Network Programmability for VPN Overlay Construction and Bandwidth Management

Bushar Yousef [1], Doan B. Hoang [1], and Glynn Rogers[2]

[1]Advanced Research in Networking Laboratory, University of Technology Sydney, Broadway NSW 2007 Australia
{byousef, dhoang}@it.uts.edu.au
[2] CSIRO ICT Centre, Epping NSW 1710 Australia
glynn.rogers@csiro.au

Abstract. Reliability and security concerns have increased demand for Virtual Private Networks (VPNs). Ideally, a VPN service should offer autonomous overlay networks with guaranteed bandwidth allocations over a shared network. Network providers seek an automated VPN creation and management process, while users of a VPN would greatly benefit from secure control over the handling of their traffic. Currently, network infrastructure does not support such partitioning services and, due to its static nature, it cannot be adapted to meet such new demands. Active and Programmable Network research has developed a number of adaptable architectures. However, its current focus is on theoretical service deployment rather than on applicability to large and shared networking environments. This paper presents the application of a new programmable architecture to enable on-demand VPN construction, bandwidth management, and secure autonomous VPN control onto shared commercial infrastructure.

1 Introduction

There is constant demand for new and sophisticated network services such as resource-assured networks for video conferencing. To support new services, network elements must perform new and unsupported tasks. As a result, many task-oriented devices have emerged such as web-switch, SSL, firewall modules, and new routers to support new QoS models. As these devices are 'closed', network providers are required to purchase new devices to support new services. However, an inspection of the underlying hardware of these devices reveals that they can be combined and reprogrammed to support new services.

Research in active and programmable networks has developed novel architectures that 'open up' network devices to support user-defined services [1, 2]. Such architectures enable users to modify network behaviour by placing software components in the forwarding plane or customising specialised forwarding plane hardware. These architectures are not designed to leverage current commercial platforms because of their inflexibility. Yet, commercial platforms are essential for the deployment of any service architecture into the real-life networking environments of the Internet.

Recently, there have been increasing demands on ISPs to offer Virtual Private Network (VPN) services across the Internet. Conceptually, each VPN represents an autonomous, resource-assured, secure, and customisable network over shared infrastructure for each user (i.e. VPN owner). Such a VPN requires partitioning network bandwidth and providing users with the ability to secure and customise their VPNs. These are features beyond the support of current Internet infrastructure and, therefore, currently offered VPNs that fall short of expectation. A current VPN is either: a simple point-to-point encrypted tunnel with best-effort delivery; or a static overlay network of Service Level Agreements manually configured by ISP operators. The former type only addresses the security requirement of a VPN, while the later has a number of shortcomings. Firstly, an ISP cannot create or modify VPNs on-demand due to the long and manual setup procedure. Secondly, at the core of a network, bandwidth assurances can only be partially enforced for a large number of VPNs. Thirdly, users cannot customise or deploy services in their VPNs.

This paper presents a new model to support VPNs and their on-demand provisioning over shared commercial infrastructure. This model utilises a new programmable architecture called Secure, Extensible, and Deployable-Programmable Network Platform (SXD-PNP). SXD-PNP is used to deploy VPN support services onto current commercial modules, and to partition network nodes into customisable User Partitions for each VPN. VPN support services enable network providers to construct and manage customisable and resource-assured VPNs on-demand. Each VPN is allocated a series of SXD-PNP User Partitions, which enable VPN-owners safe autonomous control and secure path construction by deploying their own services within their partitions. To manage a number of VPNs across shared nodes, User Partitions separate internal resources and use traffic classification mechanisms to restrict their configurations to a permissible set of traffic. SXD-PNP employs a new differentiated allocation model called Control-plane Quality of Service (C-QoS) that manages internal node resources among competing partitions and among competing services within a partition.

This paper is structured as follows. Section 2 provides an outline of SXD-PNP and its implementation focusing on features that enable VPN provisioning. Section 3 presents the VPN support services. Section 4 describes an Edge-to-Edge QoS mechanism that provides VPN resource guarantees across the network core. It also discusses our experiences with its deployment onto current network devices. Section 5 gives a brief discussion of related works. The paper concludes in section 6.

