An Extensible, Programmable, Commercial-Grade Platform for Internet Service Architecture

Tal Lavian, Doan B. Hoang, *Member, IEEE*, Franco Travostino, Phil Yonghui Wang, Siva Subramanian, and Inder Monga, *Member, IEEE*

***Abstract*—With their increasingly sophisticated applications, users propel the notion that there is more to a network (be it an intranet, or the Internet) than mere L1-3 connectivity. In what shapes as a next generation service contract between users and the network, users want the network to erogate services that are as ubiquitous and dependable as dialtones. Typical services include appplication-aware firewalls, directories, nomadic support, virtualization, load balancing, alternate site failover, etc. To fulfill this vision, a service architecture is needed. That is, an architecture wherein end-to-end services compose, on-demand, across network domains, technologies, and administration boundaries. Such an architecture requires programmable mechanisms and programmable network devices for service enabling, service negotiation, and service management. The bedrock foundation of the architecture, and also the key focus of the paper, is an open-source programmable service platform that is explicitly designed to best exploit commercial-grade network devices. The platform predicates a full separation of concerns, in that control-intensive operations are executed in software, whereas, data-intensive operations are delegated to hardware. This way, the platform is capable of performing wire-speed content filtering, and activating network services according to the state of data and control flows. The paper describes the platform and some distinguishing services realized on the platform.**

***Index Terms*—Active networks, active services, programmable networks, service architecture, service platform.**

I. Introduction

INTRA-NETS and the Internet have served extremely well as conduits for connectivity. Emerging requirements set by mobility, resource virtualization, load balancing, and security, indicate that the network can step up to a much larger role than mere L1-3 connectivity. Moving forward, the network takes connectivity for granted, and provides a global infrastructure for services. Users want services that speak to their specific needs, not just a mere connection. Users want services that are ubiquitous, effortless, valuable, and highly available. Building upon a common service infrastructure, service providers will then differentiate themselves by catering to users' needs, offering relevant services, and adding values to existing applications.

The architecture of the proposed service infrastructure allows end-to-end services to be introduced on-demand. It allows service compositions across network domains, technologies, and administration boundaries [1]. It allows services to be negotiated with adequate quality of service (QoS). The architecture requires programmable, extensible mechanisms for service enabling, service negotiation, and service management. It is currently very difficult to extend a network with new services, specifically multidomain and multiprovider services (multicasting, mobility, QoS, etc.)

To establish such a service architecture, it becomes essential to redesign network devices (e.g., routers, switches, edge devices, load balancers, and firewalls) as open and highly extensible devices. These devices must be powerful enough to accommodate many useful classes of services. They must be intelligent enough to address the programmability requirements mentioned above. Above all, they must be flexible enough to adapt to users' needs. Ideally, an active networks (AN) node with its own NodeOS, APIs, and execution environments (EE) has potential to serve as the building block for such a service architecture. In reality, however, it has proven extremely difficult for this approach to take off and measure up against the speed and dependability that users have come to expect from a production network. We are not aware of an integrated NodeOS and EE platform implementation that exploits a (commercial) hardware platform and is therefore targeted to actual deployment.

This paper describes an alternative approach, one where the commercial, hardware reality is a starting point without compromises of sort. Given one such a platform, where silicon (e.g., ASICs, FPGAs, NPUs) assists wire-speed critical functions in substantial ways, what is the headroom left for extensibility and programmability? The paper argues that the headroom is remarkably high, once the designer adopts a clean separation of concerns between rich control predicates and lean data-forwarding rules. This approach leverages as much as possible the work done by the AN community and allows for active services such as intelligent agents, computational intelligence algorithms [2] to be introduced without performance penalties.

The paper is organized as follows. Section II briefly discusses related work on active and programmable networks. Section III presents our initial design of the Openet programmable node architecture on a router. Section IV presents the design of our programmable service platform on an edge device. Section V describes a particular programmable service platform that has

Manuscript received July 23, 2002; revised March 26, 2003 and July 21, 2003. This paper was recommended by Guest Editors W. Pedrycz and A. Vasilakos.

T. Lavian is with Nortel Networks, Santa Clara, CA 95054, USA (e-mail: tlavian@nortelnetworks.com).

D. B. Hoang is with the Faculty of Information Technology, University of Technology, Sydney, NSW, Australia (e-mail: dhoang@it.uts.edu.au).

F. Travostino and I. Monga are with Nortel Networks, Billerica, MA 01821, USA (e-mail: travos@nortelnetworks.com; imonga@nortelnetworks.com).

P. Y. Wang is with Nortel Networks, Ottawa, ON, **AU: PLEASE PROVIDE POSTAL CODE,** Canada (e-mail: pywang@nortelnetworks.com).

S. Subramanian is with the Nortel Networks, Raleigh, North Carolina, USA. He is currently a Ph.D. candidate in the Electrical and Computer Engineering Dept. at North Carolina State University (e-mail: ssiva@nortelnetworks.com).

Digital Object Identifier 10.1109/TSMCC.2003.818497

been created out of a high-performance content switch. Section VI presents a number of services that demonstrate useful applications of the platform. Conclusion is in Section VII.

II. Related Work

The AN research program [3] has the goal of producing a new network platform flexible and extensible at runtime to accommodate the rapid evolution and deployment of network technologies, and also to provide the increasingly sophisticated services demanded by defense applications. AN technologies [3] expose a novel approach that allows customer value-added services to be introduced to the network "on-the-fly." Typically, through the AN, applications can deploy new protocols and change or modify their services dynamically for specific purposes in terms of active packets. The exciting opportunity is that network service providers and third parties, not just the network device providers, can program the network infrastructure to provide value-added services and applications.

Traditional network nodes (e.g. routers on the Internet) enable end-system connectivity and sharing of network resources by supporting a static and well-defined set of protocols.

In contrast, active networks approach opens the opportunity for maximizing the utility of the service by the network to individual applications by allowing the applications themselves to define the service provided by the network.

A general architecture for ANs has evolved over the last few years. This architecture specifies a three-layer stack on each active node [4]. At the lowest layer, an underlying operating system (NodeOS) abstracts the hardware and provides low-level resource management facilities such as node's communication, memory, and computational resources among the various packet flows that traverse the node. At the next layer, one or more execution environments (EEs) provide the basic application programming interface API for writing active applications. At the topmost layer are the active applications (AAs), each of which contains code injected into the network to support a network service.

A number of active networks prototypes have been developed [5] to study whether active networks can deliver their claimed benefits in terms of new and novel services, while at the same time keeping the network efficient and secure. Recent DARPA Active Networks Conference and Exposition provides an excellent up to date research and development in active networks.

Work on extensible router architectures includes Princeton's Vera [6], MIT's Click router [7] and Washington University's Dynamically Extensible Router (DER) [8]. Vera employs COTS hardware and an open operating system. It includes the design of a NodeOS that takes advantage of the hierarchical framework of the architecture, while providing an API that is portable among different EEs and distinct NodeOS implementations. DER concentrates on hardware architecture. It provides an open, flexible, high-performance active router testbed for advanced networking research. It supports the dynamic installation of software and hardware plugins in the data path of application data flows. Click offers extensibility by configuring safe extensions into the operating system kernel.

