An application network appliance having inter-module communication using a universal serial bus (USB) is described herein. According to one embodiment, a network element includes a lossless data transport fabric (LDTF), multiple service modules coupled to each other over the LDTF, and a service control module (SCM) coupled to each of the service modules over the LDTF for routing network data between the SCM and the service modules. The SCM is also coupled to each of the service modules via a universal serial bus (USB) for managing the service modules, where the network element operates as a security gateway to a datacenter having multiple servers. Other methods and apparatuses are also described.
FIG. 3
FIG. 4
FIG. 6
FIG. 7
FIG. 10
FIG. 16

NS AppC
CMP Library
SMP OS

CPU core
CPU core
CPU core

SW

HW

NSP
FIG. 18
FIG. 20
FIG. 21
APPLICATION NETWORK APPLIANCES WITH INTER-MODULE COMMUNICATIONS USING A UNIVERSAL SERIAL BUS

RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Patent Application No. 60/966,649, filed Aug. 28, 2007, which is incorporated by reference herein in its entirety.

FIELD OF THE INVENTION

[0002] The present invention relates generally to application network appliances. More particularly, this invention relates to application network appliance with inter-module communications using a universal serial bus (USB).

BACKGROUND

[0003] The ability to connect information technology infrastructure reliably, cost-effectively and securely is of high importance for today’s global enterprises. To communicate with customers, clients, business partners, employees, etc., the Internet has proven to be more appropriate compared to private communication networks. However, communication via the Internet, which typically uses TCP/IP (Transmission Control Protocol/Internet Protocol), also increases the requirements for data security. Network firewalls are one of the many examples of solutions for network security.

[0004] Enterprise Web Application Services build an important foundation for such client, customer, and employee communication. A very common configuration for hosting such enterprise web Application Services is shown in FIG. 1. As shown in FIG. 1, an enterprise can offer web Application Services to various clients and there are several possibilities for clients to connect to the servers depending on the location of the client relative to the servers’ location. The servers which provide the Application Services are typically located in the enterprise’s data center 1016 and are accessible, directly or indirectly, via World-Wide-Web (WWW) servers 1012. Sometimes enterprises provide access to the Application Services by making the application servers directly accessible by putting those application servers into a Demilitarized Zone (DMZ) 1011.

[0005] A client 1003 may connect via a Local Area Network (LAN) through the enterprise’s intranet 1013. Another client 1004 may connect through a Wireless LAN (WLAN) to the intranet 1013. Yet another client 1005 may be located inside the enterprise’s campus network 1015, which connects to the enterprise’s intranet 1013. An enterprise may have zero or more campuses 1014 and 1015. Yet another client 1001 may connect through the Internet 1000, or a client 1002 may have a mobile connection to the Internet 1000. In any case to prevent illegitimate access to the enterprise’s web Application Services, the “inside” of the enterprise’s network, the intranet 1013, is protected by having a network perimeter 1010, which may comprise firewalls, associated network interconnect, and additional resources “within” the perimeter network configured so as to be broadly accessible to users on the “outside” of the enterprise.

[0006] Behind the perimeter 1010, access is granted to legitimate client requests only, while illegitimate access is rejected. The fundamentals in determining whether an access request is legitimate or not are based on the network reference model from the International Organization for Standardization (ISO). This ISO network reference model classifies Network Services into seven layers.

[0007] Traditional security products generally assume the existence of a trusted intranet—locations where enterprises control their own LANs, switches and routers—which can be organized into or placed within some type of security perimeter, to protect its resources from the un-trusted Internet. However, in today’s business environment, enterprises no longer enjoy the same level of trust and control of their intranets, as enterprises increasingly rely on contractors, partners, consultants, vendors, and visitors on-site for daily operation. As a result, enterprises are exposing internal resources to this wide set of clients whose roles are also frequently changing. Thus, the network trust boundary, delineating inside and outside clients, is disappearing—a phenomenon referred to as “de-perimeterization”. In such an environment, protection of an enterprise’s resources—such as its intellectual property, as well as mission-critical and operational systems—becomes of critical importance. Also, most security exploits easily traverse perimeter security, as enterprises typically let through email, web and any encrypted network traffic, such as Secure Sockets Layer (SSL), Simple Mail Transfer Protocol (SMTP) with Transport Layer Security (TLS), and authenticated Virtual Private Network (VPN) traffic, for example via IP Security (IPSec). Traditional perimeter security approaches, for example firewalls, intrusion detection systems and intrusion prevention systems have little or no benefit at the perimeter in providing access control functions to the resources. They have become more attack mitigation mechanisms than access control mechanisms. Enterprises are coming to terms with the fact that a hardened perimeter strategy is un-sustainable.

[0008] Traditional firewall or router access control lists cannot protect application resources from unauthorized access because network parameters such as Internet Protocol (IP) addresses and IP port numbers no longer deterministically identify resources, nor identify users, clients, or applications accessing these resources. Network firewall technology was invented when enterprises had a limited set of applications such as Telnet, File Transfer Protocol (FTP), and Email, and its primary functions were to limit access to specific applications from the outside and to limit access by systems within the enterprise to specific applications outside the firewall. Network layer parameters such as source, destination IP address and TCP or UDP port numbers were sufficient to identify the client and/or operations the clients intended to perform on a particular resource. However, with the proliferation of mobile devices and tunnelled applications, the network layer parameters are no longer useful to identify the client, the resource accessed, and the operation. Firewalls have evolved over the time, embracing functions such as deep packet inspection and intrusion detection/prevention, to handle application-level attacks, but the core access control function remains the same.

[0009] In effect, de-perimeterization demands that access control functions are positioned close to application resources and that a micro-perimeter is established in the heart of the data center by placing an identity-based policy enforcement point in front of any application resource. Enterprise business drivers for such an enforcement point are the need for rich and uniform protection of resources, business agility via attribute-based, policy-driven provisioning, and regulatory compliance. Traditional server-centric authorization solutions providing role-based authorization often
require custom code development, extensive cross-vendor testing whenever there is a version change (of the underlying operating system, agent or application), and are costly and difficult to maintain because of their proprietary nature. Also, traditional server-based network appliances—primarily focused on low-bandwidth ISO Layer-4 to ISO Layer-7 perimeter services—are unsuitable for data center deployment, both in functional richness and in ISO Layer-7 performance.