2 SXD-PNP Overview

SXD-PNP is a flexible programmable router architecture that enables on-demand service deployment. SXD-PNP services modify the handling of traffic. This is achieved by configuring the forwarding hardware or by hosting the execution environments found in active networks. SXD-PNP is an ideal service architecture to deploy VPN support services and to facilitate VPN partition and customisation. This is due to its features of: QoS guarantees on internal node resources, isolation of users, traffic security enforcement, and commercial module utilisation.

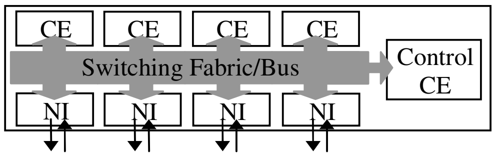

Fig. 1. Hardware Base Abstraction.

SXD-PNP builds on an abstract node model called the Hardware Base Abstraction (HBA). The HBA, depicted in figure 1, represents a switching platform composed of a configurable realtime forwarding plane and separate extensible control plane. The forwarding plane is composed of Network Interfaces (NIs) represent the realtime hardware networking modules. These perform network traffic classification, forwarding, manipulation, and scheduling operations. The control plane is composed of Computational Elements (CEs) and a Control-CE. CEs accommodate User Services that construct Active Flow Manipulation [3] Paths (AFMP) among NIs to modify traffic handling behaviours. Operations beyond NIs capabilities are performed in the control plane by redirecting AFMPs to User Services that can use FPGA hardware to optimised packet processing. The Control-CE performs User Partition setup, security and C-QoS configurations, service deployments, load monitoring, and load balancing operations. All NI and CE components are interconnected by a high-speed communication bus. Both planes can be expanding by adding components to the bus.

Hosting a number of competing VPNs on shared network infrastructure requires all network resources to be partitioned amongst the VPNs. This involves partitioning network bandwidth and each customisable node along its control and forwarding planes. To partition link bandwidth, SXD-PNP deploys and configures bandwidth management services which enforce Edge-to-Edge Resource Discovery and Admission Control mechanisms [4]. At each customisable node (SXD-PNPs), the control plane is divided into separate, resource-assured, and secure User Partitions. These nodes allocate a User Partition to each VPN. The forwarding plane is partitioned among User Partition by restricting each partition from performing flow manipulations on traffic outside of its allocated VPN. The next subsection describes SXD-PNP control plane partitioning. Subsequent sections discuss the mechanisms that enforce this traffic access restriction (2.2) and the bandwidth partitioning mechanism (4).

2.1 User Partitions

A CE, depicted in figure 2, is composed of User Virtual Machines (UserVMs), a System Process, and a Bus Management Process. Each VPN is allocated a UserVM process with assured resources and access to AFM [3] on its traffic. This represents a User Partition. A UserVM manages a number of Runtime Environments (REs) that execute the User Services supplied by VPN owners. To ensure secure partitioning, UserVMs are restricted from hardware access and communications with the exception of through the System Process. The System Process guarantees to restrict UserVMs to their partitions. Details on service interaction and AFMP construction are in 2.2.

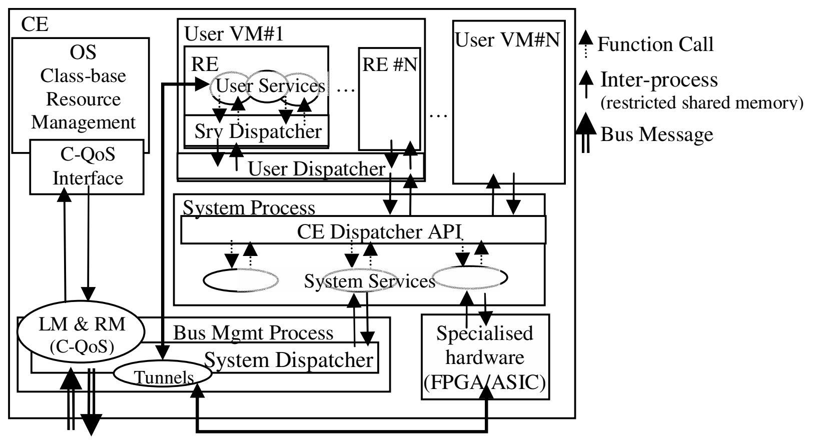

Fig. 2. Computational Element.