Work on NodeOS includes four major implementations: OSKit-based Moab [9], Princeton's Scout-based system [10], Network Associates' exokernel-based AMP system [11], and the Bowman NodeOS from Georgia Tech and Kentucky [12] and JanOS from University of Utah. The Moab focuses on resource control and domain management, the Princeton system integrates NodeOS abstractions in Scout's paths where no protection domains were maintained, and the AMP focuses on security. Bowman predates the DARPA NodOS specification, but contains many of the same abstractions and features. Bowman relies on POSIX interface and runs in user mode on Solaris and Linux.

Work on EEs includes: ANTs [13], active signaling protocol EE (ASP EE) [14] focusing on signaling applications in the network, AMP-based EE [11] focusing on administration, CANE EE [15] focusing on protocol development.

The Tempest [16] provides a customizable control plane for ATM networks. The basic idea of high-performance active networking is by decoupling the forwarding path from a programmable control plane was introduced, in a software implementation, in the control-on-demand (CoD) [17].

From the brief summary discussion on ANs, it is clear that much has been achieved in designing of a generic, extensible router, in the developing of an active networks node architecture, in developing a NodeOS, in developing execution environments. However, very little has been done to leverage existing commercial devices. Our work differentiates from others in several aspects. First, it leverages the capabilities of existing network devices by developing necessary application programming interfaces to these devices and developing runtime environment for managing active services. Secondly, it supports two types of programmable platform: one that lives in the network core and one that lives at the network edges. Thirdly, it emphasizes on a programmable "service" platform as a building block for a future Internet service infrastructure.

Industrial organizations such as programming interfaces for networks (P1520/PIN) [18], network processing forum (NPF) [19] and Parlay [20] have been working on standardization of programmable networking interfaces among hardware, middleware services and user applications. P1520 focuses on development of open signaling, control and management applications as well as higher level multimedia services on networks. NPF focuses on common specifications to develop new networking and telecommunications products based on network processing technologies. Parlay focuses on development of applications across multiple, networking-platform environments.

In IETF, the Forces working group [21] is defining an architecture and wire protocol for the physical separation between control elements and forwarding elements. A dominant scenario (albeit not the only one) is the one of an IP router wherein control and forwarding elements interact to correctly implement IP routing and IP QoS. Forces and our work share the same impetus toward optimal use of commercial-grade network gear, and the attention to silicon trends. The scope of execution is mostly different. We define and use software APIs to de-couple syntax/semantics from the actual physical realization of the network device, which may or may not have its forwarding elements loosely coupled. Our APIs (in Java) sit much higher in the stack than the abstractions defined and used by forces.

III. Openet Architecture

The Openet architecture has been developed to support the dynamic introduction of application services that can apply active network control and alter packet processing. A chief goal of Openet is to build a network programming platform for service deployments on a commercial-grade network device (e.g., a router, a switch, a firewall). In doing so, the considerations are given to

1 preserving the hardware fast-path for data packets, and
2 leveraging existing active networking research efforts as much as possible.

To support the first goal, we have introduced the active flow manipulation (AFM) enabling technology [22]. The AFM mechanism involves two abstraction levels in the control plane. One is the level at which a node can aggregate transport data into traffic flows, and the other is the level at which it can perform simple actions on the traffic flows. The abstraction allows one to think and act in terms of primitive flows whose characteristics can be identified and whose behaviors can be altered by primitive actions in real-time. With AFM, customer network services can exercise active network control by identifying specific flows and applying particular actions thereby altering network behavior in real-time. These services are dynamically deployed in the CPU-based control plane and are closely coupled with the silicon-based forwarding plane of the network node, without negatively impacting forwarding performance.

To support the second goal, we have implemented a layer over which existing active network implementations can be ported. We added a Java-based run-time environment Oplet Run-time Environment (ORE) for security and service management over which existing EEs can be run as network services.

We chose to provide support for Java-based EEs by running an embedded JVM in the control plane of the network device, and by creating a Java-compatible interface to the low-level hardware. The Java-compatible interface that provides access to the low-level hardware is the Java Forwarding API.

The forwarding API—a set of Java classes—realizes the AFM [22] technology and provides interfaces for Java applications to control a generic, platform-neutral forwarding plane. The Forwarding API therefore, represents a uniform, platform-independent portal through which software services can control the forwarding path of heterogeneous network nodes. Using the forwarding API, customer network services access the AFM technology to exercise active network control by identifying specific flows and applying particular actions thereby, altering network behavior in real-time, such as, setting packet filters, changing the forwarding priority; and providing new protocol support.

As shown in Fig. 1, the Openet architecture [23], [24] results from the union of the ORE, the forwarding API, and other ancillary services allowing nontrusted, nonnative code (e.g., Java byte code) to be safely executed on the network device.

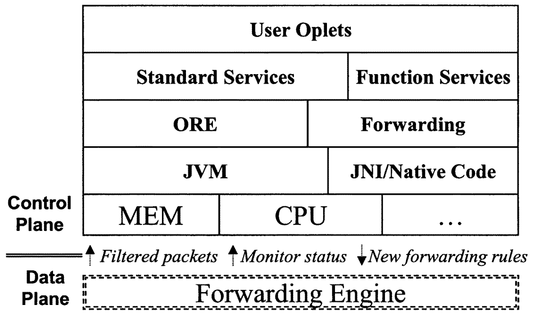

Fig. 1. Openet architecture.

IV. Design of a Programmable Service Platform

This section focuses on a high-level design of a powerful programmable service-enabling platform. The design takes the advantage of a hardware platform that possesses advanced silicon support for expedited operations. The target hardware support is not a traditional IP router but rather a network edge device where active services find their optimal place for the following reasons:

- Reasonable delay components can be tolerated at the network edge, where a packet stream is terminated or intercepted, and where some form of processing is often necessary before meaningful operation can be exercised.
- Rather than building up infrastructure support at a network device from scratch, the aim is to utilize as much as possible the advanced features of existing network devices. This helps to shorten the time for technology transfer.
- Security issue is always a difficult issue, the emphasis is on both software as well as hardware partition. An edge device is more powerful to afford additional facilities to provide additional security measures.
- Open-Closed network device issue: once it is demonstrated that the programmable active devices are realizable on existing state of the art network devices and able to address important issues such as QoS, mismatches between network domains, traffic engineering, and multimedia applications, network vendors might look seriously at the advantages to support open active nets platforms.

In designing a programmable service platform, we identify several distinguishing factors:

- The network device is an enhanced content switch. This implies that traffic can be filtered or differentiated based not only on the transport protocol or the routing behavior but also on the application content. This in turn implies that a larger range of and more personalized active services can be developed.
- The network device comprises multiple powerful computation components and hence, computationally intensive applications can be contemplated. Whereas, a simple router platform has very limited computational power that only allows light-weighted services to be accommodated.
- The network device provides fast redirection tunnels and computation components, so the AFM technology can be fully explored to accommodate both real-time and non real-time applications.

- The new platform enjoys a variety of extensible resources from computing to storage, when necessary, by allocating a specific application to a specific computation component. This practically enhances the security aspect of an application.

The focus of the paper is on the Openet architecture with a "content switch" to form a new active/programmable service platform, rather than an active router [25]. The value of an active/programmable device is centered on the relevant and significant active services it can purport. It is hence natural to consider the architecture from a service-centric viewpoint. Furthermore, our aim is to deploy service platforms as building blocks for an investigation into a feasible internet service architecture. High-level design of the platform views it as a set of comprehensive service interfaces that addresses the issues mentioned in the introduction.