[0010] Within an application network appliance, modules can upload software and firmware, can perform configuration management, and they can exchange status and control information which can, for example, be diagnostics, initialization, power up and power down commands, reset, environment monitoring, etc. This requires certain inter-module communication. Various options exist in the art for such inter-module communication. An I²C bus may be used which has very low cost but which also is very slow and does not support hot-plug connectivity. A serial RS-232 or RS-422 link may be used which has the same drawbacks as the I²C bus. Alternatively, Ethernet may be used, which has sufficient bandwidth however, generally, this is not cost effective.

SUMMARY OF THE DESCRIPTION

[0011] An application network appliance having inter-module communication using a universal serial bus (USB) is described herein. According to one embodiment, a network element includes a lossless data transport fabric (LDTF), multiple service modules coupled to each other over the LDTF, and a service control module (SCM) coupled to each of the service modules over the LDTF for routing network data between the SCM and the service modules. The SCM is also coupled to each of the service modules via a universal serial bus (USB) for managing the service modules, where the network element operates as a security gateway to a data-center having multiple servers.

[0012] Other features of the present invention will be apparent from the accompanying drawings and from the detailed description which follows.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] The present invention is illustrated by way of example and not limitation in the figures of the accompanying drawings in which like references indicate similar elements.

[0014] FIG. 1 illustrates a typical corporate computer network connected to the Internet;

[0015] FIG. 2 illustrates the application of an application network appliance (ANA) as the APS according to one embodiment of the invention;

[0016] FIG. 3 is a network connected block diagram of an ANA according to one embodiment of the invention;

[0017] FIG. 4 is a block diagram of an ANA with a System Control Module (SCM) according to one embodiment of the invention;

[0018] FIG. 5 is a block diagram of an ANA with two or more SCMs according to another embodiment of the invention;

[0019] FIG. 6 is a block diagram of an ANA using two or more ANAs with a SCM according to another embodiment of the invention;

[0020] FIG. 7 is a block diagram of an ANA using two or more ANAs with two or more SCMs according to yet another embodiment of the invention;

[0021] FIG. 8 is a block diagram of a Network Service Module (NSM) of an ANA according to one embodiment of the invention;

[0022] FIG. 9 is a block diagram of a NSM of an ANA according to another embodiment of the invention;

[0023] FIG. 10 is a block diagram of an Application Service Module (ASM) of an ANA according to one embodiment of the invention;

[0024] FIG. 11 is a block diagram of an ASM of an ANA according to another embodiment of the invention;

[0025] FIG. 12 is a block diagram which illustrates LDTF connectivity between a NSM and an ASM of an ANA according to one embodiment of the invention;

[0026] FIG. 13 is a block diagram of functional components for inter-process communication between a NSM and an ASM of an ANA according to one embodiment of the invention;

[0027] FIG. 14 is a block diagram of a SCM for an ANA according to one embodiment of the invention;

[0028] FIG. 15 is a block diagram of a NSM of an ANA according to yet another embodiment of the invention;

[0029] FIG. 16 is a block diagram of a software architecture for a NSP of an ANA according to one embodiment of the invention;

[0030] FIG. 17 is a block diagram of an operating system for a NSP of an ANA according to one embodiment of the invention;

[0031] FIG. 18 is a block diagram which illustrates the application software blocks of a NSP of an ANA according to one embodiment of the invention;

[0032] FIG. 19 is a block diagram of an ASM of an ANA according to yet another embodiment of the invention;

[0033] FIG. 20 is a block diagram which illustrates the connectivity of the LDTF according to another embodiment of the invention;

[0034] FIG. 21 is a block diagram which illustrates details of the LDTF connectivity according to another embodiment of the invention;

[0035] FIG. 22 is a block diagram which illustrates inter-process communication between a NSP and an ASP in an ANA according to one embodiment of the invention;

[0036] FIG. 23 is a block diagram which illustrates connectivity of SCMs and other modules according to one embodiment of the invention;

[0037] FIG. 24 is a block diagram which illustrates connectivity of SCMs and other modules according to another embodiment of the invention;

[0038] FIG. 25 is a block diagram of a management plane for SCMs of an ANA according to one embodiment of the invention;

[0039] FIG. 26 illustrates an operating system with drivers for a SCM of an ANA according to one embodiment of the invention;

[0040] FIG. 27 is a flow diagram which illustrates device hot-plug capability of an operating system of an ANA according to one embodiment of the invention.

DETAILED DESCRIPTION

[0041] In the following description, numerous details are set forth to provide a more thorough explanation of embodiments of the present invention. It will be apparent, however, to one skilled in the art, that embodiments of the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in
block diagram form, rather than in detail, in order to avoid obscuring embodiments of the present invention.

Reference in the specification to "one embodiment" or "an embodiment" means that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment of the invention. The appearances of the phrase "one embodiment" in various places in the specification do not necessarily all refer to the same embodiment.

One aspect of the invention is a system and method for Inter-Module Communication using USB Bus in Layer-7 Networking, comprising a networking system including at least two communication planes, one communication plane for network traffic, and one for out-of-band communication, where the out-of-band communication is done using Universal Serial Bus.

Overview

The approach described herein applies combinations of parallel, multi-processor computing technology with lossless, low-latency, high-bandwidth network fabric technology (also known as Lossless Data Transport Fabric, or LDTF) to form novel methods and systems for high performance, high-reliability, high availability, and secure network applications. The various embodiments of the inventions described herein enable the implementation of highly reliable, highly scalable solutions for enterprise networking such as, for example, the APS 2000 from Fig. 2.

Multiple network Services are efficiently provided by terminating transport protocols centrally. As can be seen, any transport protocol can be terminated centrally, each PDU’s payload can be collected and converted into a data stream and, vice versa, a data stream can be converted into PDUs for any transport protocol and be transported via the given transport protocol. A simple concatenation of the PDU payload into a byte-stream is not sufficient. Key to the conversion is that state information must be maintained about the meta-data of each connection. Such meta-data includes the session information, for example via a unique connection identification number, the transaction information, as well as the information regarding segments and packets. Finite state machines can be used to track the meta-data.

Transport protocols are protocols which are used to transport information via networks. These include, obviously, the ISO Layer-3 protocols such as IPv4, IPv6, IPsec, the ISO Layer-4 protocols such as TCP, UDP, SCTP, the various ISO Layer-5 protocols such as FTP, HTTP, IMAP, SMTP, HTTP, L2TP, PPTP, SOAP, SDP, RTSP, RTP, RTCP, SSH, TLS, DTLS, SSL, IPsec, and VPN protocols. However, other protocols and approaches are contemplated within the scope of the inventions, which serve as transport mechanisms for transmitting information and application data and can also be terminated in a centralized fashion by a protocol proxy and the corresponding PDUs can be transformed into a data stream for application layer processing. Examples of such are, CSSV2, CORBA, IOP, DCOM and other Object Request Brokers (ORB), MPEG-TS or RTP as a transport for multimedia information, RTSP or SIP as another transport for multimedia information, peer-to-peer transport mechanisms, transport mechanisms based on J2EE such as Java RMI, streaming media protocols such as VoIP, IPTV, etc.