The division and allocation of all internal node (control plane) resources among User Partitions are performed through C-QoS. User Partitions are classified into one of four C-QoS classes - Gold, Silver, Bronze, or Best Effort. C-QoS performs differentiated per-class allocation and fair allocation for partitions within each class. Due to the difficulty of determining user resource requirements or their availability in heterogenous infrastructure, C-QoS uses a dynamic allocation model - allocating resources from lower classes to an upper class as its load increase. Dynamic allocation is performed until a specified lower limit is reach. This limit is used to ensure a minimum level of resource availability to partitions. To allocate and reclaim resources, C-QoS utilises Resource Managers that interface with the Operation System (OS) resource management mechanisms. C-QoS Load Monitors track resource loads at CEs to balance UserVM load among CEs.

To maintain node integrity, a certain amount of resources is reserved before allocating resources among User Partitions. The aggregate capacity of links is reserved over the bus to ensure NI-to-NI (network) traffic is never delayed as a cause of service activity. All SXD-PNP management tasks and their messages are classified into a special *Realtime* class. *Realtime* resource requests are granted before the requests of other classes. This ensures that congestion will not affect the security or partitioning of a node.

C-QoS categorises resources as Computational or Internal Communication.

Computational resources are the traditional OS controlled resources of multi-user systems namely CPU scheduling, memory heap restrictions, I/O scheduling, and harddisk quotas. These resources are managed within each CE independently by its OS. The OS allocates resources to UserVMs in proportion to C-QoS classification and it restricts UserVMs to their allocation. The allocation details for each computational resource for each class are specified by the Control-CE and communicated via the C-QoS interface to the OS resource management mechanisms. The implementation, in section 2.3, interfaces to a modified Linux kernel.

C-QoS extends its reach to differentiate services within a partition. UserVM resources are divided among user-defined RE-subclasses into which REs are classified. REs further differentiate between User Services by classifying services into Service-subclasses. This structure permits N^3 levels of differentiations between services, where N is the number of classes. As UserVM resources belong to the same VPN, to avoid strict management overhead only 'soft' resource management mechanisms are employed for service differentiation. A UserVM uses thread priorities to differentiate REs and weighted thread slot scheduling in REs to differentiate User Services. SXD-PNP provides three RE types, each with a different scheduling model to cater for variations in service response time and allocation size requirements.

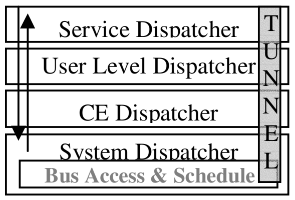

Fig. 3. Dispatcher Hierarchy.

Internal communication resources are represented by a hierarchy of dispatchers, depicted in figure 3, which handle all internal communication. Scheduling mechanisms within dispatchers are used to divide and allocate internal communication resources among partitions. Computational and bus resources are allocated to dispatchers and dispatcher time is then divided into slots which are allocated to partitions or child dispatchers according to a dynamic class ratio. The bus is divided into channels or time slots, depending on the technology in use. For identification (used in traffic access restriction) and bus resource management, each partition is restricted by System Level Dispatchers to an allocated channel or timeslot(s).

