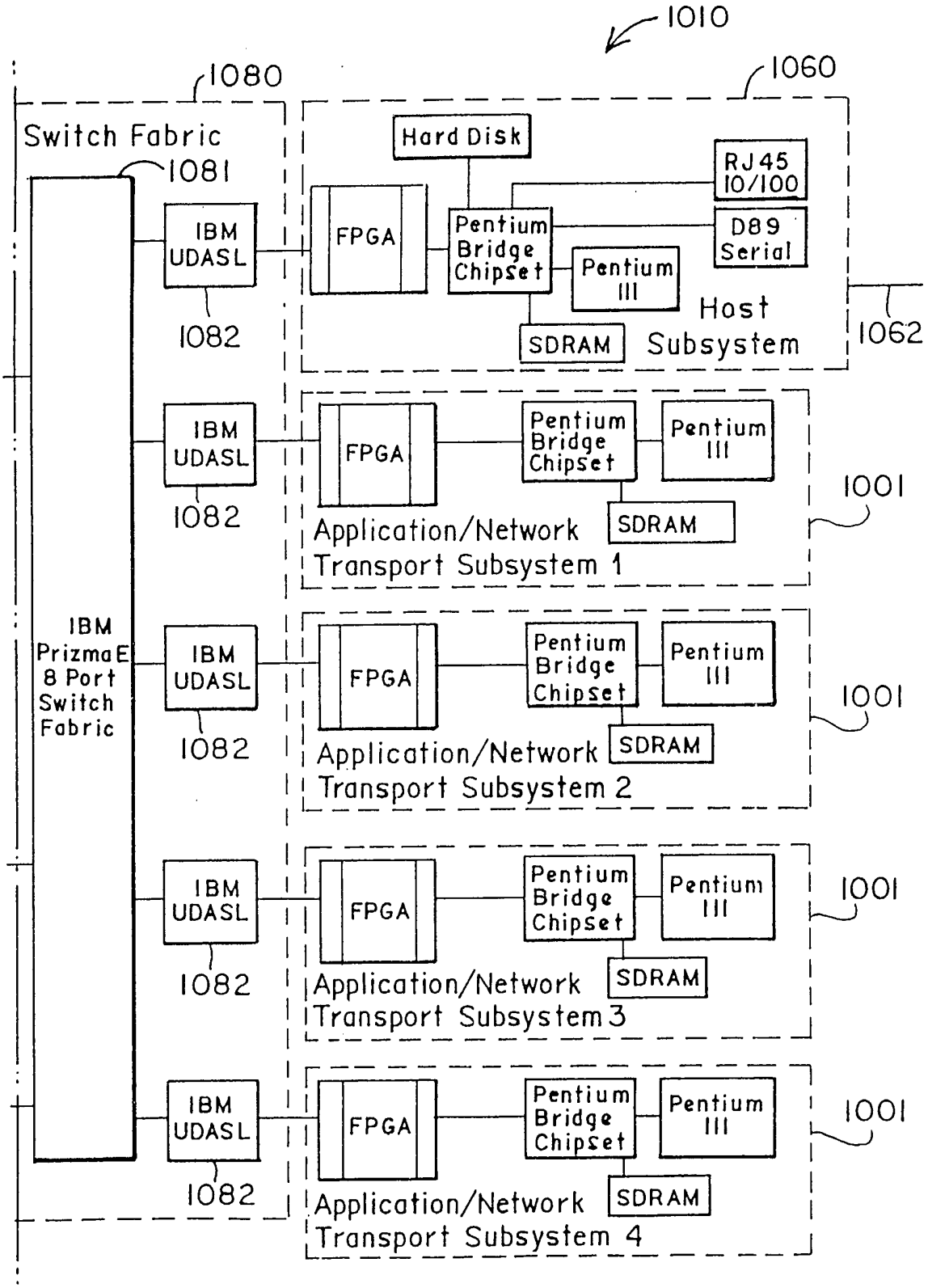

1010
1080
1060
Switch Fabric
1081
Hard Disk
RJ45 10/100
D89 Serial
IBM UDASL
FPGA
Pentium Bridge Chipset
Pentium III
SDRAM
Host Subsystem
1062
1082
Application/Network Transport Subsystem 1
1001
Application/Network Transport Subsystem 2
Application/Network Transport Subsystem 3
Application/Network Transport Subsystem 4
IBM PrizmaE 8 Port Switch Fabric

FIG. 1C[11]

FIG.1D | FIG.1D' | FIG.1D''

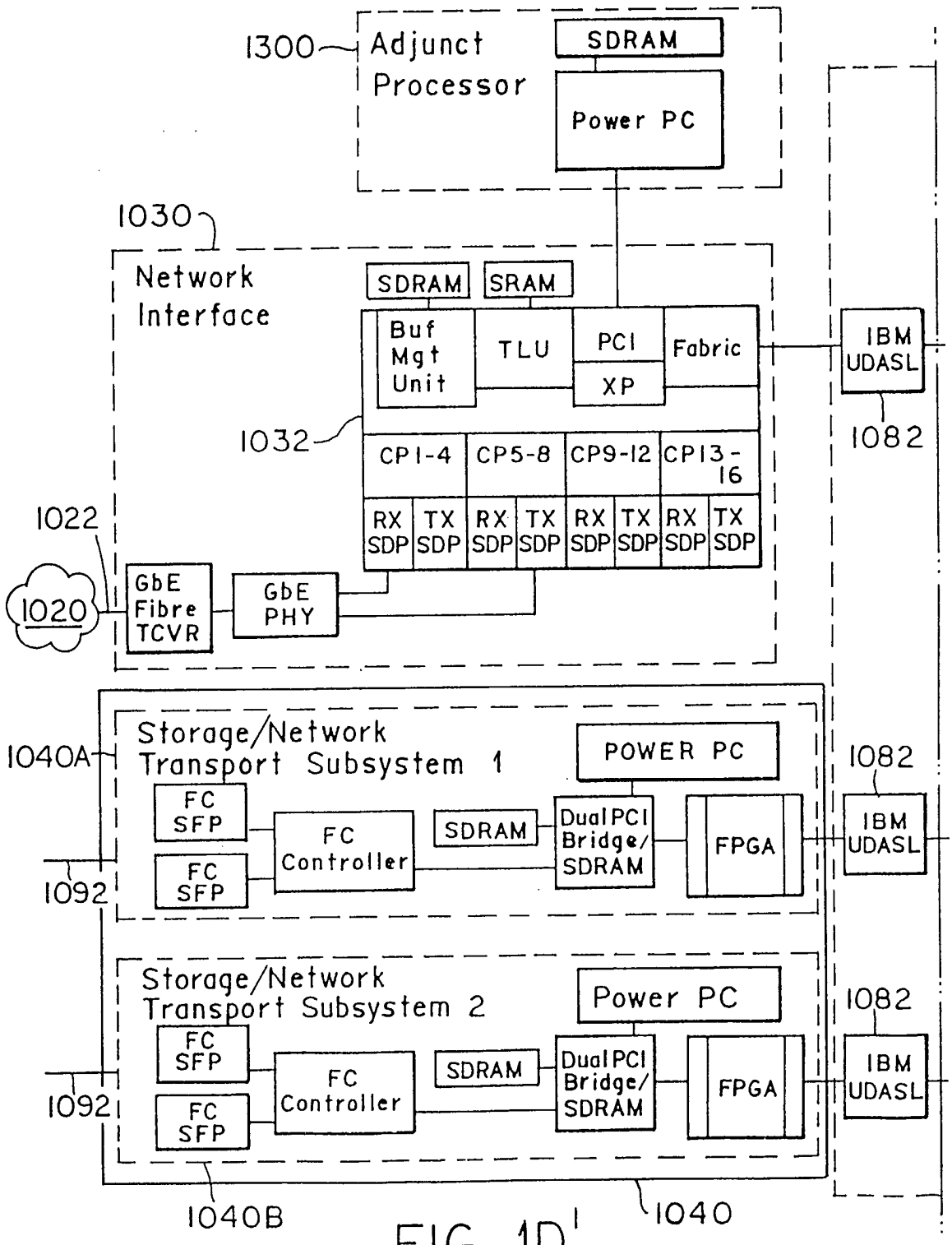


FIG. 1D'