For the sake of simplicity we will use the term Centralized Transport Protocol Termination throughout the rest of the description, however, this is for exemplary purposes only and is not intended to be limiting. Centralized Transport Protocol Termination can be performed by dedicated processing units, and different ISO Layer-7 services can be performed in other dedicated processing units. The use of a lossless low-latency high-bandwidth fabric for inter-process communication between such dedicated processing units makes it possible to simultaneously support Centralized Transport Protocol Termination for multiple services. For example, TCP can be terminated once, transformed into a data stream and this data stream is transported from one dedicated processing unit to another using the lossless low-latency high-bandwidth fabric. The low-latency nature of the fabric helps to reduce the overall latency in client-to-server transactions.

In one embodiment, the Application Protection System (APS) 2000 is a network appliance that can act as a proxy between the client 2001 and the application server 2005, and can determine whether a client 2001 shall be granted access to certain applications 2005. In one embodiment, the client 2001 is one or more of the clients 1001, 1002, 1003, 1004, or 1005 of FIG. 1. In another example, the client 2001 can be a virtual machine or a cluster of computers, or a server (for server-to-server connections, for example). The application server 2005 can be, for example, without limitation, one or more file servers, one or more web servers, one or more database servers, one or more compute servers, one or more storage servers or one or more game servers. The decision whether access is granted or rejected involves an Identity Management Server 2003 to identify the user, client, or application, for example using Lightweight Directory Access Protocol (LDAP) or Active Directory (AD), and is the result of querying a Policy Server 2002 to analyze the access policy for the requested application 2005.

The APS 2000 may use a Triangulated Authorization method which, for example, is based on multiple aspects of a client (such as the client 2001), the requested application (such as application 2005) and certain network characteristics: Who—a client (a user or a machine) and its associated attributes such as department, role, project association, seniority, citizenship, etc.; Where—network and environment attributes such as access methods (wire-line/wireless/VPN), location (e.g., USA, Switzerland, China) and time; What—on-the-wire session attributes, including protocol and content/resource attributes. The outcome of this Triangulated Authorization method can be used to determine whether access to an application is granted or rejected. Optionally, a Single-Sign-On (SSO) server such as server 2004 may be involved that allows the client 2001 to obtain authorization for accessing multiple applications at once.

One embodiment of the invention acts as a proxy between one or more clients and one or more application servers to control the access of the one or more clients to the one or more applications. This is described, for example, in FIG. 2, where the APS 2000 controls access of client 2001 to application server 2005. Thereby the approach can act as a high-speed, full proxy which terminates both client-side and server-side transport protocol connections, and which behaves as a virtual server to the one or more clients, and as a virtual client to the one or more servers. The proxy function is required because of the need to reassemble PDUs into data streams and (where needed) to decrypt the payload data for inspection such as access control. The proxy function involves ISO Layer-2 to ISO Layer-5 processing such as Centralized Transport Protocol Termination.
One embodiment of the invention is a network appliance which terminates multiple transport protocols in one central point to overcome the many drawbacks of multiple transport protocol termination, such as increased latency and lack of scalability. Therefore, the network appliance may need to perform a set of functions similar to those typical of application servers such as network proxy, deep packet inspection, cryptography, data compression, regular expression parsing, etc. Network services that may need Centralized Transport Protocol Termination include but are not limited to application authentication and authorization, application firewalls, application data routing, in-line intrusion-detection and intrusion prevention, SSL offloading/acceleration, server load balancing, XML offloading/acceleration, and application front-end engine services (also called application acceleration).

ISO Layer-2 to ISO Layer-5 processing typically involves packets, segments and records processing, whereas ISO Layer-7 processing typically involves application data processing. Full ISO Layer-7 inspection goes beyond application headers and typically involves reassembling application layer data. A general rule used in the art is that a 1 GHz processor is needed for processing ISO Layer-3 or ISO Layer-4 PDUs at 1 Gbps, whereas a 10 GHz processor is needed for application data processing at 1 Gbps (for example for SSL VPN URL mangling operation). Therefore, the computational complexity required for scaling the proxy functionality is quite different from the computational complexity required for scaling ISO Layer-7 processing.

To solve the computational complexity in an efficient way, one embodiment of the invention splits the overall ISO Layer-2 to ISO Layer-7 stack into (at least) two independent processing domains. One domain, which is called Network Service processing for ISO Layer-2 to ISO Layer-5 processing (i.e., up to TCP/SSL) provides proxy functions, and a second domain which is called Application Service processing for ISO Layer-7 processing. Splitting the stack requires a reliable, lossless, low-latency, high-bandwidth connection between those two (or more) processing domains in order for the Network Service processing to forward the data stream to the Application Service processing for further processing. As a solution, this approach uses a LDTF such as RDMA-capable fabric technology to provide this reliable lossless, low-latency, high-bandwidth interconnect between processing domains.

Fig. 3 is a block diagram illustrating an example of application service appliance system according to one embodiment of the invention. Referring to Fig. 3, ANA 2100 acts as a proxy between a client 2104 and an application server 2105. The client 2104 is connected to the ANA 2100 via a network 2107. Network 2107 can, for example, be a LAN, a WAN, a WLAN, an intranet, or the Internet. The application server 2105 is connected to the ANA 2100 via network 2106. Network 2106 can, for example, be a LAN, a WAN, a WLAN, an intranet, or the Internet. Networks 2106-2107 may be the same network or different networks. While it is apparent that multiple clients and multiple application servers may be connected to the ANA 2100, for the sake of simplicity a single client, single application server case is used as a placeholder throughout. Incoming connections, for example, a request from the client 2104 is terminated in the NSM 2103 and is transformed into a data stream. This is done by PDU processing and reassembling the payload of the PDU into a data stream of ISO Layer-7 application data. This data stream is transported via LDTF 2102 to the ASM 2101 for further ISO Layer-7 processing. LDTF 2102 may be an RDMA or IB compatible fabric. The result of ISO Layer-7 processing done by ASM 2101 is then transported back—still as a data stream—via the LDTF 2102 to the NSM 2103. The NSM 2103 then transforms the data stream into PDUs and sends the PDUs to the application server 2105 via the appropriate transport protocol. Connections which originate from the application server 2105 can be handled similarly.