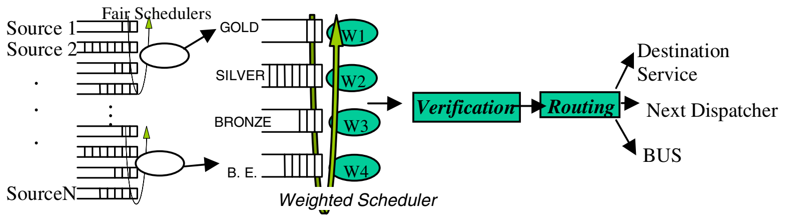

Fig. 4. Dispatcher Structure.

Dispatchers share a common structure, depicted in figure 4. However, dispatchers differ in their verification scope and routing destinations depending on the hierarchy level. Each incoming message is placed in a limited queue allocated to its source. Source queues of the same class are serviced by the same fair scheduler that moves messages to the corresponding class queue. Messages in class queues are serviced by a scheduler weighted with the C-QoS class allocation parameters. Each serviced message is removed from the queue, verified as in figure 5, and routed to its next hop.

2.2 Network Abstraction and Traffic Partitioning

User Services construct AFMPs to configure new routing and manipulation operations on their traffic. An AFMP is composed of combination manipulation points, typically NPUs and CE specialised hardware modules, but may also include User Services for control flows. Each manipulation point along an AFMP is configured to filter ingress traffic into flows, perform specified manipulation operations on flows, and route flows to other manipulation points or onto the network.

User Services construct AFMPs by sending configuration messages to System Services. System Services are located at CEs to abstract the NPU configuration interfaces from User Services. System Services then send NI specific commands on the User Services behalf to each NI manipulation point. These commands are sent to the system dispatcher as message originating from the User Services partition.

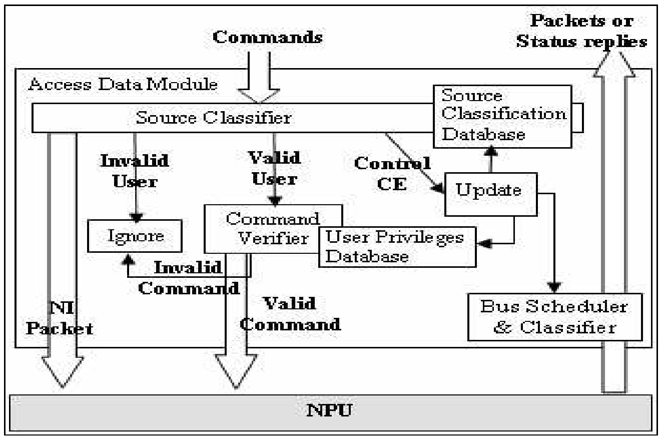

Fig. 5. NI Structure.

An NI is divided into two logical components, depicted in figure 5, the Access Data Module (ADM) and the Network Processing Unit (NPU). NPUs represent the configurable traffic forwarding and manipulation hardware. These NPUs operate at wire-speed with no modification by SXD-PNP. ADMs ensure partitions only perform operations that affect their VPNs by verifying all user configuration commands to NPUs. This model partitions the forwarding plane without placing VPN classification or security checking in the path of network traffic.

At an ADM, each message arrives at the Source Classifier that identifies the type of each message by its source. This is established by the bus channel on which the message arrived. NI messages are traffic packets and are immediately passed to the NPU. Control-CE messages are configuration messages that update source classifications, User Partition-to-VPN associations, or the bus scheduling details. UserVM messages are verified by a Command Verifier before being forwarded to the NPU. The Command Verifier ensures that the flow filter of any forwarded configuration command falls within the Access Control List of its source partition's VPN.

2.3 Implementation

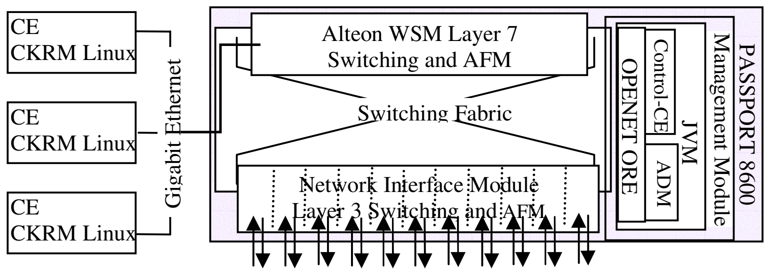

Fig. 6. SXD-PNP Implementation.