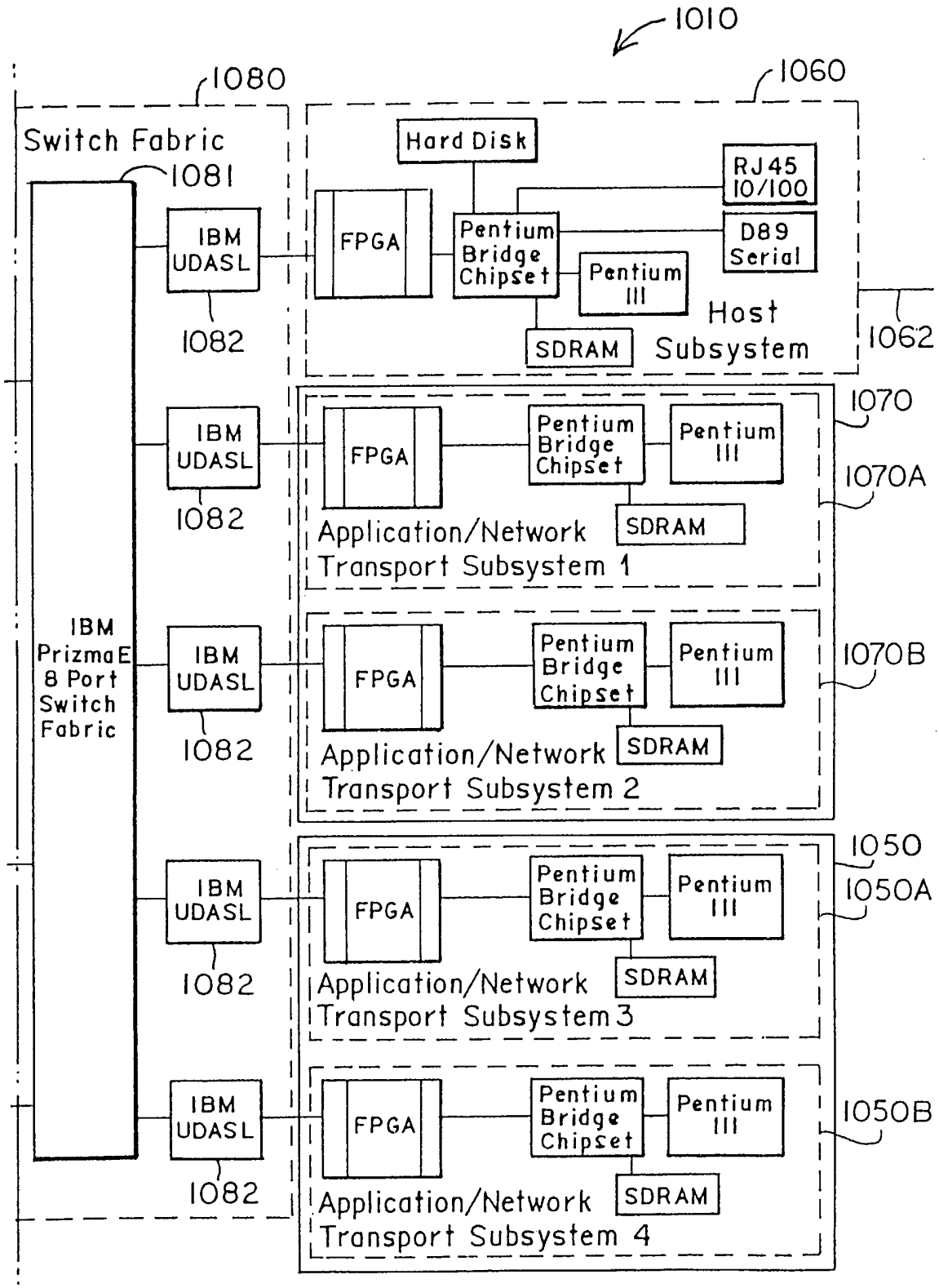

1010
1080
Switch Fabric
1081
IBM PrizmaE 8 Port Switch Fabric
IBM UDASL
1082
1060
Hard Disk
RJ45 10/100
D89 Serial
FPGA
Pentium Bridge Chipset
Pentium III
SDRAM
Host Subsystem
1062
1070
1070A
Application/Network Transport Subsystem 1
1070B
Application/Network Transport Subsystem 2
1050
1050A
Application/Network Transport Subsystem 3
1050B
Application/Network Transport Subsystem 4

FIG. 1D[11]

FIG.1E | FIG.1E' | FIG.1E''

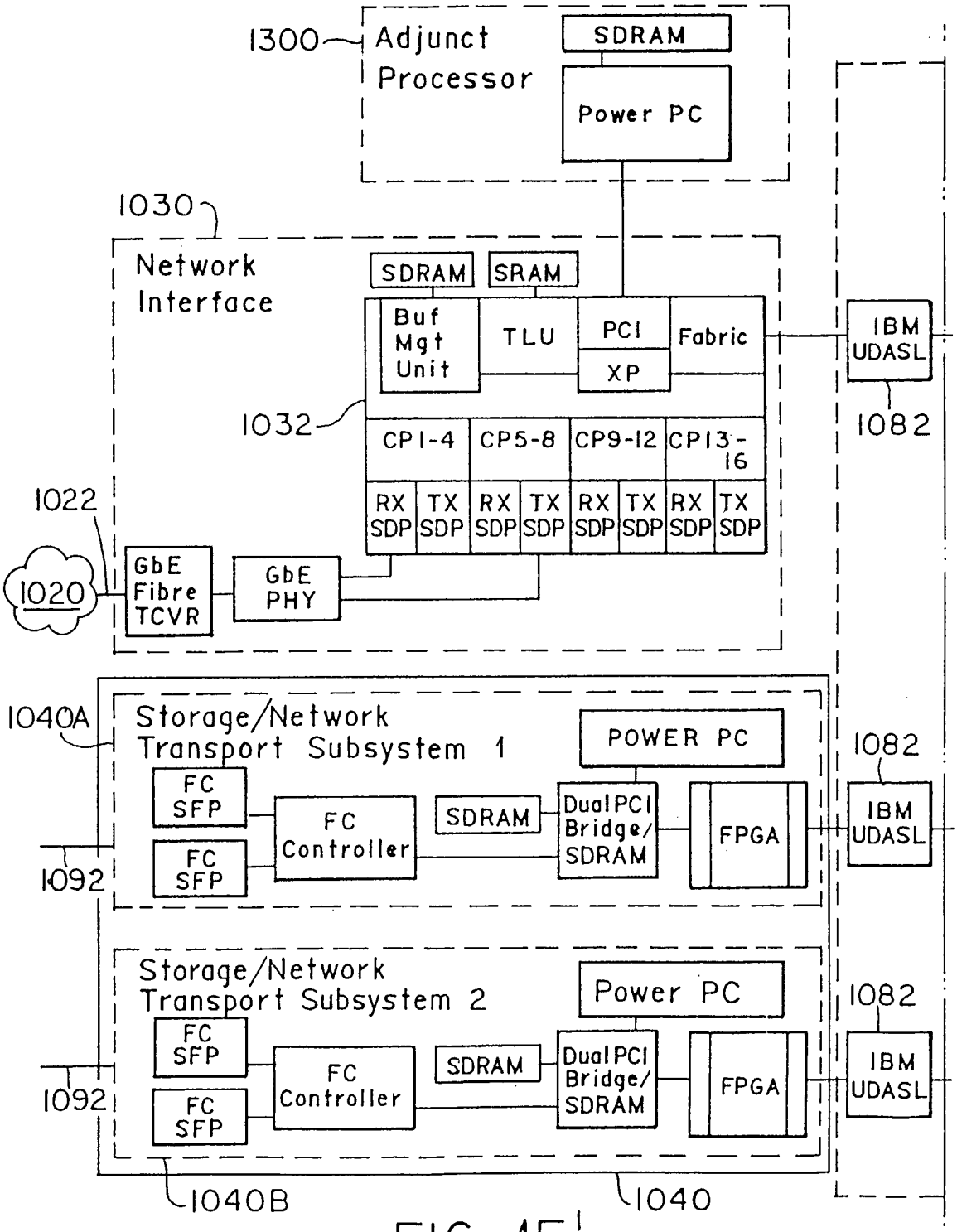


FIG. 1E'