Using this novel approach, both processing domains can be scaled independent of each other and a well-balanced system can be achieved at reasonable costs.

One embodiment of the invention described herein is a system for access control in enterprise networking. Transport protocol connections can be terminated in one network appliance in a centralized manner, and how the different computational complexities of lower network layer processing and higher network layer processing can be addressed by splitting the network processing into two separate processing domains. A LDTF can be used for the inter-process communication between those domains.

In one embodiment of the invention, the LDTF is implemented using the IB point-to-point switch fabric architecture. Incoming connections from the client are terminated in the NSM and are transformed into a data stream. This data stream can, for example, without limitation, be transported via the IB fabric. In one other embodiment of the invention, the LDTF is implemented using an RDMA-capable interconnect fabric such as fabric. In further embodiments of the invention, it is contemplated that other LDTFs may be used as interconnect fabrics, for example, without limitation, I WARP and other interconnect fabrics such as are known or may become known to one of ordinary skill in the art.

This can be done by PDU processing and reassembling the payload of the PDUs into their corresponding data stream. This data stream is transported via IB fabric to the ASM for further ISO Layer-7 processing. The result of ISO Layer-7 processing done by ASM is then transported back—still as a data stream—again via the IB fabric to the NSM. The NSM then transforms the data stream into PDUs and sends the PDUs to the application server using the appropriate transport protocol. Connections which originate from the application server can be handled similarly.

One benefit of the present approach is the overall reduction of latency in the communication link between clients and application servers. Yet another benefit is that the approach can be scaled with various, specialized, dedicated processing modules.

Fig. 4 illustrates how SCMs can be connected to the other components. The ANA 2300, which can, for example, be the ANA 2100 of Fig. 2, behaves as a proxy for client-to-server connections and can be connected, for example, to a client 2304 and an application server 2305. The ANA 2300 can have one or more NSMs, such as NSM 2303, connected via LDTF 2302 to one or more ASMs 2301 for network processing. Also connected to the LDTF 2302 is a SCM 2306 which performs the administrative tasks. In one embodiment of the invention, IB is used as the LDTF, which can support virtual lanes and a dedicated virtual lane may be reserved just for system management communication involving the SCM.

For performance scaling purposes and to support high-availability, two or more SCMs can be connected to the LDTF. For example, in one embodiment of the invention, which is illustrated in Fig. 5, an ANA 2310, which behaves as
a proxy for client-to-server connections and connected for network processing, for example, to a client 2314 and an application server 2315. The ANA 2310 can have one or more NSMs, such as NSM 2313, connected via LDTF 2312 to one or more ASMs, such as ASM 2311. The ANA 2310 can also have two—or more—SCMs, such as SCM 2316 and SCM 2317, also connected to LDTF 2312.

[0062] In yet another embodiment of the invention, as is illustrated in FIG. 6, two—or more—ANAs, such as ANA 2340 and ANA 2350, can be connected via a high-availability link using LDTF. The high-availability link can be an external extension of the internal LDTFs 2342 and 2352. Each ANA can then operate as a backup ANA for one of its peers as it is described above. Similarly to NSMs and ASMs, the two—or more—SCMs can replicate their state information and update their state information in their backup ANA's SCM by writing state information into the peer's memory via the LDTF using, for example, RDMA. Similarly, in yet another embodiment of the invention, as is illustrated in FIG. 7, two—or more—ANAs, such as ANA 2360 and ANA 2370, can comprise two—or more—SCMs, such as SCM 2366 and SCM 2367, and SCM 2376 and SCM 2377, respectively.

L2-L5 Processing Unit—NSM

[0063] A NSM processes the lower network layers, ISO Layer-2 to ISO Layer-5. In any embodiment of the invention, such a NSM can be constructed as shown in FIG. 8. The NSM 2800 comprises a host channel adapter (HCA) 2801, a network services processor (NSP) 2802, and physical network layer receiver (Phy) 2803 and memory 2804. The host channel adapter 2801 connects to the LDTF, which can be IB fabric. The physical network layer receiver 2803 connects to Ethernet. The NSP 2803 runs programs stored in memory 2804 to perform ISO Layer-2 to ISO Layer-5 processing, such as Centralized Transport Protocol Termination, PDU reassembly to transform the PDU payload into a data stream, cryptographic processing, etc.

[0064] For better scalability, in one embodiment of the invention, a NSM can be a multi-processor architecture, as shown in FIG. 9. Here the NSM 2810 can comprise two—or more—NSPs, such as NSP 2812, NSP 2822, NSP 2832, each having a dedicated host channel adapter, such as host channel adapter 2811, host channel adapter 2821, and host channel adapter 2831, and dedicated memory, such as memory 2814, memory 2824, and memory 2834. A load balancer 2815 is in between the NSPs and the physical network layer receiver 2813 and balances the network load between the two—or more—NSPs. The load balancer 2815 can use common approaches known in the art to balance ingress or egress network traffic.

L7 Processing Unit—ASM

[0065] An ASM performs the ISO Layer-7 services, including application data processing on the data stream, which is the data stream of the transport protocol's PDU payload transformed by one or more NSMs. FIG. 10 illustrates how an ASM can be constructed in one embodiment of the invention. The ASM 3300 comprises a host channel adapter (HCA) 3301, an Application Service Processor (ASP) 3302, a bridge 3303 and memory 3304. The host channel adapter 3301 connects to the converged data center fabric which can be, for example, without limitation, LDTF or IB fabric. The bridge 3303 connects to the LDTF as a link to NSMs, for example. The ASP 3302 runs programs stored in memory 3304 to examine all ISO Layer-7 traffic and to perform ISO Layer-7 processing such as regular expression parsing, compression and decompression, standard and custom protocol proxy functions, etc.

[0066] For those tasks a high compute power is needed, typically more than for plain ISO Layer-2 to ISO Layer-5 processing. Therefore, a single-processor architecture using existing micro-processors may require hardware assist to provide sufficient compute power for high-bandwidth client-to-server connections. Alternatively, it may be advantageous to implement an ASM either as a homogeneous multi-processor system of generic ISO Layer-7 processing units, or as a heterogeneous multi-processor system using a sea of different, specialized ISO Layer-7 processing units. FIG. 11 shows such a multi-processor architecture: Here the ASM 3310 can comprise two—or more—ASPs, such as ASP 3312, ASP 3322, ASP 3332, each having a dedicated host channel adapter, such as host channel adapter 3311, host channel adapter 3321, and host channel adapter 3331, and dedicated memory, such as memory 3314, memory 3324, and memory 3334. The LDTF bridge 3313 connects the ASPs via the LDTF to the NSMs, for example.