For sake of generality, the Openet architecture has been ported to other devices, including a gigabit routing switch (Nortel Passport 8600). In this platform, the forwarding plane along the data path is implemented using ASIC's that can forward packets up to 256 Gb/s (gigabits p/s) in wire speed without consuming any CPU resource. The control plane is based on a CPU blade and contains the embedded Java VM. It executes ORE and in turn enables execution of diversified network services. When contrasted to the afore-mentioned context switch, however, the Passport platform is substantially different. Firstly, the hardware can only perform L2–L4 filtering, and finer differentiation has to be done by the active service within the control plane itself. Secondly, the computational power of the control plane is limited by the CPU of the router whose optimal design does not allow much use of the CPU cycles on tasks other than routing. Thirdly, it is difficult to partition the resources and allocate them securely to different Execution Environments.

Lastly, the Openet platform was retrofitted to a PC Linux, yielding an opportunistic technology vehicle, without any special hardware assist (a degenerate case for the forwarding API and its AFM technologies).

A. Service-Centric Architecture

From a service-centric viewpoint, the active platform is designed to provide programmable support for both network and user services. To function as an active platform, we identified a number of programming interfaces.

Service Enabling Interface. This interface allows services elsewhere in the network to be deployed, to be initialized and launched within its own execution environment in the platform.

Control Interface. This interface basically controls the hardware of the devices. The interface can be grouped into three groups of functions: Forwarding API, Content API, and Compute API.

- *The Forwarding API* deals with various filtering, monitoring functions at the L2/L3 layers, for routing incoming data flow from an input port to an output port. This interface may include functions for classifying flows, monitor flow usage, and policing.
- *The Content API* deals with the filtering of an application flow at L4–L7 layers and directing the flow to a particular processing and/or storage unit (or service).
- *The Compute API* deals with the service processing with adaptively configuring service processors, processing units, and storage units to carry out an on-demand service.

Management Interface. This interface allows the device to be managed in the usual way. This may include CLI, GUI, and schemes to get access to the device MIB.

Intra-Service Communications Interface. This interface allows a service in one control plane to communicate with its peers in other control planes for exchanging information as necessary to the operation of the device. This may involve concurrent processing, data synchronization, socket API, or other communications mechanisms.

Impedance Matching Interface. This interface allows the devices to interact intelligently to another device may belong to a different technology or different networks (wired, wireless, optical, etc.) to handle the impedance mismatches in terms of bandwidth, reliability, delay, etc. The interface will also address the concerns of QoS. Presently, it is empty, but it is envisaged that the interface will accommodate various interdomain signaling protocols.

Security Interface. This interface addresses important security issues pertaining the operation of the gateway. The interface addresses the secured the download, the partition of services in the control plane, the authenticated components, and the security of the communications. Fig. 2 depicts this service-centric platform architecture.

B. Software Architecture

Communications between the control plane and its forwarding/processing hardware is through the control interface. It provides three main functional groups: content processing functions, filtering and reconfiguring functions, and forwarding functions. Communication between the control plane and external services, administrators, users are through the management interface. Some of the functions in the interface are for configuring the initial set up of the device, for setting and collecting information about the device, some of the functions are for downloading. From the architectural viewpoint, it is not possible to show all the interfaces of the service-centric viewpoint (Figs. 3 and 4). These are subsumed in the control, the service/management and other interfaces.

From a software framework viewpoint, the control plane consists of a virtual machine on top of the device real operating system, a service framework, and application services. Fig. 5 shows our general architectural framework. The service framework provides mechanisms for downloading, managing the life cycles and the security of injected execution environments. It may also provide systems and/or a set of standard services for supporting application services.

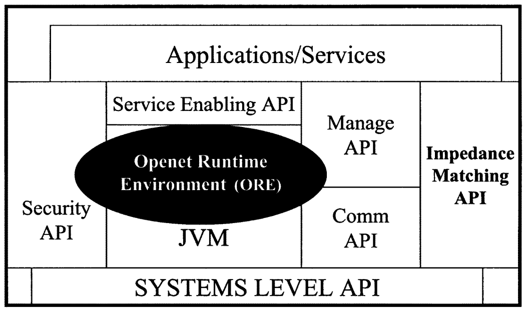

Fig. 2. Programmable platform architecture: a service-centric view.

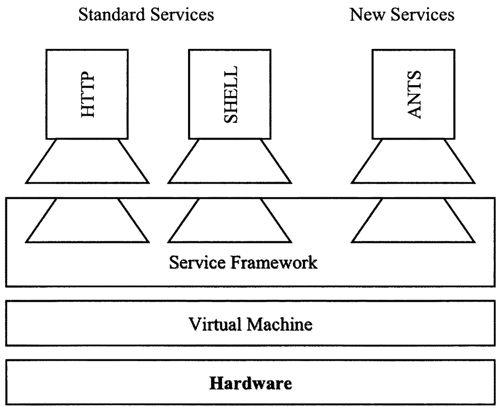

Fig. 3. Programmable service platform architecture.

The ORE and the forwarding API are but two major software components added to the service-platform.

C. Hardware Architecture

Fig. 4 illustrates essential hardware features of the architecture.

The hardware architecture consists of a control unit which is the control plane and a switch unit which is the data plane. It also includes multiple external units such as processing and storage that are connected to the switch unit through fast tunnels or ports.

A network processing unit consists of multiple network processors, some of which can be configured as processing engines. These networks processors are interconnected through a switch fabric. The switch fabric may employ crossbar, bus, ring, or banyan topologies. The main function is to process packet headers and switch packets from an input interface to an output interface. Processing engines can also perform additional processing, transformation on the packet before passing them to the output interface. A processing unit generally consists of a CPU, NPU, ASIC logic, and/or FPGA logic circuits. The choice depends on the particular application. The device promotes computation within the network and hence, is ideal for housing computational intelligence services such as content routing, personalization services [26]. The device is extensible in that additional network processing units can be plugged into the backplane of the device.

An external processing/storage device is a computing device with more powerful computational power and/or sizable short-term storage capacity. The control plane of the external processing/storage device has a similar set of functions to those in the control plane of the main unit. The device is connected to the main unit through a high-speed port. Packets are encapsulated with minimum overhead and send through the tunnel.

The reason for including external processing/storage devices in the architecture is two fold. It makes the architecture more scalable since the backplane of the main unit can only support a limited number of expansion slots. It also makes the architecture more secure since sensitive services can be configured and executed in physically separate components; no interferences between components are possible. Access to a physical port can be secured if the device attached to the port is authenticated and verified with appropriate signatures and access rights. The next section describes a particular realization of the design of the service platform deploying the Alteon Web switch.

V. OPENET-ALTEON PROGRAMMABLE SERVICE PLATFORM

In this section we describe the Openet-Alteon programmable service platform by way of mapping the generic design in Section IV onto a specific hardware and software architecture.

A. Openet-Alteon Hardware Architecture

The underlying hardware support for the new service platform is the Nortel Networks Alteon platform. It is a commercial multigigabit product integrating an Alteon switch with a set of network attached computation (NACs) units. The Alteon content switch offers a high-performance forwarding plane and a real-time content switching/filtering capability. The set of NAC units provides a rich control plane and a powerful computational plane for accommodating and processing sophisticated active services.