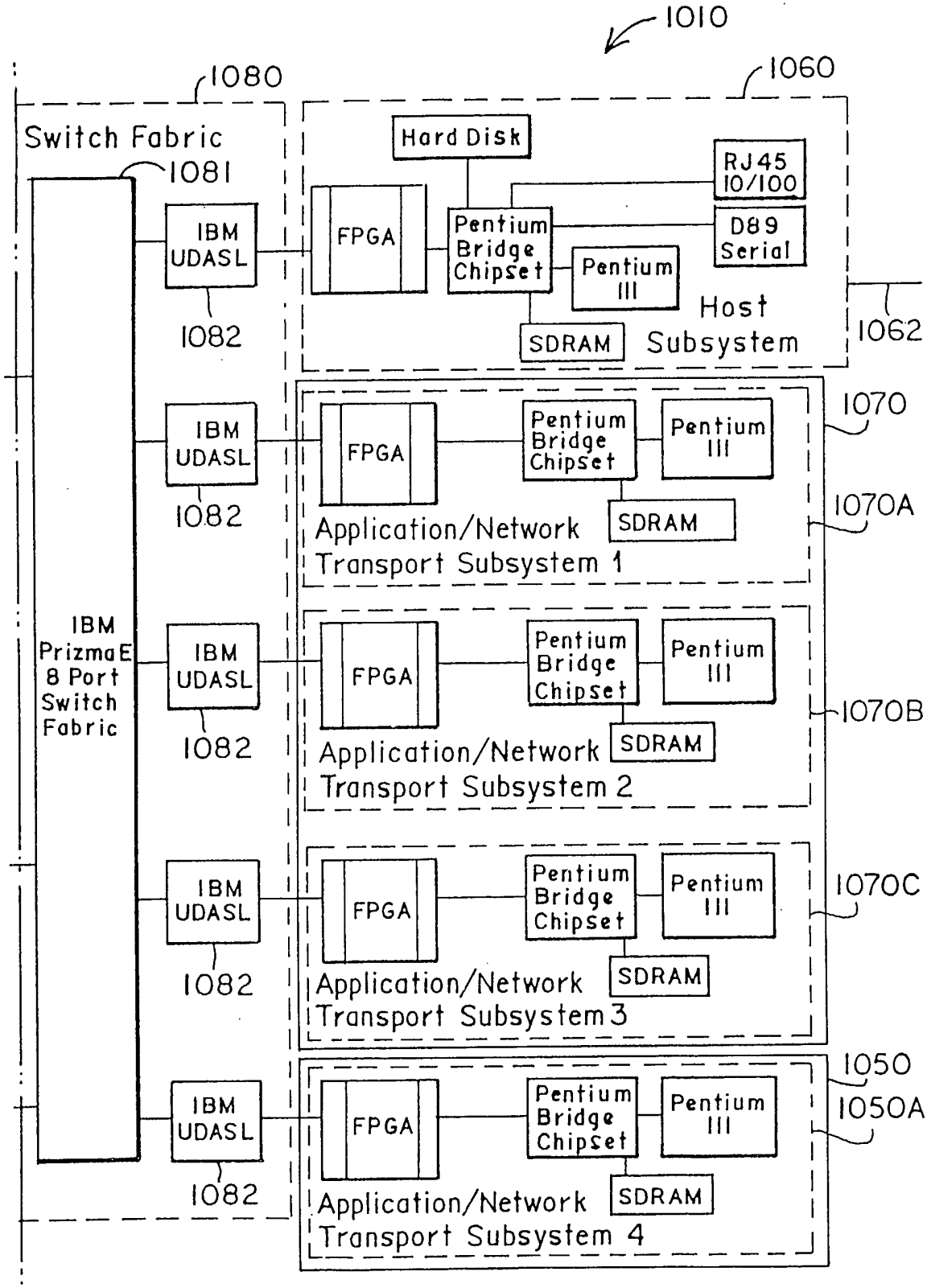

1010
1080
1060
Switch Fabric
1081
Hard Disk
RJ45 10/100
D89 Serial
IBM UDASL
FPGA
Pentium Bridge Chipset
Pentium III
SDRAM
Host Subsystem
1062
1082
1070
1070A
Application/Network Transport Subsystem 1
1070B
Application/Network Transport Subsystem 2
1070C
Application/Network Transport Subsystem 3
1050
1050A
Application/Network Transport Subsystem 4
IBM PrizmaE 8 Port Switch Fabric

FIG. 1E[11]

FIG.1F | FIG.1F' | FIG.1F''

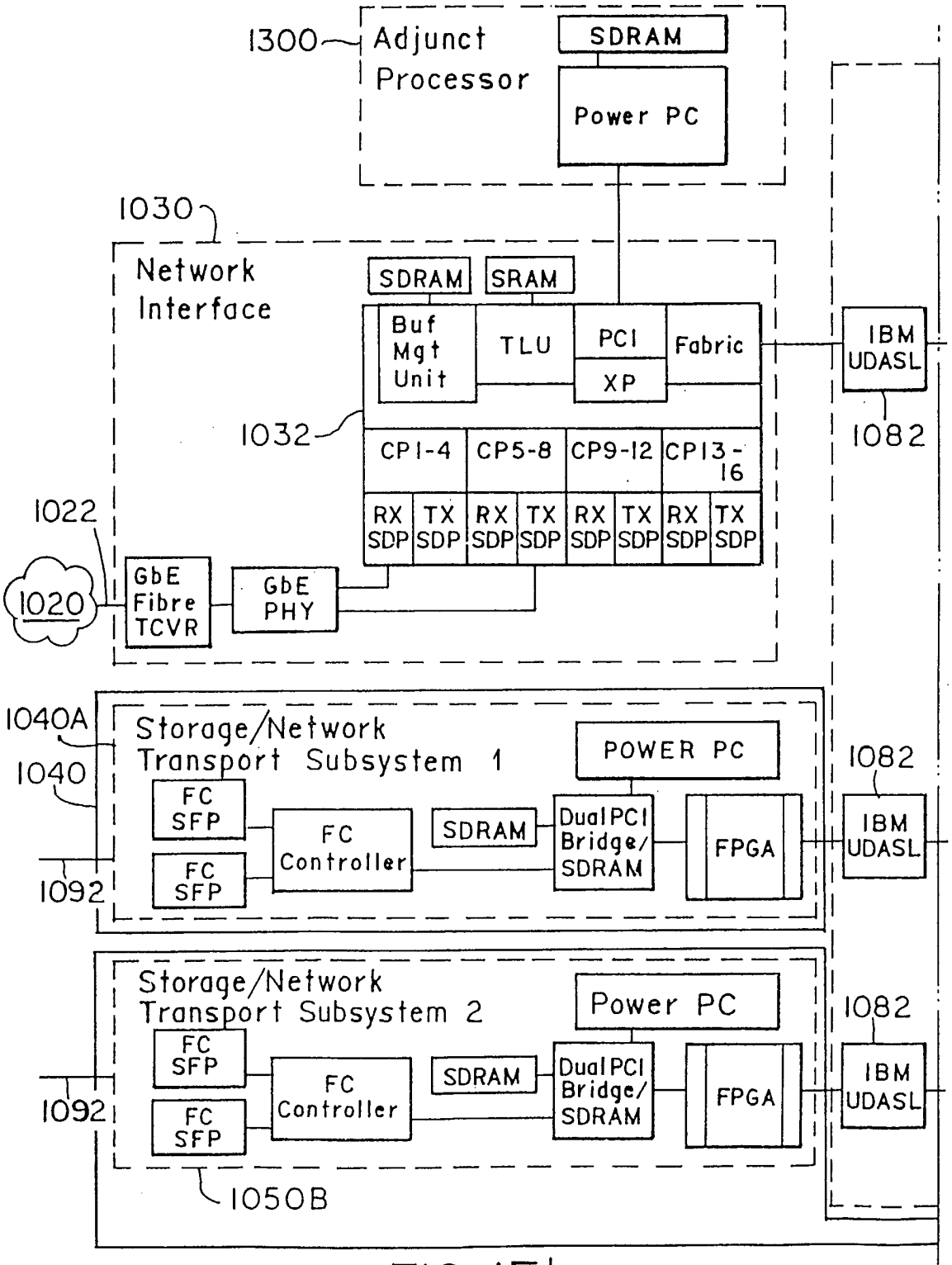


FIG. 1F'