[0067] For building the multi-processor architecture of the ASM several options exist: A multi-core processor technology can be used, which can be a System-on-a-Chip with on-chip hardware accelerators; or one can use multi-core processors with external co-processors, for example, a co-processor for cryptographic operations, a co-processor for regular expression analysis, a co-processor for data compression and decompression, etc. A parallel-mode compute architecture can be deployed which will require a flow dispatcher to distribute incoming traffic across the multiple processors. A pipelined-mode compute architecture can be used, where one processing element acts as a pre-processor for a subsequent processing element. Or, a hybrid approach can be used combining parallel mode with pipelined compute architectures. Further, any other architecture contemplated by one of skill in the art may be used.

LDTF to Connect L2-L5 Unit with L7 Units

[0068] In any case, the compute architecture requires a lossless, low-latency, high-bandwidth fabric for any-to-any inter-process communication links between the one or more NSMs (which each may comprise one or more NSPs) and the one or more ASMs (which each may comprise one or more ASPs). FIG. 12 shows how in one embodiment of the invention, one ISO Layer-2 to ISO Layer-5 processing unit, ASM 3441, and one ISO Layer-7 processing unit, ASM 3443, can be connected via the LDTF 3442. Key to the connection is the use of an RDMA network interface connector (RNIC) which can be a host channel adapter for IB, for example, host channel adapter 2801, or host channel adapter 2811, or host channel adapter 2821, or host channel adapter 2831, or host channel adapter 3301, or host channel adapter 3311, or host channel adapter 3321, or host channel adapter 3331. Of course, two or more ISO Layer-2 to ISO Layer-5 processing units can be connected to two or more ISO Layer-7 processing units accordingly.

[0069] Many options exist for implementing the LDTF 3442: In one embodiment of the invention the LDTF can be IB. In another embodiment of the invention the LDTF can be Data Center Ethernet with RDMA support. In yet another embodiment of the invention, the LDTF can be iWARP which supports RDMA over TCP. Besides being a lossless, low-
latency, high-bandwidth interconnect means RDMA enables the performance of RDMA one-sided read-based load monitoring and can be used to map connection level flow control using RDMA queue-pair flow control.

Stream Switch Architecture Based on LDTF

One fundamental, novel principle of this approach is to split the processing architecture into separate planes: A Management Service plane, a Network Service plane and an Application Service plane. The Management Service plane comprises one or more SCMs and is used for all out-of-band connectivity to processing elements on the Network Service plane and to processing elements on the Application Service plane and can be used, for example, for software image downloading, command-line interface, statistic collection messages, general system management functions, configuration management, etc. The Network Service plane comprises one or more NSMs for ISO Layer-2 to ISO Layer-5 processing and proxy functions. The Application Service plane comprises one or more ASMs for ISO Layer-7 services processing and for data stream analysis. As discussed above, this division into a Network Service plane and Application Service plane should be viewed as exemplary only, and other divisions and arrangements and number of service planes may be contemplated by one of skill in the art.

This tri-planar architecture is, for example, shown in FIG. 4, where ASM 2301 performs the processing for the Application Services, NSM 2303 performs the processing for the Network Services and SCM 2305 performs the processing for the Management Service plane. The lossless, low-latency, high-bandwidth LDTF 2302 connects these processing planes for efficient, reliable and scalable inter-process communication. While FIG. 4 explains the tri-planar architecture for the ease of converged data center fabric connections to application servers, this tri-planar architecture can easily be adjusted to function with standard Ethernet for application server connections.

One embodiment of the invention is shown in FIG. 13, which shows exemplary, non-limiting functional components of an ANA. The processing in Application Service plane is done by ASP components 3601, the processing in the Network Service plane is done by NSP components 3630, the processing in the Management Service plane is done by Management Service processor components 3621 and the LDTF inter-process communication is done by the IB Verb API 3620 which utilizes standard IB techniques known in the art. The ASP components 3601 comprise an ASP configuration agent 3602, the rule engine run-time build API 3603, the user/attribute manager 3604, the Virtual Directory Infrastructure 3605, the rule engine PDP and PEP 3606, the session manager 3607, the HTTP proxy 3608, the high-availability manager 3609, the protocol extension languages 3610, the socket or event library 3611, the application switch upper half 3612. The ASP configuration agent 3602 interacts with the ASP configuration broker 3622 from the Management Service plane 3621 to perform administrative tasks, such as configuration of components with appropriate parameters. The rule engine run-time build API 3603 provides a procedural interface for building rules based on the policies loaded. The user and attribute manager 3604 extracts the various attributes from the data stream which are needed to evaluate policy rules. The user and attribute manager 3604 can, for example, comprise the user/attribute manager and the content attribute manager. The Virtual Directory Infrastructure 3605 provides routines for interacting with Virtual Directory Infrastructure. The rule engine PDP and PEP 3606 provide routines for evaluating rules from policies. The session manager 3607 provides routines for extracting, managing and storing session information. The HTTP proxy 3608 provides routines to perform operations required when proxying the HTTP protocol in this centrally terminated stream-switch architecture. The high-availability manager 3609 performs routines for monitoring components and for synchronizing redundant stateful data in the various components. The protocol extension languages 3610 provide routines required for proxying custom protocols from Application Services. The socket or event library 3611 provides, for example, routines for non-RDMA communication which uses TCP sockets. The application switch upper half 3612 interacts with the IB Verb API 3620 and provides routines for RDMA-based inter-process communication.

Modules Overview—SCM

The SCM comprises one or more Management Service processors which run, for example, routines for chassis management, configuration management, software image management, auditing and logging, and platform high-availability. Because these do not require a lot of compute power a low-end standard micro-processor for the one or more Management Service processors is sufficient. FIG. 14 shows how a SCM can be connected with other components according to one embodiment of the invention. In an ANA a chassis backplane 2511 provides various alternatives for connectivity such as RS232, USB, and Ethernet. The chassis backplane 2511 also provides connectivity for the LDTF 2506. The chassis backplane 2511 can comprise a Display 2510. The ANA can comprise a PCI-local bus 2508 which connects to a USB Host Controller 2501 for off-of-band connectivity to other components and line cards, a SAS/SATA Controller 2502 to attach hard-disks for storage, and a Netchip USB 2503 for connectivity to SCMs. The PCI-local bus 2508 can be controlled by the Local bus control FPGA 2509. The PCI-local bus 2508 can connect to a PCI-X to PCI Bridge 2507 which has a PCI-X bus 2512 on the other side. Connected to the PCI-X bus 2512 can be an IB host channel adapter (HCA) 2504 and a Management Service processor 2505. The Management Service processor, which can, for example, be an Octeon CN3630 CPU from Cavium Networks, is also connected to the Chassis backplane 2511 via RS232 and via Ethernet.