Alteon-Platform: Modern routers can filter, switch and manipulate L2, L3, and L4 traffic flows in real-time; but they are designed mainly for routing/switching. They are not equipped with huge processing power and/or adequate memory for processing intensive applications. The Alteon, however, is not designed as a normal router, but as a comprehensive L2 to L7 switch and hence, it is much more powerful as an edge device. The Alteon has in its architecture multiple processors, multiple ASIC devices, internal fast buses, and dual ARM core on each port running programmable micro codes. This allows the Alteon to be configured to perform L2, L3, L4 and application filtering and additional processing.

An NAC is essentially a computing element based on the Pentium compatible processor. Each NAC is connected to the Alteon switch through a high-speed port (tunnel). Packets are encapsulated with minimum overhead and sent through the tunnel. The Alteon platform is an integration of an Alteon switch and one or more NACs as shown in Fig. 5.

B. Openet-Alteon Software Architecture

The high-level software architecture of the Openet-Alteon programmable service platform is shown in Fig. 6. The Openet's two macro functional blocks, ORE, and forwarding API, now reveal their inner constituents.

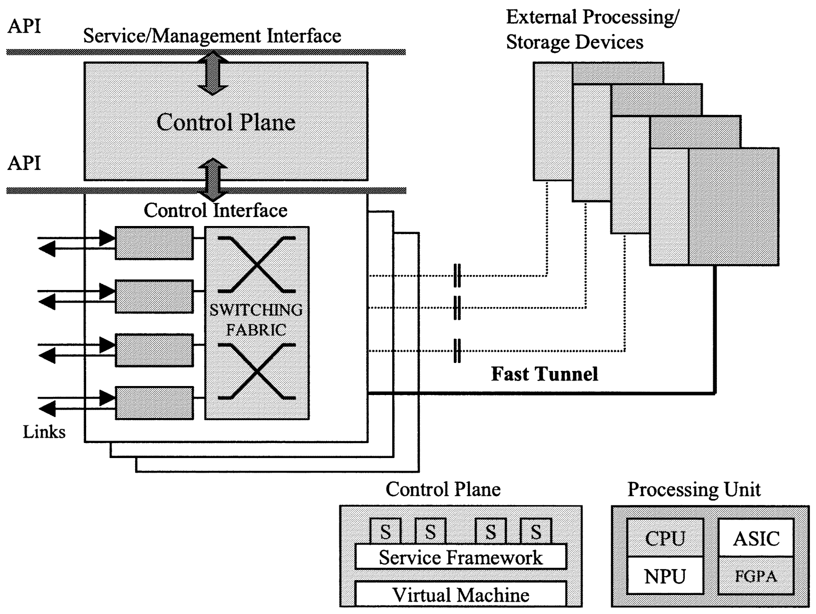

Fig. 4. Hardware architecture: essential features.

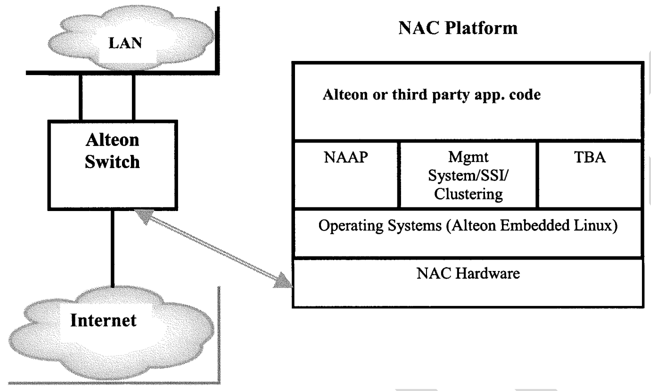
LAN
Alteon Switch
Internet
NAC Platform
Alteon or third party app. code
NAAP
Mgmt System/SSI/ Clustering
TBA
Operating Systems (Alteon Embedded Linux)
NAC Hardware

Fig. 5. Alteon platform.

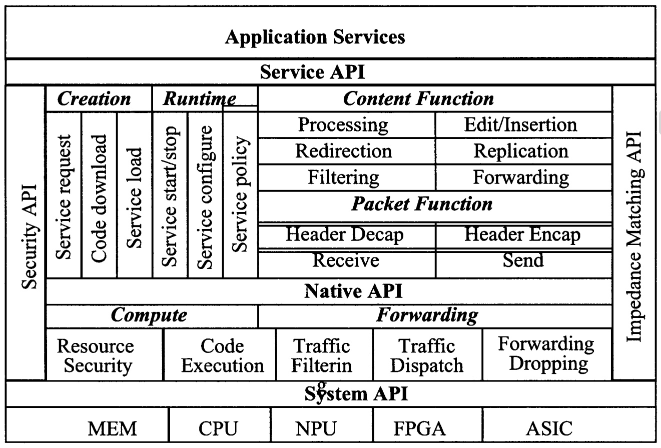

Fig. 6. Openet Alteon service platform architecture.

The Openet comprises an ORE and hierarchical services from low-level system to high-level application, and provides a neutral service-based programmability to network devices. In the architecture in Fig. 6, application services at the top are built by high-level service APIs. These service APIs provide applications the capabilities to create new services, execute existing services, and perform content-based functionality. Mapping into the service-centric architecture, various APIs appears as follows.

The service enabling interface proposed in Section IV is covered by the service runtime API and the service creation API. The service creation API is required for service creation, installation, and security. The service runtime API is required for computing or network resource reservation and allocation, configuration, signaling, and management.

The control interface consists of the Forwarding API, the content function API and the compute API. The forwarding API supports the regular networking functions such as packet filtering, deep packet inspection, diverting and forwarding/dropping, typically realized in hardware. Unlike traditional stateless router, the Alteon is a wire-speed stateful machine, providing new type of inspection capabilities. The stateful mechanism allows the memory (historical) for the duration of the stream, and not only packet by packet like traditional L3 router. The content function API provides the services to do various content manipulations, including content filtering, redirection, parsing, editing, insertion, replicating/multicasting, and forwarding. The compute API provides the functions required to configure the resources, read and write to memory or storage and resource-specific functions etc.

Part of the communication interface is the packet function API which receives and decapsulates individual incoming packets, encapsulates and sends outgoing packets, and sets up any-layer packet filters, packet data buffer allocation, data capture, data insertion, and tagging.

The management interface and the security interface are integral part of a network device dealing with general integrity, monitoring and configuring functionality of the device.