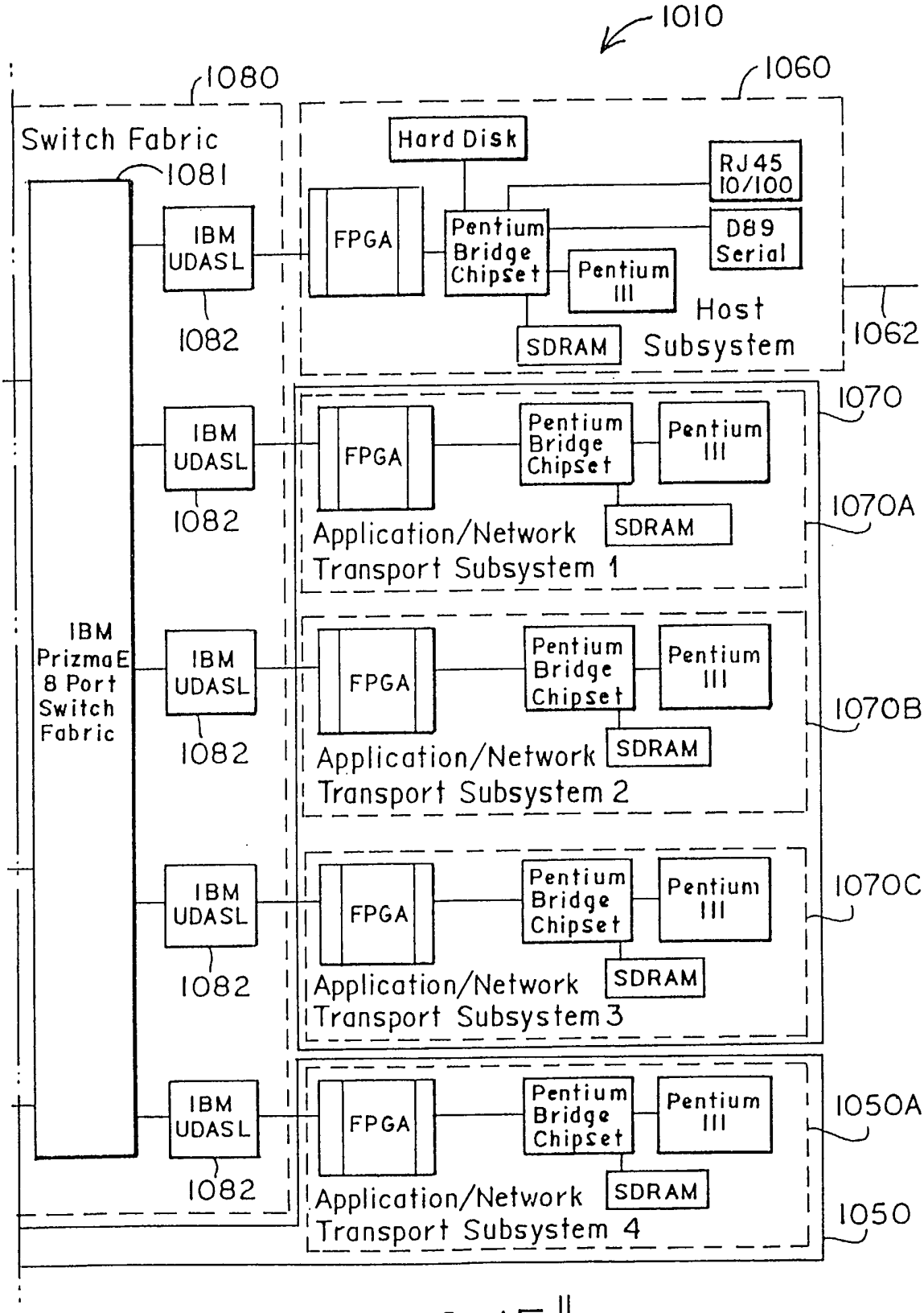

1010
1080
1060
Switch Fabric
1081
Hard Disk
RJ45 10/100
D89 Serial
IBM UDASL
FPGA
Pentium Bridge Chipset
Pentium III
SDRAM
Host Subsystem
1062
1082
1070
1070A
Application/Network Transport Subsystem 1
1070B
Application/Network Transport Subsystem 2
1070C
Application/Network Transport Subsystem 3
1050A
Application/Network Transport Subsystem 4
1050
IBM PrizmaE 8 Port Switch Fabric

FIG. 1F[11]