Figure 6 show the implementation of SXD-PNP a Passport 8600, an Alteon Web Switch Module, and a number of PCs. The Control-CE and ADM have been implemented in the Management Module's Java Virtual Machine. System Services have been implemented to interface with the manufacturer's Oplet Runtime Environment (ORE) [5] that configures AFM on its hardware modules. UserVMs are implemented as JavaVMs that initiate a specialised Java Security Manager and UserVM thread to manage control requests. UserVMs are hosted across a number of CEs that are implemented on PCs running a Linux kernel 2.6.7 patched with the Class-base Kernel Resource Management (CKRM) [6]. CKRM enables differentiated class-based resource management on CPU, memory pages, and I/O. A C-QoS interface to the CKRM file-system was implemented to enable the Control-CE to create classes, configure their allocation size, and to classify UserVMs into classes.

3 VPN Construction and Management

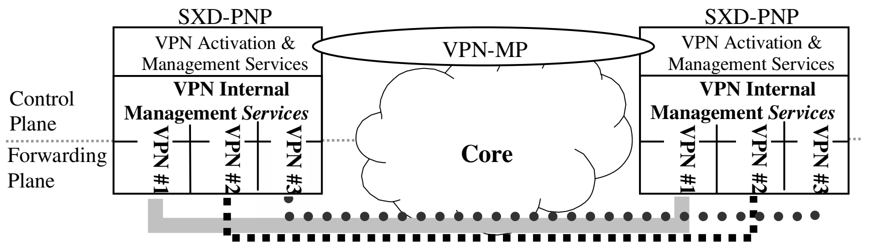
SXD-PNP
VPN Activation & Management Services
VPN Internal Management Services
VPN #1
VPN #2
VPN #3
VPN-MP
Core
SXD-PNP
VPN Activation & Management Services
VPN Internal Management Services
VPN #1
VPN #2
VPN #3
Control Plane
Forwarding Plane

Fig. 7. SXD-PNP VPN Structure.

This section presents the SXD-PNP VPN Construction and Management services depicted in figure 7. They consist are of sets of services. The first, VPN Activation & Management Services, construct and manage VPNs. The second, VPN Internal Management Services, enable user autonomous control over their VPN.

3.1 VPN Activation & Management Services

VPN Activation & Management Services are SXD-PNP System Services that enable administrators to create, monitor, and modify VPNs across a network. These are composed of a distributed Network Division Service (NDS), and three local node services: a VPN Classification Service; a Bandwidth Management Service; and, a User Environment Service. These services are placed in Control-CEs and a NDS is placed at each node along VPN paths. NDSs collectively utilise the three local services to perform VPN activation and management.

Inter-NDS communication is achieved via an activated flow that spans all NDSs, called a VPN Management Protocol (VPN-MP) channel. A network administrator uses the VPN-MP channel to setup, monitor, and modify VPNs. VPN-MP employs security mechanisms to restrict NDS access to network administrators.

To perform VPN activation and management operations, network administrators construct a single VPN-MP *Request* and send it to the ingress edge SXD-PNP. This request is sent across the network and captured by NDSs along the *Paths* parameter. Each NDS fulfils the request, tightens the filters, and propagates it to nodes along the *Paths.* Filters are tightened using routing table information and subnet masks to eliminate filter ranges inapplicable to the current node or to nodes remaining in the propagation path. The VPN-MP Request structure is as follows:

1. *Paths* - Source and Destination network address combinations of all paths of the VPN.
2. *Instruction Type* – 0/setup 1/modify 3/remove 4/list details
3. *VPN Number* – 0 for new VPNs or operation on all VPNs
4. *Credentials* - User ID and Encrypted Password
5. *VPN ACL* (Access Control List) - List of Traffic Filters
6. *Network QoS class*
7. *C-QoS class*
8. *Signature* – MD5 hash encrypted using admin private key

To fulfil a VPN-MP Request each NDS uses the three localised services. The *User Environment Service* sets up a SXD-PNP User Partition according to *C-QoS class*. The *VPN Classification Service* configures the appropriate NIs to classify traffic into a VPN and configures their ADMs to restrict configuration access to the associated User Partition. The *Bandwidth Management Service* configures the link bandwidth for the VPN according to *Network QoS class*.