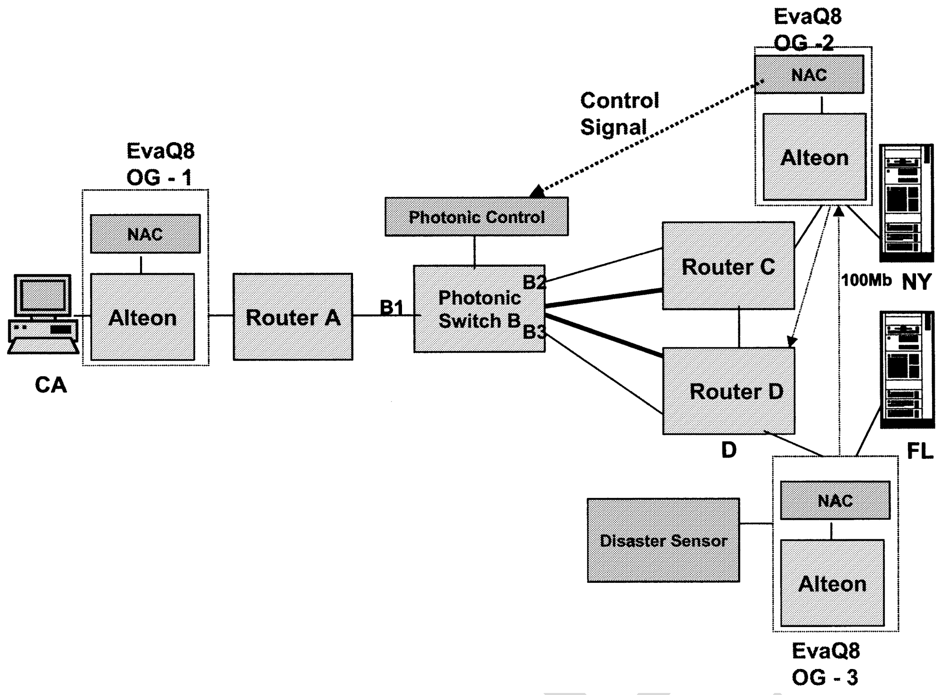

Fig. 7. Disaster recovery service demonstration.

The service APIs for service creation and runtime APIs are part of the Openet SDK (ODK) [23], [24]. The native API provides low-level access to native resources of computing and forwarding. The system APIs provide access to hardware or micro-code.

VI. Application Examples

This section presents four applications that have been implemented to demonstrate our programmable service platform.

A. *Disaster Recovery Service*

In spite of best current practices for on-schedule backups and business continuance, precious data may still be unrecoverably lost when adverse events such as fire, flood, explosions come to damage storage elements and network wires within a data-centric community. Important data includes information of critical business value, or highly sensitive nature. It also includes real-time data generated during the disaster, such as sensors' output, surveillance's output, network traffic digests, and any other last-minute "cyber-picture" of a disaster unfolding, so as to help survivors' rescue, to explain causes for a structure's collapse, etc. EvaQ8 [27] is an end-to-end solution that attempts to minimize the time needed to evacuate data out of a disaster area. It does so by allocating and using dedicated optical resources for the duration of the evacuation, across metropolitan or even wide area network distances. Within the end-to-end network, EvaQ8 is supported by automatically switched transport network (ASTN)-based agile, meshed optical networks, and by service-enable optical gatekeepers (OG), and by IP QoS in access ramp to the ASTN network. An OG is an edge device that has the capability to negotiate bandwidth on demand (as well as other services such as diversity routing, ad-hoc failure semantics, etc.).

Upon disaster, the OG closest to disaster requests an optical circuit to reach into the OG to which the safe-end disk-array is attached. The EvaQ8 is triggered by mechanism by which a disaster is sensed. Once the optical circuit is established, evacuation data get pushed into the ingress OG through the lower sped tributaries that connect the OG to the disaster area. The muxing resources in such ingress OG are all assigned to evacuation data with the highest priority.

We have demonstrated the proof of concept in the DARPA Active Networks Conference and Exposition in May 2002. Fig. 7 illustrates the demonstration set up of our disaster recovery concept. The sequence of events is as follows. The experimental results are shown in Table I.

1) Normal Application flow: Client CA [California]->Server FL [Florida];
2) Disaster strikes at location FL [Florida];
3) EvaQ8 OG 3 sends a signal [RSVP] to OG1;
4) OG1 instructs MEM Switch prototype to connect ports B2 and B3; Server FL and Server NY [New York] data synchronize;
5) On successful synchronization, OG2 instructs the MEM Switch prototype to connect B1->B2;
6) Service Restored for Client SF->server NY.

In the demonstration, our Openet-Alteon programmable service platform plays the role of the OG. Its control plane houses various mechanisms for EvaQ8 operation. The ORE environment manages various oplets (downloadable service units) such as active services download, service management, policy check, AFM flow selection, event request, and AFM action oplets. Fig. 8 illustrates components of an OG service platform for EvaQ8.

TABLE I
EvaQ8 NETWORK SETUP TIME

Time (ms)	Test 1	Test 2	Test 3	Test 4	Test 5	Average	Remarks
Alarm processing	0	0	0	0	0	0	Sensor
Disaster trigger	0	0	0	0	0	0	OG-3
Photonic control	2	1	2	1	1	1.4	Photonic Control
Photonic activation	12	12	12	12	12	12	Photonic switch B
Router control	1143	1145	1148	1154	1147	1147.4	Router C&D
Intercommunication	3	3	3	3	3	3	Inter-service signaling
SubTotal	1160	1161	1165	1170	1163	1163.8	Control time
Network update	12000	12000	14000	11000	16000	13000	STG update
Total	13160	13161	15165	12170	17163	14163.8	

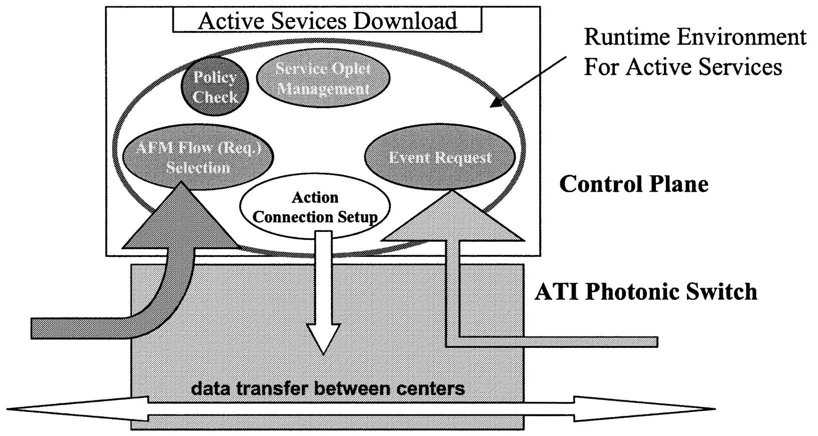

Fig. 8. Components of an OG service platform.

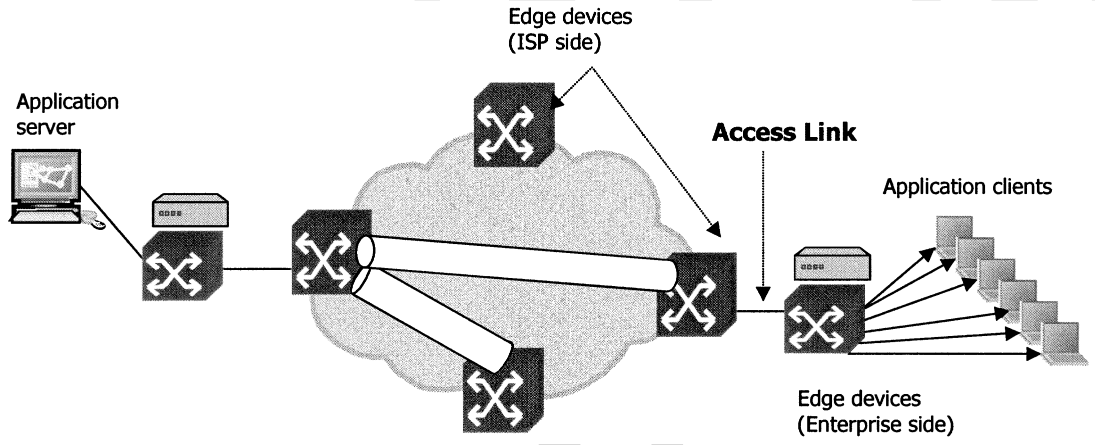

Fig. 9. General application layer multiunicast from edge device architecture.