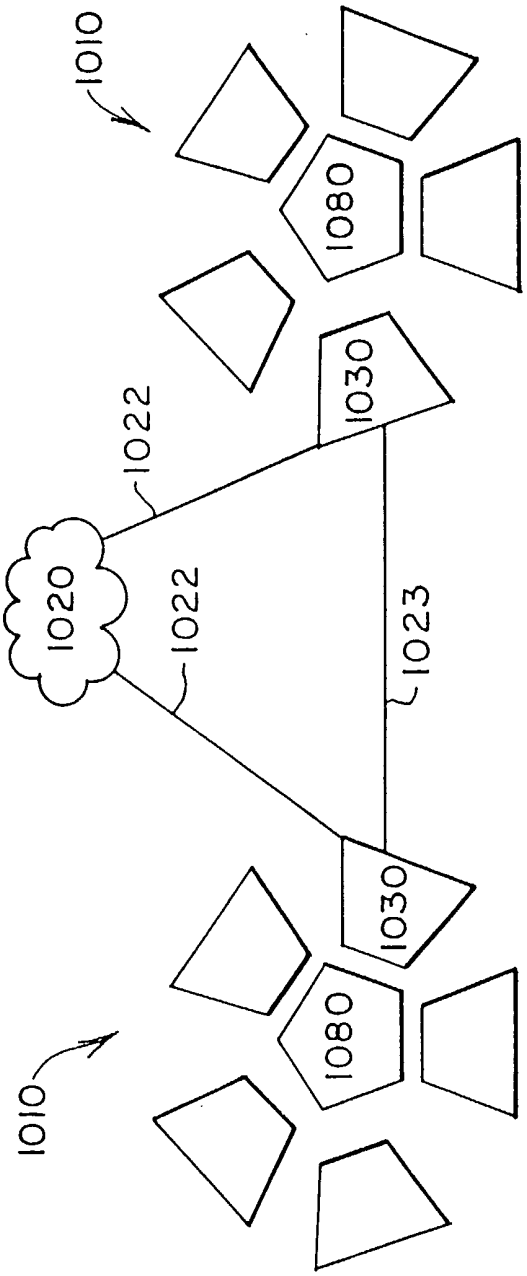


FIG. 1G

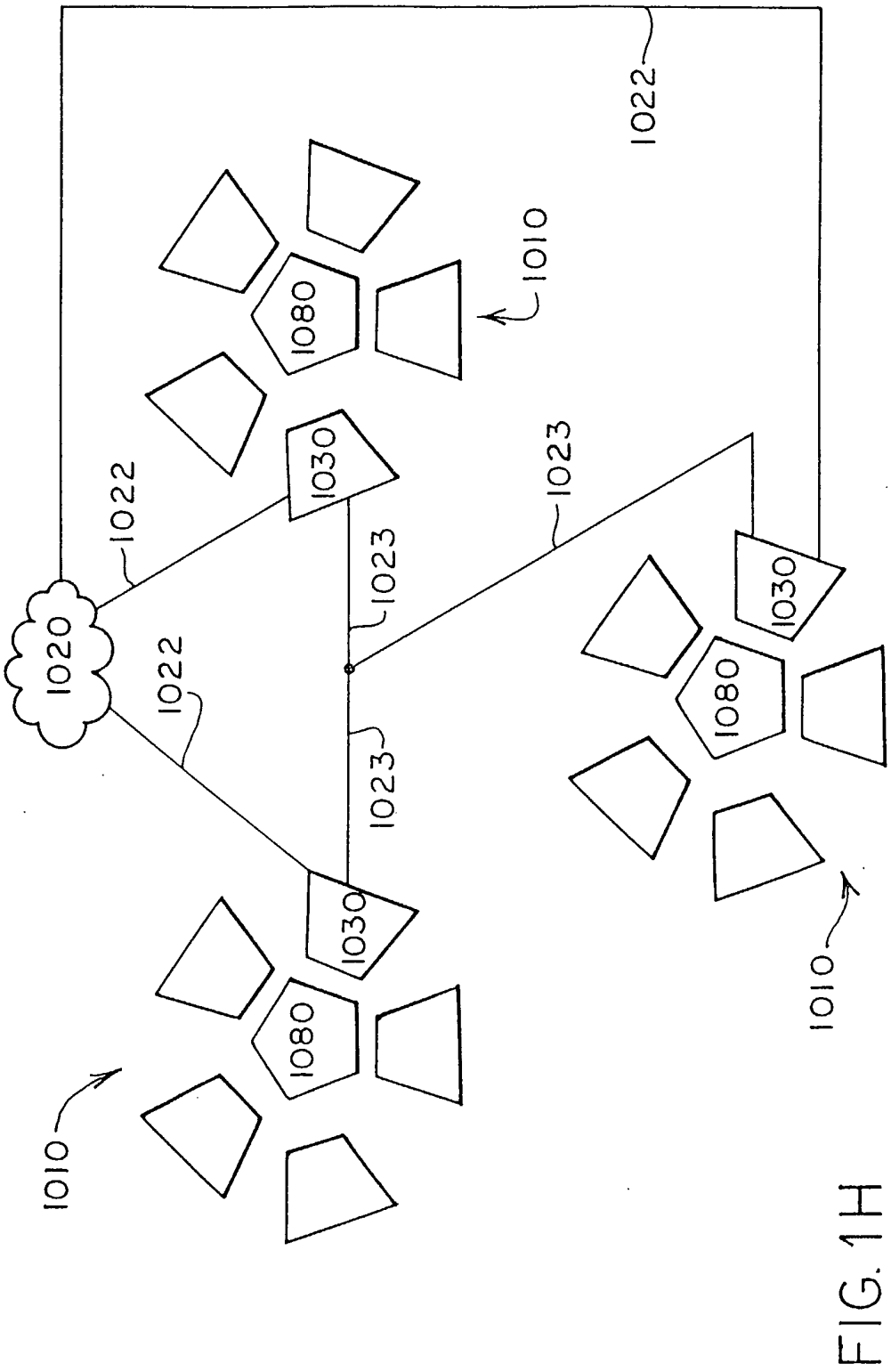

1022
1010
1080
1030
1022
1023
1023
1020
1022
1023
1030
1080
1030
1080
1010
1010

FIG. 1H

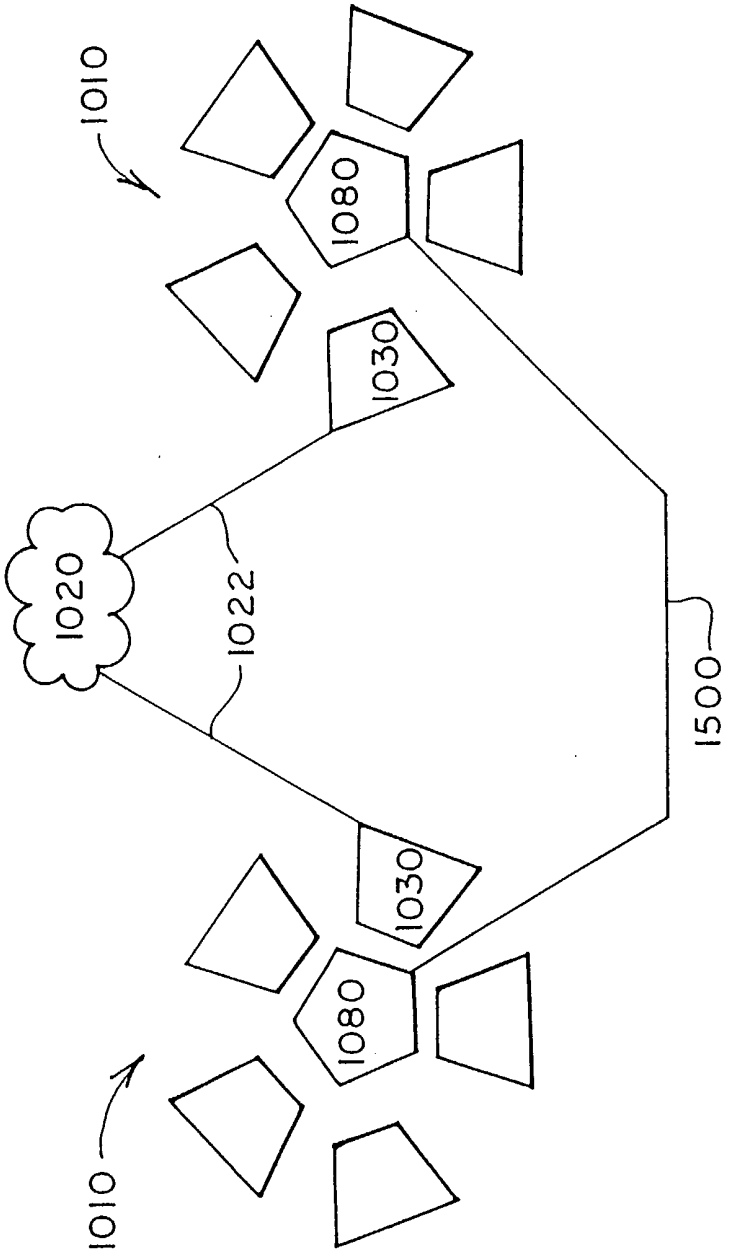


FIG. 1I

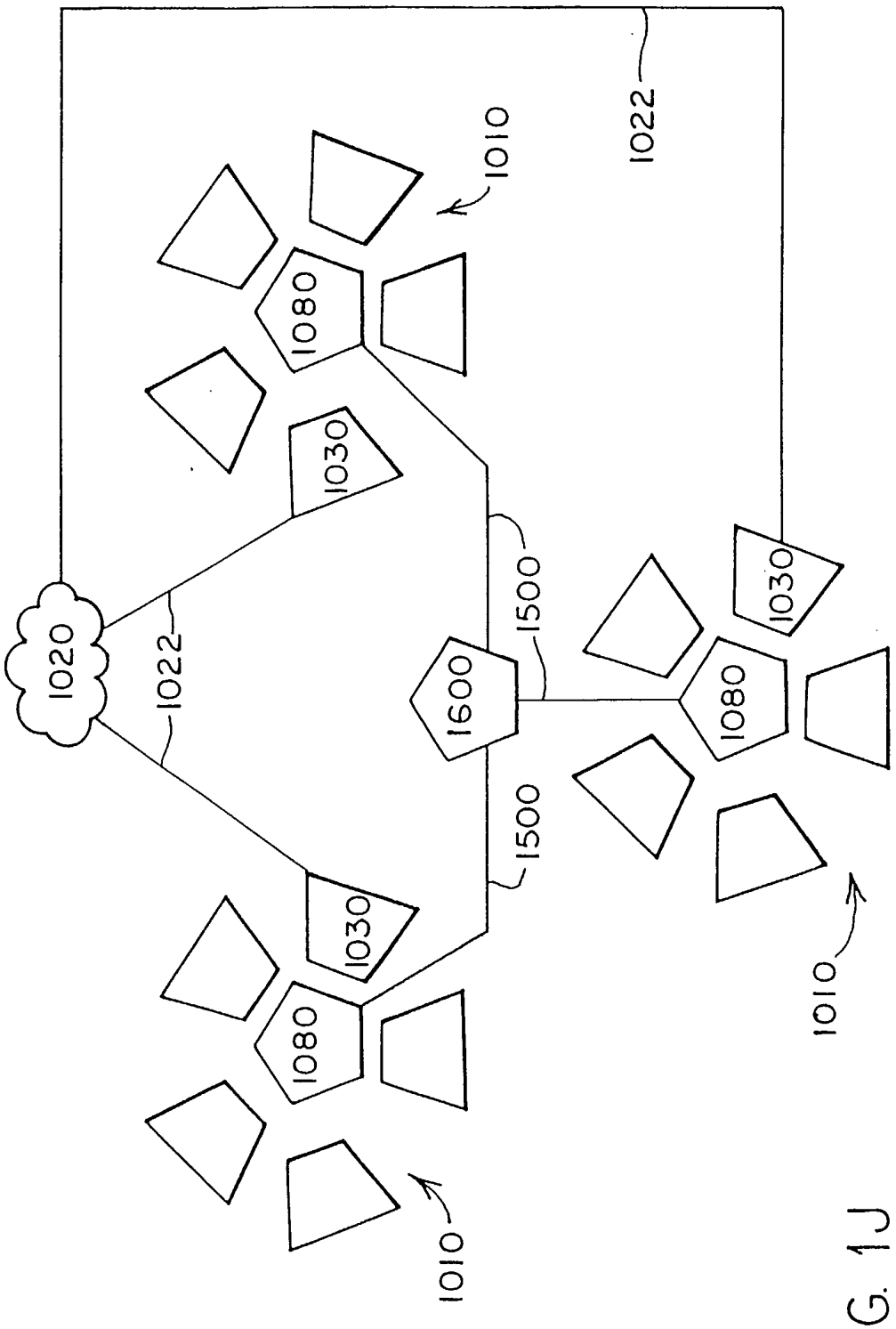

1022
1010
1080
1030
1020
1022
1500
1600
1030
1080
1500
1030
1080
1010
1010

FIG. 1J