Modules Overview—NSM

On the hardware side, a NSM comprises one or more NSPs. In one embodiment of the invention the NSM is the NSM 2800 of FIG. 8. In another embodiment of the invention the NSM is the NSM 2840 of FIG. 15. The NSM 2840 can comprise one or more 4× Gigabit Ethernet Phys 2845 and 2855, one or more 10 Gigabit Small Form Factor Pluggable Modules 2847 and 2857, an FPGA 2856, one or more NSPs 2842 and 2852, one or more host channel adapters 2841 and 2851, and one or more memory banks 2844 and 2854. The one or more NSPs 2842 and 2852 can, for example, be Octeon CN5860 CPUs from Cavium Networks. The NSPs 2842 and 2852 are connected to the FPGA 2856 via a SPI 4.2 bus. The NSPs 2842 and 2852 are also connected to the Ethernet Phys 2845 and 2855, respectively. And the NSPs 2842 and 2852 are connected to IB host channel adapters
2841 and 2851, respectively, via a PCI-X bus. The FPGA 2856 can perform network load balancing between two or more 10 Gigabit Small Form Factor Pluggable Modules 2847 and 2857 and two or more NSPs 2842 and 2852.

[0075] While FIG. 15 uses Octeon CNS860 CPUs from Cavium Networks, in general, many different possibilities exist for implementing a NSF and are contemplated within the scope of the present inventions. Because a NSF has to perform compute-intensive tasks which can be parallelized efficiently, it is desirable to use a multi-processing system for the NSF. In one embodiment of the invention, the NSF comprises multiple CPU cores for parallel processing. Because very specialized processing—namely ISO Layer-2 to ISO Layer-5 processing—needs to be done within a NSF, it is also desirable to deploy special purpose hardware accelerator units within a NSF. While the figures are described in conjunction with particular hardware, this is not intended to be a limitation. Other hardware, as known to one of skill in the art is contemplated within the scope of the present inventions.

[0076] On the software side, the one or more NSPs of a NSF runs, for example, without limitations, routines for ingress and egress processing for external data-path, for processing of the IP stack, for TCP and SSL termination, for fast-path flow switching, for data stream load balancing among multiple ASPs, for stream replication to backup NSPs, etc. FIG. 16 shows an exemplary software architecture for a NSF according to one embodiment of the invention described herein. The NSF 2940, which comprises one or more CPU cores, runs the symmetric multiprocessing operating system 3000. Above the SMP OS 3000 sits a Chip-Multi-Processing library 3100, which has special routines to exploit the parallel compute elements within the NSF. The Chip-Multi-Processing library 3100 can support parallel or pipelined multi-processing. On top of that sits the Network Service Application Container 3200. In one embodiment of the invention, the SMP OS 3000 of FIG. 16 can be the R-OS 3001 of FIG. 17. R-OS 3001 comprises the Linux kernel 26.x 302, the Configuration Manager 3003, the Event Manager 3004, Linux device drivers 3005, the RIMS layer 3006 which is an inter-process communication layer which provides R-OS infrastructure messaging services to service access points, the License Manager 3007, the Interface Manager 3008, the Chassis Manager 3009, the Feature Manager 3010, the Crypto Vault Manager 3011, and the High-Availability Manager 3012. In one embodiment of the invention the Network Service Application Container 3200 can comprise the routines 3201 as shown in FIG. 18. These can, for example, be used to perform data stream load balancing of incoming client traffic among two or more ASPs. Load balancing uses one-sided RDMA read operations for checking an ASP’s load without interrupting the processing on the ASPs.

[0078] Many different possibilities exist for implementing an ASP. Because an ASP has to perform compute intensive tasks which can be parallelized efficiently, it is desirable to use a multi-processing for the ASP. In one embodiment of the invention, the ASP comprises multiple CPU cores for parallel processing. Because very specialized processing—namely data stream processing—needs to be done within an ASP it is also desirable to deploy special purpose hardware accelerator units within an ASP.

[0079] On the software side, the one or more ASPs of an ASM run, for example, routines for HTTP protocol proxy functions, CIFS protocol proxy functions, JDBC protocol proxy functions, regular expression checks, protocol recognition, application authorization, and state replication to backup ASPs. The software architecture of an ASM is similar to the one of NSM described above.

Modules Overview—LDTF Connectivity

[0080] The LDTF provides the data plane connectivity between the one or more NSMs and the one or more ASMs. The LDTF can also provide management plane connectivity between the one or more SCMs, the one or more NSMs and the one or more ASMs. This is shown in FIG. 20 where, for example, two SCMs SCM1 2324 and SCM2 2325 provide LDTF switch 2321 and 2322. Connected to LDTF switch 2321 is Management Service processor MSP 2323—via host channel adapter HCA 2320—via host channel adapter HCA 2326—and NSP NSP 2329—via host channel adapter HCA 2328. Connected to LDTF switch 2322 is Management Service processor MSP 2323—via host channel adapter HCA 2320. In one embodiment of the invention, IB fabric is used to provide lossless, low-latency, high-bandwidth any-to-any switching. The IB fabric supports multicast communication and credit-based flow control. IB can support 16 virtual lanes; 15 virtual lanes can be used to implement the data plane and one virtual lane can be used to implement the management plane. The detailed connectivity of the IB fabric is shown in FIG. 21: The IB fabric 2331 which belongs to one SCM and the IB fabric 2322 which belongs to another SCM can be connected channel-wise via host channel adapters HCA 2333 and HCA 2334. Other IB fabric connections can go to other line card slots within the same ANA or can be used for inter-chassis high-availability links. Other LDTF fabrics may provide different limitations on the number and type of virtual and physical lanes. Further, the IB specification may evolve and improve in future. In addition, as will be appreciated by one of skill in the art, it is possible to combine multiple fabrics, for example, IB fabrics, and both aggregate and further virtualize the virtual lanes available within and among them.