The *User Environment Service* uses C-QoS Load Monitors to locate the least loaded CE for new UserVM deployment. This service then sends a Control-CE request to Management Processes of the least load CE. This request creates a new UserVM with the *C-QoS class* specified. Once the UserVM is started, it is instructed to deploy the *VPN Internal Management Services*. These services are configured to restrict access according to the *Credentials* specified in the request.

The *VPN Classification Service* utilises ORE System Services to configure appropriate ingress NPUs to classify traffic according to the *VPN ACL* and mark its traffic with the *VPN number*. It also configures ADMs to allow the newly created UserVM to configure operations on traffic marked with *VPN number*.

The *Bandwidth Management Service* partitions network link resource among VPNs by configuring the *Edge-to-Edge QoS Services*, which are presented in section 4.

3.2 VPN Internal Management Services

VPN Internal Management Services are User Service running within each User Partition to provide users with a configuration interface to the routing and manipulation operations on their VPN traffic. They are composed of a *VPN Command Interface Service*, an *AFMP Construction and Management Service* that is used by two specialised services providing VPN specific features: a *VPN Overlay Management Service* and a *Secure Path Construction and Management Service*.

The *VPN Command Interface Service* provides a remote interface for users to interact with their User Services. It provides a command-line interface with username and password authentication (the *Credential* in VPN-MP request). It parses user instructions and parameters, calls the appropriate service, and displays the results. We plan to develop this service to be accessible via an active path similar to VPN-MP, enabling users to configure all nodes in a VPN with one request.

The *AFMP Construction and Management Service* provides the user and other VPN Internal Management Services with the capability to construct, monitor, modify, and remove AFMPs. It allows users to specify an AFMP Number, flow classification filters, and a sequence of actions to be performed on the flow. To construct a AFMP, this service uses the ORE System Service. Firstly, it locates a NI that is capable of performing the specified action and configures it. It then configures redirection and classification operations, to direct the path from the ingress NI, to the action NI, and finally to the egress NI. It also maintains a database to track AFMPs and their associated filters and actions, facilitating quick removable or modification of paths.

The *VPN Overlay Management Service* enables users to modify routes within their VPN by adding or removing route entries. For a new entry, it constructs an AFMP to redirect VPN flows to new destinations. It maintains a viewable table of routes, and their AFMPs. To remove a route, this service removes its table entry and AFMP.

The *Secure Path Construction and Management Service* enables users to secure sensitive flows in their VPN. As selective encryption at user hosts leads to security holes, this service allows VPN users to guarantee security encryption mechanisms on critical traffic at the network layer. It also allows the sharing of costly and specialised equipment that guarantees high-speed encryption and decryption. This service sets up AFMPs to redirect sensitive flows to encryption accelerators at ingress edge nodes, and to decryption accelerators at egress edge nodes.

4 Bandwidth Management

Networks must enforce bandwidth management mechanisms to partition link resources among VPNs. Current bandwidth management is performed by a DiffServ [ref] model that employs proprietary QoS mechanisms locally at each router. These mechanisms classify traffic at ingress into QoS classes by placing packets into a queue for each class. A Weighted Round-Robin algorithm services these queues, spending more link resources on higher-class queues than lower queues.

To provide bandwidth resource partitioning among VPNs, traffic of each VPN is classified into a single flow that is placed in a QoS class. However, this partitioning cannot be mapped to the DiffServ model for two reasons:

1. *Per-flow fairness* – In DiffServ, same class flows are placed in the same queue where packets are randomly dropped as the queue reaches its limit size. This model does not treat all flows within the same class fairly. Therefore, it cannot be applied to VPNs, as VPNs of the same class do not get equal treatment.
2. *Congestion Control* – DiffServ does not employ admission control mechanisms to prevent uncontrolled flows from causing congestion. Therefore, a VPN can affect the resources of others by causing congestion.