B. Streaming Media Distribution Service

The aim is to deploy the Openet-Alteon service platform to perform multiunicast live video streams with QoS to multiple clients. Repeated video streaming to multiple users wastes a large amount of bandwidth, yet the access link to WAN is limited or expensive. It is desirable to avoid the bottleneck at the access link by shifting it to the enterprise's network where LAN bandwidth is abundant and inexpensive; or to the network core where bandwidth is more readily available.

In [28] an architecture that supports the application layer multicast model, but delegates any multicasting tasks to the edge devices, is proposed. The architecture allows the decoupling of an application from its underlying multicast mechanisms, hence permits rapid deployments of the application. By using intelligent edge devices with appropriate active services the bottleneck problem at the access link often encountered by ALM applications can be avoided. The working of the architecture is demonstrated through a real-life video streaming application. In our demo setup (Fig. 10), the streaming media distribution service (SMDS) service that resides on the NAC, intercepts all the requests that are made from a Real Player to a Real Server over RSTP and responds back to the Real Player as if it were the Real Player.

It separately makes the connection to Real Media Server and requests the media content, gets it and delivers it to the waiting

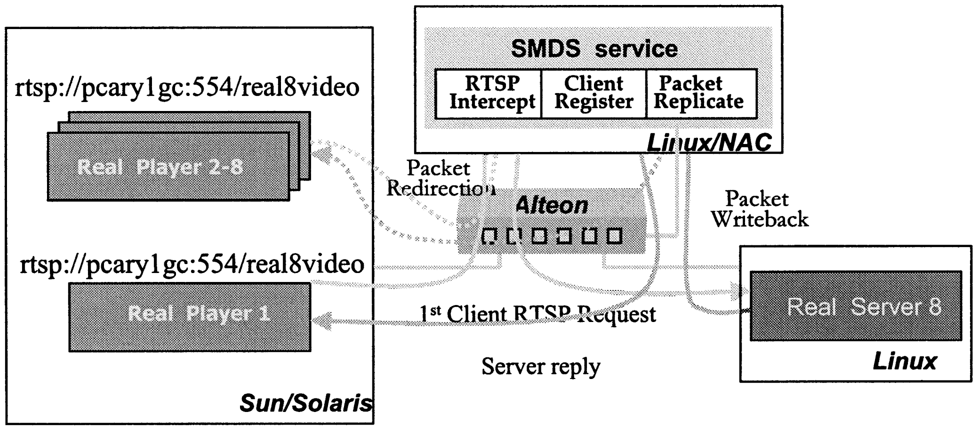

Fig. 10. Openet-Alteon service platform and media streaming application.

Real Media Player. Though this resembles a Real Proxy server function, the similarity ends here. For the next request that comes from another player for the same live broadcast content, instead of contacting the real player for another separate stream of content delivery, the application replicates the packets that it is already getting for the first session and sends a unicast copy back to the second Real Media player. The same thing is repeated for third request, fourth request and so on. In effect, only one copy of the media content is sent across the access link. The platform intelligently intercepts selective requests and operates on the data stream to produce a service equivalent to an application layer multicasting. See [28] for detail description of the experiment and measured performance. Brief description of the results is presented below.

An experiment was when all clients requested streaming at about the same time. The server had to serve multiple clients simultaneously; streaming data went directly to each of the clients without being intercepted by the Alteon. The results demonstrated the throughput was distributed unfairly across the multiple streams and that the bottleneck occurs at the access link. When more connections share the access link, the quality of the streaming video was degraded since some players did not receive their required bandwidth. With our service platform, the Real server sent only one stream to the NAC across the access link. The NAC then replicated the stream as necessary and sent them to each Real player across separate connections. The results showed that each client received the maximum fair share of bandwidth. The scheme worked by basically shifting the bottleneck to the enterprise end of the access link where bandwidth is more readily available.

C. Dynamic Content Adaptation Service

A wide choice of web access devices such as wireless phones, televisions, PDAs and PCs allows almost ubiquitous web connectivity. The wide ranging display capacities of these devices result in inefficient rendering of content originally intended for display on a PC monitor (typically a 15″–17″ SVGA screen). As the types of access devices in use increases, content providers are faced with the costly proposal of replicating content in multiple formats, a scenario that can quickly become a content management nightmare. Another attribute that varies among access technologies is the access bandwidth such as wireless access (from 9.6 kbps to 128 kbps), dialup modem (up to 52 kbps) and broadband access (in the order of Mbps). Large sized content, such as graphics-rich web pages, results in increased download times over slow access links. We present the dynamic content adaptation service (DCAS) that executes on the edge node and dynamically customizes content for presentation, personalization, or transportation.

The dynamic content adaptation service (DCAS) [20] is deployed on the Openet Alteon-platform at the edge of an internet service provider (ISP) access network or the content provider (enterprise) network. The service performs two functions: compression and content manipulation for presentation. Compression is based on the link speeds between the user and the ISP network. Content manipulation is based on the device display capacities. These pieces of information are made available to the service in the form of user profiles when the user subscribes to this service. The set up for service is similar to that for the SMDS service, however, the platform performs data compression and content manipulation rather than data replication in real-time.

We developed and tested the DCAS on our prototype testbed [20]. Apache web servers on PCs were used for the content provider. The clients were regular PC end systems. We deployed the DCAS on only one destination edge node in our experimental enterprise network while the node at the source edge simply acted as an IP router. For the initial proof of concept, the DCAS service only compresses JPEG graphics images.

D. Dynamic Allocation of Metro Bandwidth

It is here at the edge of the Metro core that our service platform again proves its worth in bridging the solution gap between the users and the Optical core through the Metro. Although the metro core is not optically agile, the service platform can intelligently intercept user request and intelligently provision bandwidth before user's traffic takes its course over the Metro's SONET. This effectively demonstrates that we can control the optical network based on the application requirements and the content by allocating bandwidth to user dynamically. We are able to set and remove a connection between any two points in metro network. The bandwidth for the connection can be dynamically set to any value between 1 Mb/s–1 Gbp/s in in-

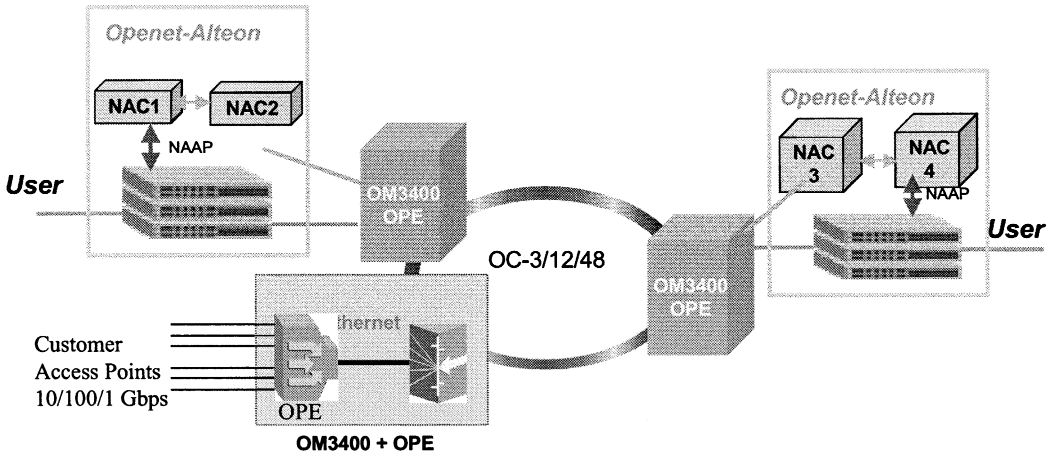

Fig. 11. Openet-Alteon platform at the metro edge demonstration.

crements of 1 Mb/s. The task involves setting and enforcing the allocated bandwidth for the end-to-end path, involving both the electrical side and the optical side.