Processing Flows

[0081] Splitting the data network processing into two separate domains, Network Service processing and Application Service processing—especially when constrained by scalability and high-availability—may require a particular processing flow between the one or more NSPs and the one or more ASPs.

[0082] For example, it is desirable to enforce flow-control because the proxy splits the client-server connection into two portions: One client-to-proxy connection which typically has a high round-trip delay time and low throughput and a proxy-to-server connection which typically has low round-trip delay
time and high throughput. The flow control for the client connection and the server connection mimic the behavior of the end-to-end flow-control of the original client-to-server connection. The internal LDTF enables the mapping of connection-level flow-control using RDMA queue-pair flow-control and therefore solves the problem created by splitting the client-server connection with a proxy.

The flow control for the client connection and the server connection mimic the behavior of the end-to-end flow-control of the original client-to-server connection. The internal LDTF enables the mapping of connection-level flow-control using RDMA queue-pair flow-control and therefore solves the problem created by splitting the client-server connection with a proxy. [0083] FIG. 22 shows a processing flow in accordance to one embodiment of the invention. The network processing is split between the Network Service processing 4020 and the Application Service processing 4010. The Network Service processing 4020 can, for example, be done by NSM 3300 of FIG. 10. The Network Service processing 4020 comprises Flow Manager 4025, TCP Proxy 4024, SSL Proxy 4022, Application Switch 4023, Channel API 4012, and Multi-Core Scheduling 4026. The Flow Manager 4025 performs network load balancing on ingress and egress network connections. The TCP Proxy 4024 does TCP termination and acts as an ISO Layer-2 to ISO Layer-4 proxy between client and server. The Application Switch 4023 transforms (among other processing) the PDU payload into a data stream. In case the network data is SSL encrypted, the data stream is forwarded to SSL Proxy 4022. Then the data stream is sent to the Channel API 4012 which sends the data stream data via the LDTF to the ASM’s Channel API 4014. The Multi-Core Scheduling 4026 performs load balancing of the network processing among two or more NSPs. The Application Service processing 4010 comprises the Channel API 4014, the Application Switch 4013, the Socket API 4012, the Application processing 4011, and the Application Container 4015. The Channel API 4014 receives the data stream data from the NSM’s Channel API 4021 and forwards it to the Application Switch 4013, which performs ISO Layer-7 processing on the data stream data such as Triangulated Authorization, etc. To submit the data stream data to the Application 4011, the Socket API 4012 is used. The Application 4011 can, for example, be applications 2005 from FIG. 2. The Application Container 4015 performs load balancing on the two or more ASPs such that the data stream information is either processed in a parallel fashion, in a pipelined fashion, or in a hybrid fashion.

[0084] Based on the granularity of the processing steps that can be distributed among the two or more NSPs, or the two or more ASPs, several options exist for load balancing, for example, in the Multi-Core Scheduling 4026 or in the Application Container 4015. In order to handle the events for multiple sockets, a typical application will map each socket to a thread or a process. The advantage with this approach is that the scheduling for different socket events is taken care of by the operating system, but the disadvantage is that process and thread scheduling is a very costly operation. Especially for high-speed network applications, which handle many connections, considerable CPU resources will be used just for process and thread scheduling. A library of ultra-light-weight threads can solve this problem by providing a light-weight execution context (the so-called strand) and by mapping a socket to each strand. The strand library enables having multiple strands within a system scheduling context of either processes or threads. Strand scheduling can be performed by a secondary scheduler. Essentially the operating system schedules the processes and threads, and the strand library schedules the strands. The strand scheduler can be completely I/O driven; i.e., a strand is scheduled whenever there is an incoming or outgoing event for a given socket. In order to provide an independent execution context for each strand, a separate stack can be allocated for each strand.

4.1.6 Scalability

[0085] Various embodiments of some of the inventions for scalability have been described in this disclosure, for example, the embodiment of the invention can not only be used for high-availability but also to scale an ANA for higher bandwidth and network processing demands. When two or more NSMs or two or more ASMs are connected via LDTF within one ANA, the inter-process communication between NSMs and ASMs then operates via so-called intra-chassis communication. Alternatively, when two or more ANAs are connected via LDTF, the inter-process communication then operates via so-called inter-chassis communication. Or, when both approaches are combined, both intra-chassis and inter-chassis communication goes over the LDTF.

Inter-Module Communication Using USB

[0086] In one embodiment of the invention, a modular architecture is used. Within this architecture the one or more SCMs or the one or more NSMs or the one or more ASMs can upload software and firmware, can perform configuration management, and they can exchange status and control information which can, for example, be diagnostics, initialization, power up and power down commands, reset, environment monitoring, etc. This requires certain inter-module communication. Various options exist in the art for such inter-module communication. An I²C bus may be used which has very low cost but which also is very slow and does not support hot-plug connectivity. A serial RS-232 or RS-422 link may be used which has the same drawbacks as the I²C bus. Alternatively, Ethernet may be used, which has sufficient bandwidth however, generally, this is not cost effective.

[0087] To overcome these disadvantages, in one embodiment of the invention, the various inter-module communication buses are consolidated into one common out-of-band bus, which utilizes USB technology. USB technology is well-established in PC and consumer products and is very fast (USB 2.0 supports up to 480 Mbps), cost-efficient, supports hot-plug connectivity and has high reliability. FIG. 23 shows how such inter-module communication can be implemented using USB technology. An ANA which can, for example, be ANA 2000 from FIG. 2, comprises one or more SCMs 2600 and 2610 which each have one Management Service Processor 2601 and 2611, respectively. Each SCM also has one USB Host Controller 2602 and 2612, respectively, which each is connected to a Line Card Module 2604 (via USB Slave 2603), to a Fan Module 2606 (via USB Slave 2606), to a Display Module 2614 (via USB Slave 2613) and to a Power Supply 2616 (via USB Slave 2615). Each SCM can then act as a USB master for inter-module communication. The USB connectivity can either be half-duplex or full-duplex.