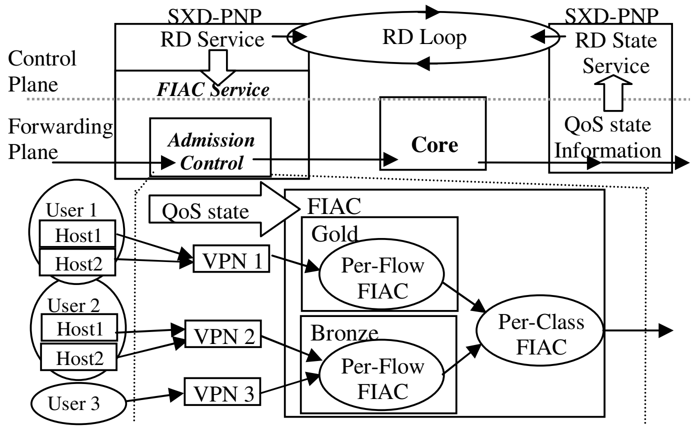

Fig. 10. Bandwidth Management Services.

SXD-PNP uses Ed*ge-to-Edge QoS Services* to partition bandwidth fairly among VPNs and prevent congestion. These services configure existing QoS router mechanisms to enforce a Fair Intelligent Admission Control (FIAC) [4] scheme, as depicted in figure 10. It uses a Resource Discovery (RD) feedback loop to gather the congestion state information of the network. At each ingress node, an Admission Control Module (ACM) reconciles with available resources via the RD loop, and admits traffic intelligently according to the FIAC algorithm. The FIAC algorithm guarantees to admit traffic according to class weights while providing fairness among flows, and, based on the RD state report, prevent congestion by dropping packets at ingress intelligently.

Two services implement the RD feedback. At edge nodes, we implement an *RD Service*, which sends and receives "RD packets" along the feedback loop of VPN paths. These packets are captured by a *RD State Service* at each node in the path where the packets are updated with the congestion report. A demonstration of the RD loop implementation can be found in [7].

At edge nodes, the ACM is implemented through a *FIAC Service*. This service configures ingress NI to classify VPN traffic into flows and to classify the flows into its appropriate QoS class. It periodically uses state information supplied by *RD State Service* to configure NI queue lengths and drop rates to prevent congestion.

FIAC required per-flow fairness mechanisms at NIs. Unfortunately, most networking modules manufactured today are DiffServ compliant and do not provide a scalable per-flow management model. To achieve per-flow fairness in our implementation, presented in 2.3, we construct a static AFMP to redirect flows to the Web Switch Module where Load Balancing mechanisms perform flow management. However, this approach is still in initial design and testing stages.

The *Bandwidth Management Service* presented is section 3.1 uses the *Edge-to-Edge QoS Services* to partition the network link resources among VPNs. It instructs the *RD Service* to initiates feedback loops along VPN paths, if loops do not already exist. The *FIAC Service* is updated to treat the VPN traffic as a separate flow and classify the flow into the *network QoS class* specified in the VPN-MP Request.

5 Related Work

The extensive research in the field of Programmable and Active Networks has developed a number of flexible and relatively secure network service architectures [1, 2]. These architectures have not addressed the partitioning of access to traffic among users/VPNs, while few addressed the allocation and management of internal node resources.

Architectures [8-10] provide mechanisms to explicitly allocate each resource to services, collection of services, or paths according to a specified or estimated requirement. However, this fine-grained model is it is not scalable to a large number of heterogeneous users and is impractical as it depends on prior knowledge of resource requirements.