For the Metro demo, the service platform, the Optera Metro 3400 with OPE are utilized. This device allows fine-grain Ethernet speed limits on top of Resilient Packet Ring (RPR) to be configured. RPR is currently in the definition process in IEEE 802.17. The distance limitation of this RPR is 120 km, but this distance can be extended to reach a RPR in a different geographical area via a DWDM point-to-point connection. Examples of the applications include: granting X Mb/s for the duration of a videoconference request; allocating Y Mb/s for Z minutes for a backup application between San Francisco office and San Jose office; advising and provisioning most economical backup for price sensitive applications. In this case the provisioning took 4.5 seconds and the signaling setup took 30 s. We used RSVP as our signaling and in the new version we needed to optimize this large signaling time.

Fig. 11 illustrates the demonstration set up of our dynamic bandwidth alocation in Metro networks.

It should be noted that the Metro network is not on the public Internet. It is secured, isolated, L2 network, without the limitations of L3 and packet switching. The connection bandwidth is guaranteed.

In another demo, we are able to show that we can set dynamically the network configuration of the all-optical switches based on application requirements, employing intelligent traffic filtering property of the platform.

VII. Conclusion

The paper has described the architecture, implementation, and utilization of the Openet programmable platform. Such platform is meant to enable the custom-defined services that define a new Internet service architecture. At the network edge, the platform realizes and/or supervises services such as nomadic support, alternate site failover, load balancing, security, etc. It does so without compromising on performances. As shown with the port of Openet to the Alteon commercial-grade device, the clean separation between forwarding tasks (best implemented in silicon) and control tasks (best implemented in software) yields a continuum of tradeoffs between sophisticated control and fast forwarding. Trivially, there is no overhead when there are no services defined. It is then up to the service code to incur overhead based on the complexity of the service (whereas the traffic extraneous to the service continues to flow unaffected through the device, at wire speed). Four real-life service scenarios validate the Openet platform, and the notion that the network edge is the place where active network services can be realized with maximum impact on the user experience.

References

[1] B. Raman *et al.*, "The SAHARA model for service composition across multiple providers," in *Proc. 1st Int. Conf. Pervasive Computing*, Zürich, Switzerland, Aug. 26–28, 2002.

[2] S. Karnouskos and A. Vasilakos, "Active electronic mail," in *Proc. ACM SAC* ***AU: PLEASE PROVIDE PAGE NUMBERS...........*** Madrid, Spain, Mar. 2002.

[3] D. L. Tennenhouse *et al.*, "A survey of active network research," *IEEE Commun. Mag.*, vol. 35, pp. 80–86, Jan. 1997.

[4] L. Peterson, Y. Gottlieb, M. Hibler, P. Tullmann, J. Lepreau, S. Schwab, H. Dandelkar, A. Purtell, and J. Hartman, "An OS interface for active routers," *IEEE J. Selected Areas Commun.*, vol. 19, pp. 473–487, Mar. 2001.

[5] D. Wetherall, "Active vision and reality: lessons from a capsule-based system," in *Proc. DARPA Active Networks Conf. Exposition*, San Francisco, CA, May 2002, pp. 25–39.

[6] N. Shalaby, L. Peterson, A. Bavier, G. Gottlieb, S. Karlin, A. Nakao, X. Qie, T. Spalink, and M. Wawrzoniak, "Extensible router for active networks," in *Proc. DARPA Active Networks Conf. Exposition*, San Francisco, May 2002, pp. 92–116.

[7] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. Kaashoek, "The Click modular router," *ACM Trans. Comput. Syst.*, vol. 17, no. 3, pp. 263–297, Aug. 2000.

[8] F. Kuhns, J. DeHart, A. Kantawala, R. Keller, J. Lockwood, P. Pappu, D. Richard, D. Taylor, J. Parwatikar, E. Spitznagel, J. Turner, and K. Wong, "Design and evaluation of a high performance dynamically extensible router," in *Proc. DARPA Active Networks Conf. Exposition*, San Francisco, CA, May 2002, pp. 42–64.

[9] P. Tullmann, M. Hibler, and J. Lepreau, "Janos: a Java-oriented OS for active network nodes," in *Proc. DARPA Active Networks Conf. Exposition*, San Francisco, CA, May 29–30, 2002, pp. 117–129.

[10] D. Mosberger and L. Peterson, "Making paths explicit in the scout operating system," in *Proc. 2nd USENIX Symp. Operating Systems Design Implementation*, Seatle, WA, 1996, pp. 153–167.

[11] H. Dandekar, A. Purtell, and S. Schwab, "AMP: experiences with building an exokernel-based platform for active networking," in *Proc. DARPA Active Networks Conf. Exposition*, San Francisco, CA, May 29–30, 2002, pp. 77–91.

[12] S. Meguru, S. Bhattacharjee, E. Zegura, and K. Calvert, "BOWMAN: a node OS for active networks," in *Proceedings of the 2000 IEEE INFOCOM*, Tel-Aviv, Israel, Mar. 2000.

[13] D. Wetherall, J. Guttag, and D. Tennenhouse, "ANTS: a toolkit for building and dynamically deploying network protocols," in *IEEE OPENARCH* ***AU: PLEASE PROVIDE PAGE NUMBERS.......***, San Francisco, CA, Apr. 1998.

[14] R. Braden, B. Lindell, S. Berson, and T. Faber, "The ASP EE: an active network execution environment," in *Proc. DARPA Active Networks Conf. Exposition*, San Francisco, May 2002, pp. 238–254.

[15] S. Bhattacharjee, K. Calvert, Y. Chae, S. Merugu, M. Sanders, and E. Zegura, "CANEs: an execution environment for composable services," in *Proc. DARPA Active Networks Conf. Exposition*, San Francisco, May 2002, pp. 255–272.

[16] J. van der Merwe, S. Rooney, M. Leslie, and S. A. Crosby, "The Tempest—a practical framework for network programmability," *IEEE Network*, vol. 12, pp. 20–28, May/June 1998.

[17] G. Hjalmtysson and S. Bhattacharjee, "Control-on-demand: an efficient approach to router programmability," *IEEE J. Select. Areas Commun.*, vol. 17, pp. 1549–1562, Sept. 1999.