[0088] FIG. 24 shows the detailed connectivity between the various processing elements if USB is used for inter-module communication. USB communication can be duplex or n-plex. In an ANA, one or more Management Service processors 2701 and 2704, for example, are connected to the USB Host Controller 2702 and 2704, respectively, via a PCI bus. Connected to these USB Host Controllers 2702 and 2704 are the various modules in the ANA, for example, NSM 2705, ASM 2706 and ASM 2707. In yet another embodiment of the invention, shown in FIG. 25, two
(or more) SCMs and their respective USB inter-module connectivity are given. The SCM SCM0 comprises the USB Multiplexer 2731, the USB Host Controller 2732, the USB Device 2733, the USB Hub 2734, the USB Fan Module 2736, and the Power Supply 2737. The SCM SCM1 comprises the USB Multiplexer 2744, the USB Host Controller 2745, the USB Device 2738, the USB Hub 2740, the USB Fan Module 2741, and the Power Supply 2742. The USB Device 2733 is connected to the USB Host Controller 2732 via PCI bus 2734. The USB Device 2738 is connected to the USB Host Controller 2745 via PCI bus 2739. USB Multiplexers 2731 and 2744 can, for example, be connected to a front panel of a chassis. The USB Fan Module 2736 and the Power Supply 2737 are connected with the USB Host Controller 2732 via the USB Hub 2735. The USB Fan Module 2741 and the Power Supply 2742 are connected with the USB Host Controller 2745 via the USB Hub 2740. Also connected to both USB Host Controllers 2732 and 2745 is the Line Card Management FPGA 2743 which can, for example, be located on a chassis backplane to support line card management functions.

One of the advantages of using USB technology, compared to other approaches known in the art, is that common operating systems, for example the R-OS shown in FIG. 26, support so-called hot-plug of components. FIG. 27 shows the Linux user-space hot-plug manager, the so-called udev device. This facilitates the administration of a high-availability enterprise ANA, such as the ANA 2000 from FIG. 2. Modules can be added, removed, replaced, etc. during runtime while the ANA is operating, without disrupting the services performed by the ANA.

Some portions of the preceding detailed descriptions have been presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the ways used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of operations leading to a desired result. The operations are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to those signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the above discussion, it is appreciated that throughout the description, discussions utilizing terms such as “processing” or “computing” or “calculating” or “determining” or “displaying” or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system’s registers and memories into other data similarly represented as physical quantities within the computer system’s registers or registers or other such information storage, transmission or display devices.

Embodiments of the present invention also relate to an apparatus for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general-purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer readable medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, and magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), erasable programmable ROMs (EPROMs), electronically erasable programmable ROMs (EEPROMs), magnetic or optical cards, or any type of media suitable for storing electronic instructions, and each coupled to a computer system bus.

The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general-purpose systems may be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatus to perform the required method operations. The required structure for a variety of these systems will appear from the description below. In addition, embodiments of the present invention are not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of embodiments of the invention as described herein.

A machine-readable medium may include any mechanism for storing or transmitting information in a form readable by a machine (e.g., a computer). For example, a machine-readable medium includes read only memory (“ROM”); random access memory (“RAM”); magnetic disk storage media; optical storage media; flash memory devices; electrical, optical, acoustical or other form of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.); etc.

In the foregoing specification, embodiments of the invention have been described with reference to specific exemplary embodiments thereof. It will be evident that various modifications may be made thereto without departing from the broader spirit and scope of the invention as set forth in the following claims. The specification and drawings are, accordingly, to be regarded in an illustrative sense rather than a restrictive sense.

What is claimed is:

1. A network element, comprising:
a lossless data transport fabric (LDTF);
a plurality of service modules coupled to each other over the LDTF; and
a service control module (SCM) coupled to each of the service modules over the LDTF for routing data between the SCM and the service modules, wherein the SCM is also coupled to each of the service modules via a universal serial bus (USB) for managing the service modules, and wherein the network element operates as a security gateway to a datacenter having a plurality of servers.

2. The network element of claim 1, wherein the service modules are configured to perform different layer processes on packets of a network transaction between a client and a server of the datacenter.

3. The network element of claim 2, wherein the service modules comprise at least one network service module (NSM) and at least one application service module (ASM), and wherein the at least one NSM is configured to perform
layer 2 to layer 5 (layer 2-5) processes and the at least one ASM is configured to perform layer 5 to layer 7 (layer 5-7) processes.

4. The network element of claim 1, wherein the SCM is configured to communicate with only one selected service module, which is configured to perform layer 2 to layer 7 (layer 2-7) processes.

5. The network element of claim 1, wherein the SCM comprises a host USB device and each of the service modules comprises a slave USB device, and wherein the host USB device initiates exchange of data between the SCM and a service module.

6. The network element of claim 5, wherein the USB bus is used to communicate between the SCM and a service module for exchanging status and control messages regarding diagnostics, initialization, power up/down, reset, and health monitoring of the service modules.

7. The network element of claim 6, wherein the USB bus is used to communicate between the SCM and a service module for software/firmware download.

8. The network element of claim 7, wherein the USB bus is used to communicate between the SCM and a service module for configuration management of ports and various device blocks.

9. The network element of claim 6, further comprising a fan module for cooling purposes, wherein the SCM is configured to control the fan module via the USB bus.

10. The network element of claim 9, further comprising a power supply module for providing power to the network element, wherein the SCM is configured to control the power supply module via the USB bus.

11. The network element of claim 10, wherein the SCM is implemented as a control card and the service modules are implemented in a plurality of line cards communicating with the control card via an backplane, wherein the backplane includes a portion of the USB bus and a portion of LDTF.

12. The network element of claim 11, wherein the SCM further comprises a USB host controller coupled to each line card via a line card management FPGA (field programmable gate array) for communicating with each line card using a USB compatible protocol, wherein the line card management FPGA is used to convert other management signals that are not USB compatible into USB compatible signals.

13. The network element of claim 11, wherein the SCM further comprises a USB hub device coupled to the USB host controller, wherein the fan module and the power supply module are coupled to the USB hub device.

14. The network element of claim 13, wherein the SCM further comprises a PCI (peripheral component interconnect) bus coupled to the USB host controller to allow one or more other USB devices to reach the USB host controller via the PCI bus.

15. The network element of claim 14, further comprising a redundant SCM module implemented on a redundant control card coupled to the backplane via the LDTF and USB bus, wherein the USB hub device is coupled with a fan module and a power supply module of a redundant control card having a redundant SCM module.

16. The network element of claim 15, wherein the fan module and the power supply module are also coupled with a corresponding USB hub device of the redundant control card.

17. The network element of claim 15, wherein the USB host controller is also coupled with one or more USB devices of the redundant control card.

18. The network element of claim 9, further comprising a display module for displaying information regarding operations of the network element, wherein the SCM is configured to control the display module via the USB bus.

19. The network element of claim 15, wherein the one or more other USB devices are coupled with a corresponding USB host controller of the redundant control card.

20. The network element of claim 1, wherein each of the service modules and the SCM module are hot pluggable using a plug-and-play protocol via the USB bus.

* * * * *