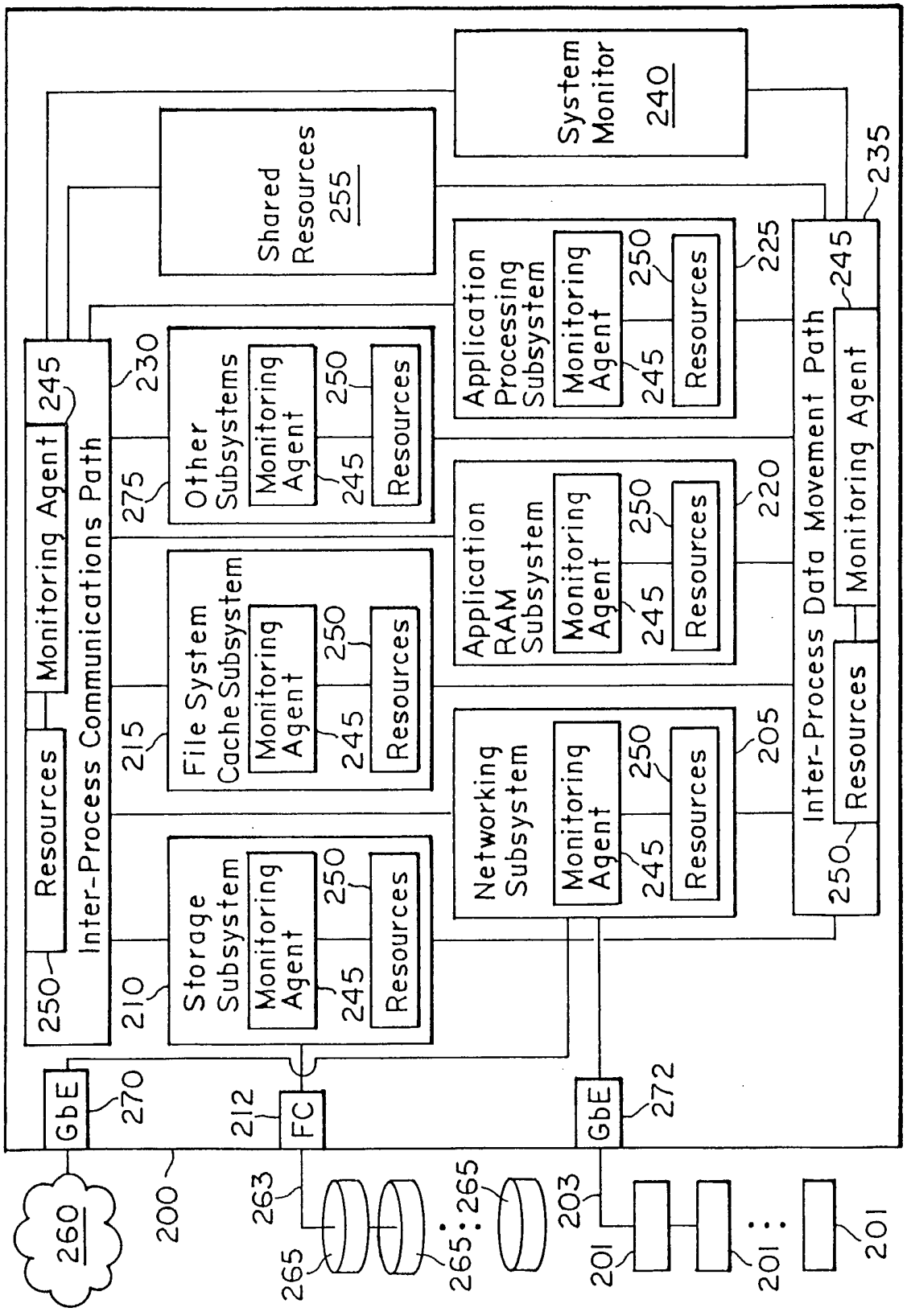

System Monitor 240
Shared Resources 255
Application Processing Subsystem
Monitoring Agent
Resources
225
Other Subsystems
230
275
Application RAM Subsystem
220
File System Cache Subsystem
215
Networking Subsystem
205
Storage Subsystem
210
Inter-Process Communications Path
Inter-Process Data Movement Path
235
245
250
GbE
270
FC
212
272
260
200
263
265
203
201