Due to these deficiencies, separate projects have been conducted to address active node resource management [11-13]. These projects throttle the input of packets to services to control their resource consumption. Their models hinge on the difficult task of pre-determining or estimating the resources consumed by each packet. Furthermore, these projects do not manage memory or internal communication resources. They also do not account for services resource consumption outside of packet input influence.

6 Conclusion

We have presented a new platform programmability architecture, SXD-PNP, that allows practical, secured, and true partitioning of network resources on commercial devices. We deployed this architecture to construct programmable VPNs, address difficult issues such as VPN traffic classification, bandwidth management, user service deployment, and secure path construction. We described a solution that enables a VPN on shared infrastructure as close as ever to a privately owned WAN.

Currently, VPN support service and bandwidth management services are partially implemented on our SXD-PNP testbed. Performance evaluation and VPN separation analysis on a complete implementation will be presented in future publications.

It is anticipated that the SXD-PNP will be deployed both in the network core and at the network edge. As a core router, it can be used to partition network resources into separate and secured user domains (such as VPNs), and allow operators to introduce services for the timely resolution of various traffic-engineering problems.

SXD-PNP is most useful when used in an edge device where all designed features can be deployed. A service provider can rapidly introduce new services to address the mismatches between domains in terms of network boundary, technology, and administration. Our next step is to deploy the platform for constructing overlay networks where network resources can be safely partitioned and shared in Grid service environments.

References

1. Campbell, A., H. De Meer, M. Kounavis, K. Miki, J. Vicente, D. Villela, *A Survey of Programmable Networks*. ACM SIGCOMM Computer Communications Review, 1999. **29**(2): p. 7-23.
2. Gottlieb, Y., L. Peterson, *A Comparative Study of Extensible Routers.* IEEE Open Architectures and Network Programming Proceedings (OPENARCH), 2002.
3. Lavian, T., P. Wang, F. Travostino, S. Subramanian, D. Hoang, V. Sethaput, D. Culler, *Enabling Active Flow Manipulation in Silicon-based Network Forwarding Engines.* IEEE Journal of Communications and Networks, March 2001.
4. Li, M., D. B. Hoang, A. J. Simmonds, *Fair Intelligent Admission Control over Differentiated Service Network.* The Computer Communication Journal: the Special Issue on Quality of Service, 2004.
5. Nortel Network's Openet Lab, http://www.openetlab.org/.
6. Class-based Kernel Resource Management (CKRM), http://ckrm.sourceforge.net/.
7. Hoang, D., T. Lavian, I. Zhao, C. Nguyen. *Implementation of a Quality of Service Feedback Control Loop on Programmable Routers*. *IEEE International Conference On Networks (ICON'04)*. 2004. Singapore.
8. Shalaby, N., L. Peterson, A. Bavier, G. Gottlieb, S. Karlin, A. Nakao, X. Qie, T. Spalink, and M. Wawrzoniak, *Extensible Router for Active Networks.* Proceedings of DARPA Active Networks Conference and Exposition, 2002: p. 92-116.
9. Tullmann, P., M. Hibler, J. Lepreau, *Janos: A Java-Oriented OS for Active Network Nodes.* Proceedings of DARPA Active Networks Conference and Exposition, 2002: p. 117-129.
10. Braden, R., B. Lindell, S. Berson, T. Faber, *The ASP EE: An Active Network Execution Environment.* Proceedings of DARPA Active Networks Conference and Exposition, 2002: p. 238-254.
11. Qie, X., A. Bavier, L. Peterson, S. Karlin. *Scheduling Computations on a Software-Based Router*. in *Proceedings of the ACM SIGMETRICS 2001*. 2001.
12. Ramachandran, V., R. Pandey, S-H. G. Chan. *Fair Resource Allocation in Active Networks*. in *Proceedings of the IEEE International Conference on Computer Communications and Networks (ICCCN)*. 2000. Las Vegas, Nevada.
13. Pappu, P., T. Wolf. *Scheduling Processing Resources in Programmable Routers*. in *Proc. of the Twenty-First IEEE Conference on Computer Communications (INFOCOM)*. 2002. New York, NY.