[18] J.Jit Biswas *et al.*. Programming Interfaces for IP Networks. [Online] Available: http://www.ieee-pin.org

[19] The Network Processing Forum AU: PLEASE PROVIDE AN AUTHOR'S NAME............... [Online]Available: http://www.npforum.org/

[20] The Parlay Group AU: PLEASE PROVIDE AN AUTHOR'S NAME................ [Online] Available: http://www.parlay.org/

[21] Forwarding and Control Element Separation (forces) IETF Working Group AU: PLEASE PROVIDE AN AUTHOR'S NAME................ [Online] Available: http://www.ietf.org/html.charters/forces-charter.html

[22] T. Lavian, P. Wang, F. Travostino, S. Subramanian, D. Hoang, V. Sethaput, and D. Culler, "Enabling active flow manipulation in silicon-based network forwarding engines," *J. Commun. Networks* ***AU: PLEASE PROVIDE PAGE NUMBERS.............***, Mar. 2001.

[23] T. Lavian and P. Wang, "Active networking on a programmable networking platform," in *Proc. IEEE OpenArch* ***AU: PLEASE PROVIDE PAGE NUMBERS........***, Anchorage, AK, Apr. 2001.

[24] T. Lavian, R. Jaeger, and J. Hollingsworth, "Open programmable architecture for Java-enable network devices," in *Proc. Stanford Hot Interconnects* ***AU: PLEASE PROVIDE PAGE NUMBERS...***, Aug. 1999.

[25] S. Subramanian, P. Wang, R. Durairaj, J. Rasimas, F. Travostino, T. Lavian, and D. Hoang, "Practical active network services within content-aware gateways," in *Proc. DARPA Active Networks Conf. Exposition*. San Francisco, CA, May 29–31, 2002, pp. 344–354.

[26] A. Vasilakos, K. Anagnostakis, and W. Pedrycz, "Application of computational intelligence in active networks," *Soft Computing*, vol. 5, no. 4, 2001.

[27] I. Monga, B. Schofield, and F. Travostino. EvaQ8—abrupt, high-throughput digital evacuations over agile optical networks. presented at *Proc. 1st IEEE Workshop Disaster Recovery Networks* ***AU: PLEASE PROVIDE PAGE NUMBERS.......***. [Online] Available: http://comet.ctr.columbia.edu/diren/

[28] T. Lavian, P. Wang, R. Durairaj, D. B. Hoang, and F. Travostino, "Edge device multi-unicasting for video streaming," in *Proc. 10th Int. Telecommunications* ***AU: PLEASE PROVIDE PAGE NUMBERS...***, Tahiti, Hawaii, Feb. 2003, ICT2003.

Tal Lavian received the M.Sc. and B.Sc. degrees in electrical engineering and in computer science from Tel Aviv University, Tel Aviv, Israel, **AU: IN WHAT YEARS.** He is currently pursuing the Ph.D. degree in computer science at the University of California, Berkeley.

He is a Distinguished Scientist at Nortel Networks Advanced Technology Lab., Santa Clara, CA. He is also a DARPA Principal Investigator in the Computer Science Department, University of California, Berkeley, where he acts as Visiting Researcher. In the area of networking, he has authored 27 patents and has 18 years of experience in computer science and networking. His current research interests are grid computing, agile optical control, optical networks, and high bandwidth networking.

Mr. Lavian received the Top Inventor and Top Talent awards while at Nortel Networks.

Doan B. Hoang (M'90) received the Ph.D. degree in electrical and computer engineering from the University of Newcastle, **AU: PLEASE PROVIDE LOCATION.**

He is an Associate Professor of networking in the Department of Computer Systems, Faculty of Information Technology, University of Technology (UTS), Sydney, Australia. Before joining UTS, he worked in the Basser Department of Computer Science, University of Sydney, Sydney, Australia. He also held various visiting positions, i.e., Visiting Associate Professor, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Visiting Research Professor, Nortel Networks Technology Center, Santa Clara, CA, Visiting Assistant Professor in Computer Communications Network Group, University of Waterloo, Waterloo, ON, Canada, and Senior Research Fellow in the School of Electrical and Electronic Engineering, Nanyang Technological University, Nanyang, Singapore. His industrial experiences include Voice and Data Communications Manager in the Research and Development Department, Telecommunications, Fujitsu Australia Limited, Radiocommunications Engineer, Telecom Australia. **AU: WHAT IS THE NAME OF THE COMPANY? PLEASE CLARIFY. PLEASE ALSO PROVIDE THE LOCATIONS.** His research interests include active/programmable network platforms, quality of service (QoS) control mechanisms, Internet service architecture, peer-to-peer networks, mobile wireless Internet, and interesting network applications. He is the Director of the ARN networking research group within the UTS Advanced Research Institute for Information and Communcation Technology. He has published over 90 research papers in these areas.

Franco Travostino AU: PLEASE PROVIDE EDUCATIONAL BACKGROUND.

He has contributed to the operating systems and networking research communities for over 15 years. Prior to joining Nortel, he was running the networking group at the Open Software Foundation Research Institute. He served as DARPA Principal Investigator for six years, and produced several publications on real-time fault tolerance and active networks. At Nortel, he is the Director for the Content Internetworking Lab. in the Advanced Technology Investments division. His current interests include storage area networks, programmable networks, content intelligent networks, and peer-to-peer networking. In his career, he has had over 25 peer-reviewed publications and sat in the Technical Program Committees for several conferences.

Dr Travostino is the general chair for the IEEE Openarch 2002 event.

Phil Yonghui Wang received the Ph.D. degree in engineering from Tsinghua University, Beijing, China in 1990.

From Feb. 1998 and Jan. 2000, he was a Visiting Research Scientist with the Distributed Computing and Communications Lab of Computer Science Department, Columbia University, New York, and conducting research in distributed communications, resource control, QoS and active networks. In 1997, he was conferred with the Professor title of Computer Science by the Ministry of Electronic Industry, China. Since Jan. 2000, he has been with Nortel Networks, Ottawa, ON, Canada, and is currently working on open programmable networking technology, active networks, content networking, grid computing, intelligent optical networks, QoS and network management. His publications include over 40 articles and five books in a broad range of academic fields.

Siva Subramanian received the Ph.D. degree from North Carolina State University, Raleigh **AU: PLEASE PROVIDE THE FIELD OF STUDY.**.

is a Senior Architect and Manager for Advanced Technology with Nortel Networks, Raleigh, NC, where he conducts advanced research and contributes to the vision of Nortel Networks by planting mature differentiating technologies into next-generation products. His team has already contributed several innovations in programmable networking, content switching/processing, configurable computing, service architectures, and security to Nortel Networks business units. He has participated in panel discussions, given seminars at universities, and continues to conduct joint research with leading universities in the Research Triangle Park of NC, USA. Among other things, he has pioneered service-enabled networks and accelerated content processing work within Nortel Networks. Throughout a 10-year career, Siva has held technical and managerial positions within commercial product development groups as well as technology R&D groups. His breadth and depth in telecommunication and networking has enabled him to provide direction and advise on strategy within Nortel Networks as well as to external partners in the semiconductor industry. In the past 5 years, he has authored 5 peer-reviewed technical publications and ∼10 patent applications.

Inder Monga (M'98) recieved the B.S. degree in engineering from Indian Institute of Technology, Kanpur, India, and the M.S. degree in computer engineering from Boston University, Boston, MA, in 1990 and 1992, respectively.

He is a Consulting Engineer in advanced technology with Nortel Networks, Billerica, MA, and has over 10 years of industry experience in data networking including, IP routing, security, MPLS and data link technologies at Nortel Networks, bay networks and wellfleet communications. Current foci include protocols for control plane of dynamic transport networks, storage networking, data security, and disaster networks.