FIG. 2

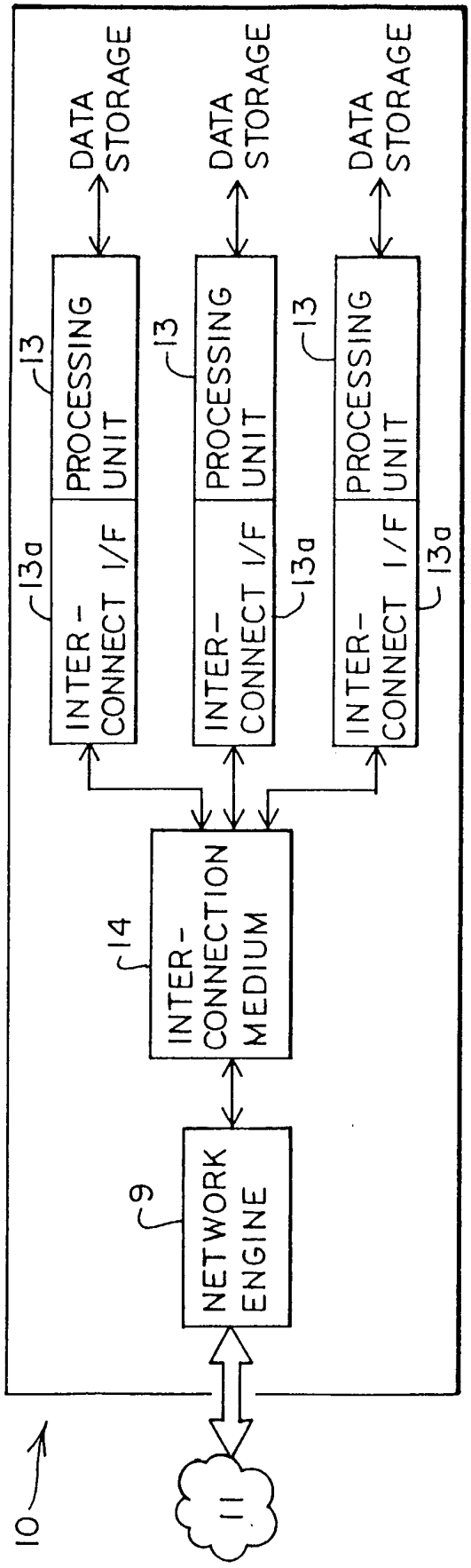

10
11
NETWORK ENGINE
9
INTER-CONNECTION MEDIUM
14
INTER-CONNECT I/F
13a
PROCESSING UNIT
13
DATA STORAGE

FIG. 2A

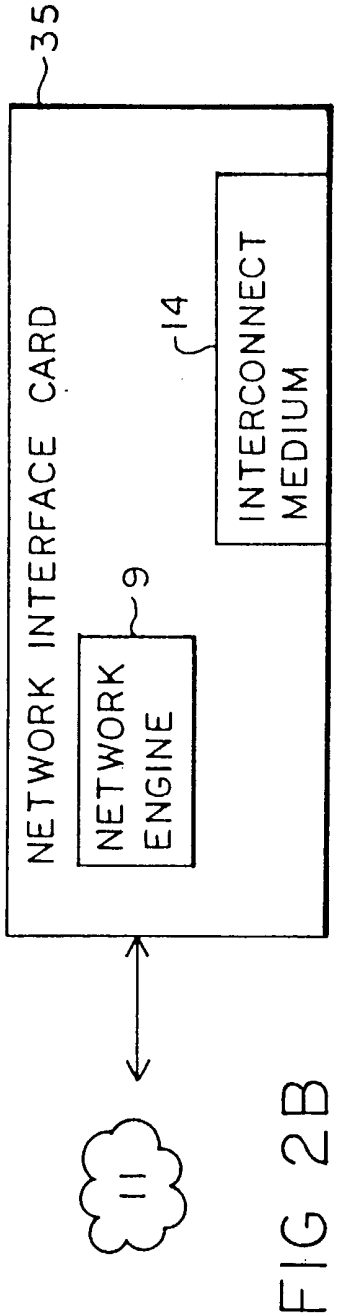

35
NETWORK INTERFACE CARD
NETWORK ENGINE
9
INTERCONNECT MEDIUM
14
11

FIG 2B

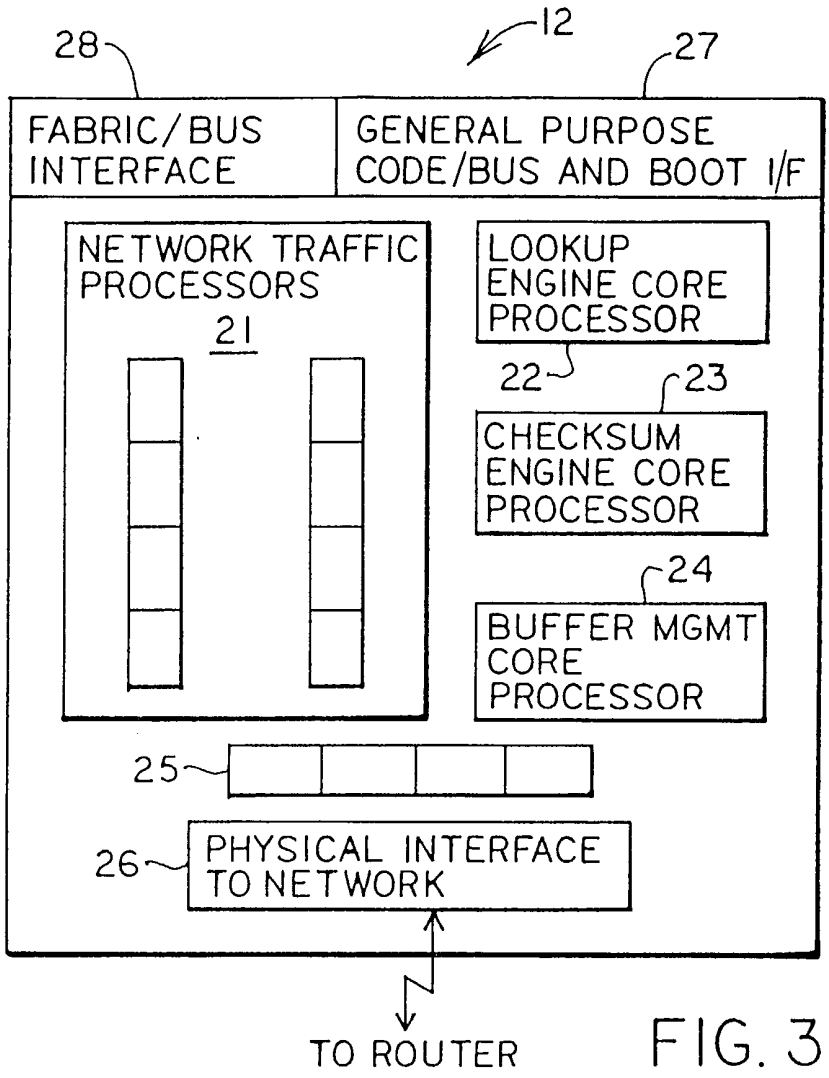

12
28
27
FABRIC/BUS INTERFACE
GENERAL PURPOSE CODE/BUS AND BOOT I/F
NETWORK TRAFFIC PROCESSORS
21
LOOKUP ENGINE CORE PROCESSOR
22
23
CHECKSUM ENGINE CORE PROCESSOR
24
BUFFER MGMT CORE PROCESSOR
25
26
PHYSICAL INTERFACE TO NETWORK
TO ROUTER

FIG. 3

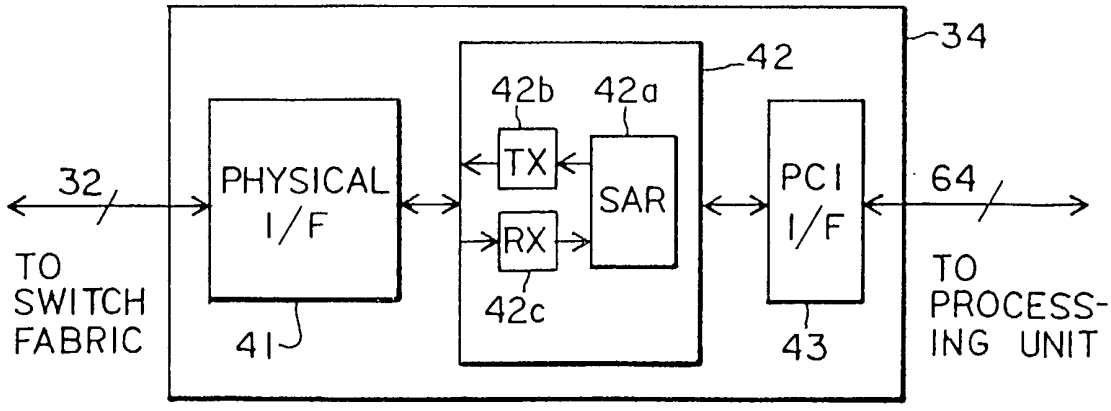

34
42
42b
42a
TX
SAR
RX
42c
32
PHYSICAL I/F
41
PCI I/F
43
64
TO SWITCH FABRIC
TO PROCESSING UNIT

FIG. 4

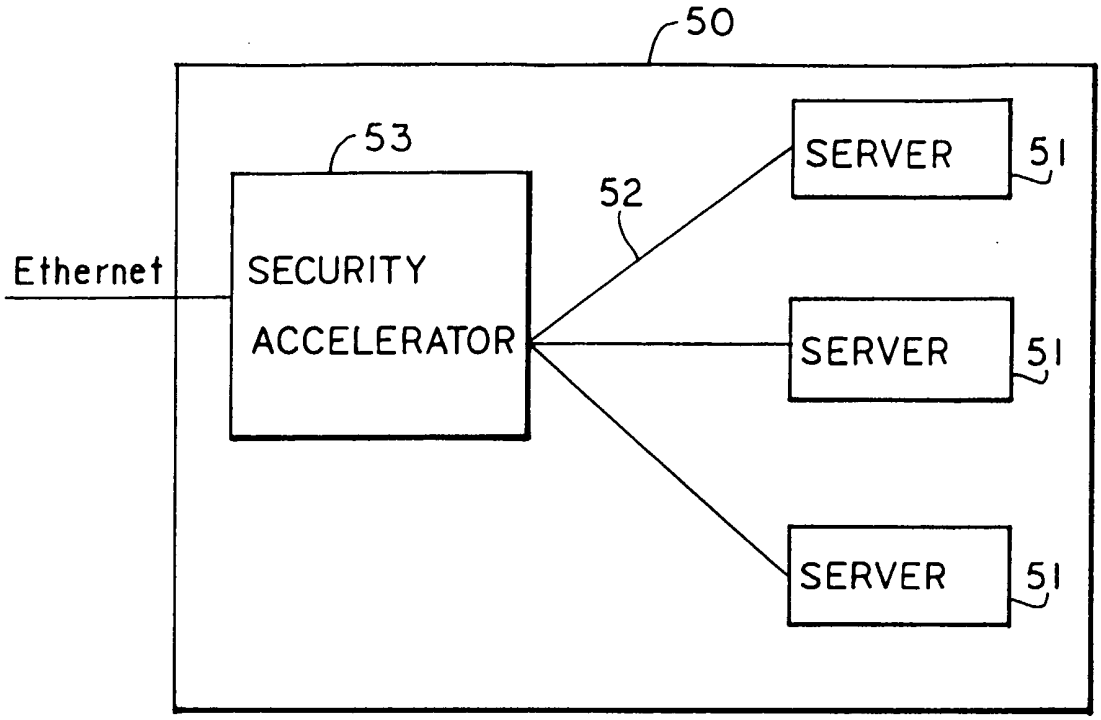

50
53
52
Ethernet
SECURITY
ACCELERATOR
SERVER
51
SERVER
51
SERVER
51

FIG. 5

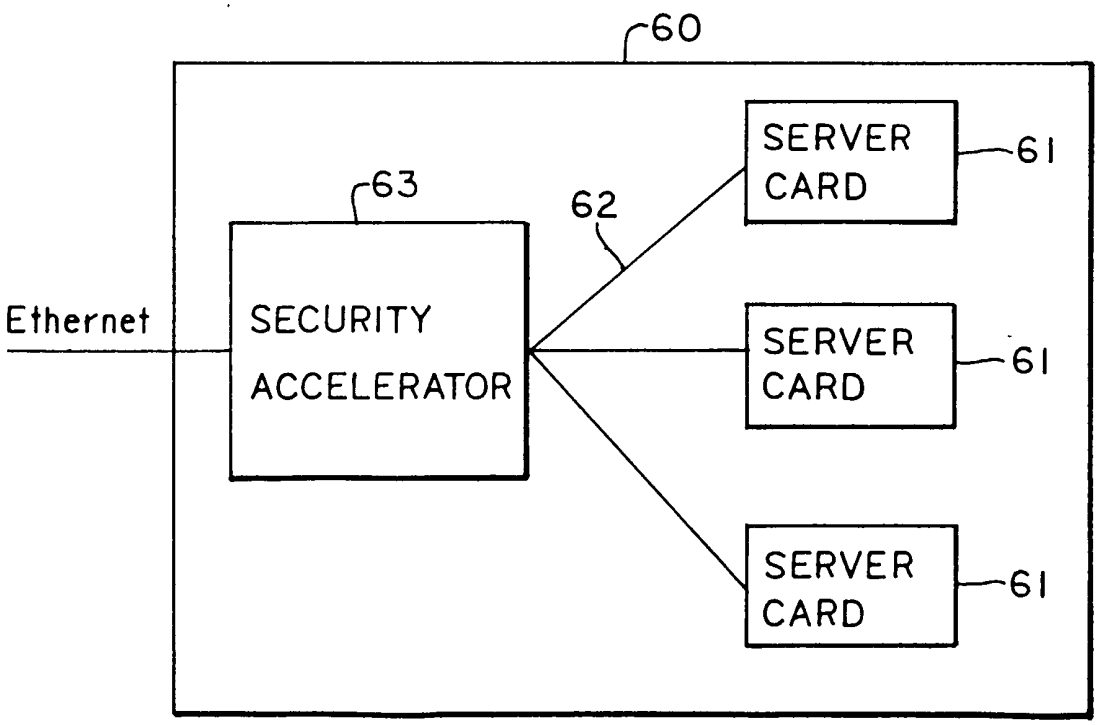

60
63
62
Ethernet
SECURITY
ACCELERATOR
SERVER
CARD
61
SERVER
CARD
61
SERVER
CARD
61

FIG. 6

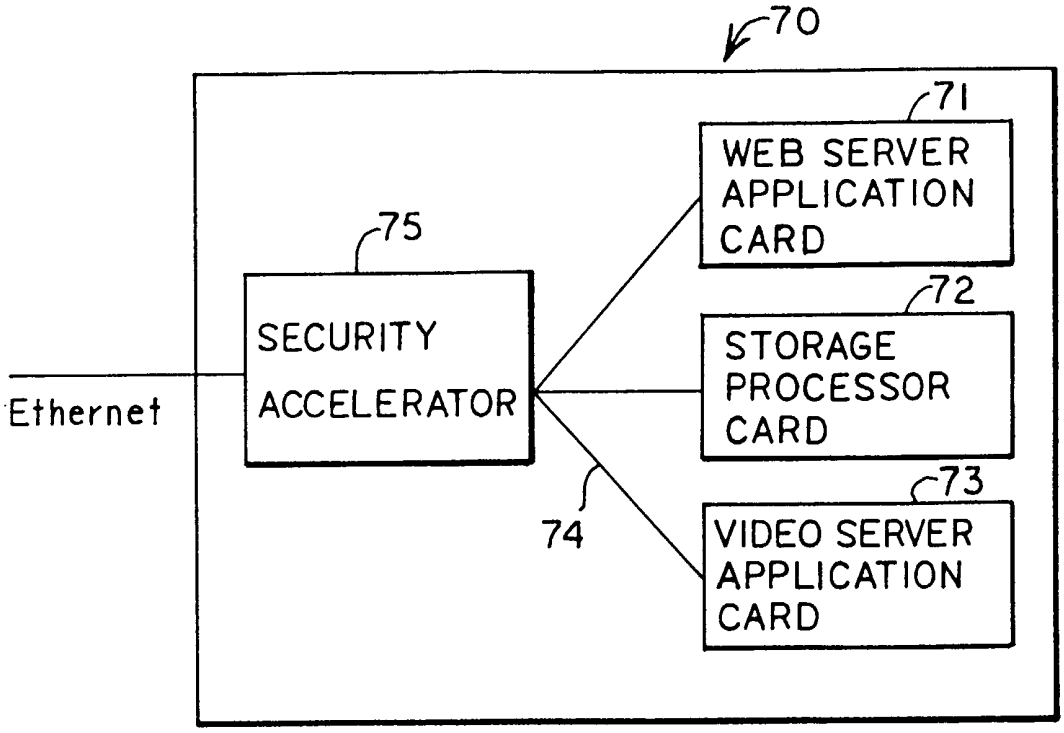

70
71
WEB SERVER APPLICATION CARD
75
SECURITY ACCELERATOR
72
STORAGE PROCESSOR CARD
Ethernet
74
73
VIDEO SERVER APPLICATION CARD

FIG. 7

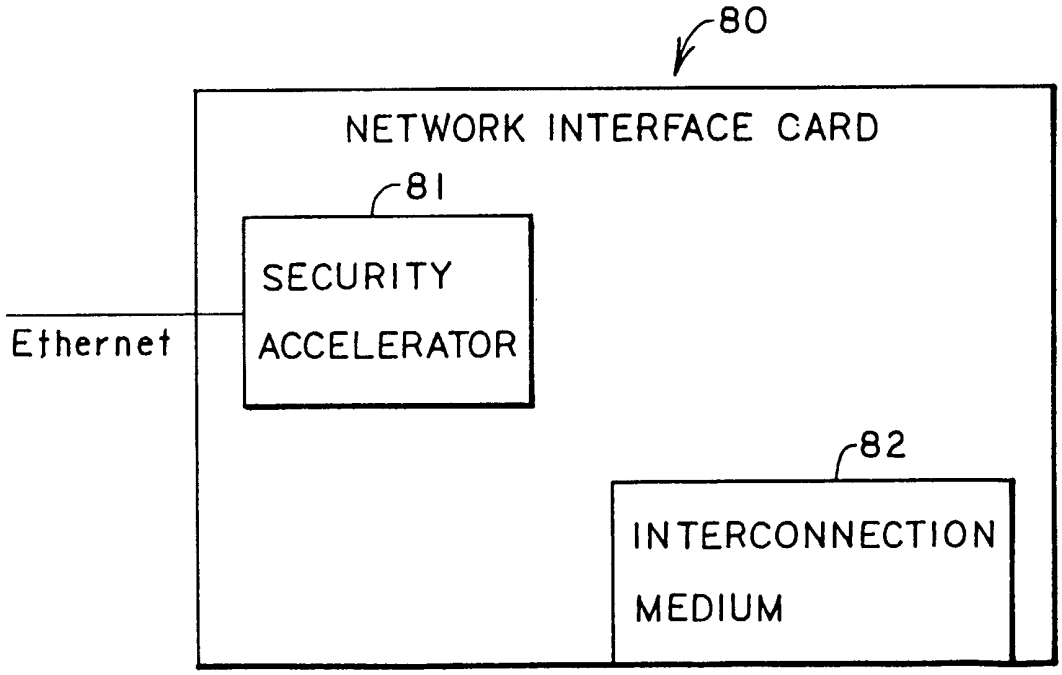

80
NETWORK INTERFACE CARD
81
SECURITY ACCELERATOR
Ethernet
82
INTERCONNECTION MEDIUM

FIG. 8