2004

# Resilience-differentiation in programmable virtual networks

F. Rosenbaum
*University of New South Wales*

S. Jha
*University of New South Wales*

P. Boustead
*University of Wollongong*, boustead@uow.edu.au

Farzad Safaei
*University of Wollongong*, farzad@uow.edu.au

# Resilience-differentiation in programmable virtual networks


## Abstract

Service and application requirements on network resilience have increased over the past few years. New on-line services such as e-commerce and connection-oriented interactive real-time services require higher network resilience than the more traditional off-line services. Programmable virtual networks promise fast and easy provisioning of new services but no consideration to meet the diverse resilience requirements has been made. This paper discusses issues related to resilience-differentiation in programmable virtual networks. A set of general guidelines is presented that apply to resilience-differentiation in programmable virtual network architectures. A case study is used to illustrate how the proposed guidelines can be met by extending an existing programmable virtual network architecture.



## Disciplines

Physical Sciences and Mathematics

## Publication Details

This paper originally appeared as: Rosenbaum, F, Jha, S, Boustead, P & Safaei, F, Resilience-differentiation in programmable virtual networks, 2004 IEEE International Conference on Communications, 20-24 June 2004, vol 4, 2117-2121.

# Resilience-Differentiation in Programmable Virtual Networks

Filip Rosenbaum and Sanjay Jha
School of Computer Science and Engineering
University of New South Wales, Australia
Email: {filipr, sjha}@cse.unsw.edu.au

Paul Boustead and Farzad Safaei
Telecommunications and Information Technology
Research Institute
University of Wollongong, Australia
Email: paul@titr.uow.edu.au; farzad@uow.edu.au

***Abstract*— Service and application requirements on network resilience have increased over the past few years. New on-line services such as e-commerce and connection-oriented interactive real-time services require higher network resilience than more traditional off-line services. Programmable virtual networks promise fast and easy provisioning of new services but no consideration to meet the diverse resilience requirements has been made. This paper discusses issues related to resilience-differentiation in programmable virtual networks. A set of general guidelines is presented that apply to resilience-differentiation in programmable virtual network architectures. A case study is used to illustrate how the proposed guidelines can be met by extending an existing programmable virtual network architecture.**

## I. Introduction

Network service survivability, or resilience, has gained increasing interest over the past few years. The main reason is that new on-line services such as e-commerce and connection-oriented interactive real-time services require ever-increasing network availability to meet their users' expectations while other off-line services such as email have low resilience requirements. The diverse resilience demands add new requirements on the network infrastructure. To increase efficient utilisation of network resources, recovery scheme design should take the different resilience requirements into account. Traffic engineering methods [1] are a requirement to efficiently provision resilience in a network. Recovery mechanisms can be implemented at multiple network layers. They can even be in operation simultaneously. The authors of [2] present some guidelines on where to implement recovery mechanisms and how to co-ordinate them if they are applied in more than one layer. In general, lower layers such as wavelength-division multiplexing (WDM) and synchronous digital hierarchy (SDH) provide very fast recovery while higher layers like IP and multiprotocol label switching (MPLS) provide higher resource efficiency, flow and quality-of-service (QoS) granularity.

Programmable virtual networks are promoted as a solution for fast and easy provisioning of new innovative services over the Internet. The basic idea is to provide multiple programmable virtual private networks (PVNs) over one physical infrastructure. Each virtual network customer (VNC) can install and run customized code in virtual nodes to control routing and support application specific tasks, such as content adaptation and monitoring. A number of different programmable virtual network architectures have been proposed over the past few years, for example Tempest [3], Virtual Active Network [4] and the Programmable Virtual Network architecture [5]. However, the resilience issues related to a programmable virtual network infrastructure have not been addressed.

End-to-end resilience differentiation in overlay infrastructures introduces a new dimension, as the end-to-end resilience provisioning can be split between the network provider and its VNCs. Some of the challenges to provide resilience differentiation in programmable virtual networks lie in the fact that a VNC controls its own routing. Furthermore, it is important to uphold PVN integrity and privacy. Traffic must never leak out of or into a PVN. Scalability issues must be considered as well. A programmable virtual network infrastructure can potentially provide hundreds of PVNs, each supporting thousands of end-to-end flows.

This paper addresses general resilience differentiation issues related to programmable virtual networks. A set of general guidelines that apply to programmable virtual network architectures is presented. To illustrate how the guidelines can be met, a resilience differentiation enhancement of the PVN architecture [5] is proposed as a case study.

The rest of this document is organized as follows: Section II discusses general issues related to resilience differentiation in programmable virtual networks and presents a set of general guidelines. Section III presents a case study of how resilience differentiation can be incorporated in the Programmable Virtual Network architecture [5] and Section IV shows how the extended architecture meets both the general guidelines and the specific goals of the original architecture. Conclusions are given in Section V.

## II. Resilience Issues in Programmable Virtual Networks

A programmable virtual network realizes a number of overlay networks over one physical infrastructure. Each overlay network is owned and managed by a VNC, while the physical infrastructure is owned and managed by a network provider (NP). This separation introduces new aspects on end-to-end resilience provisioning. The goal is to satisfy end-users' needs for end-to-end resilience in an efficient way. What is considered efficient varies, but in general it can be measured

 

TABLE I

RESILIENCE PROVISIONING MODELS IN VIRTUAL NETWORKS

| Recovery implementation | End-user signaling | Scalability | Flexibility | Overhead |
|---|---|---|---|---|
| NP | NP | Low | Low | Low |
| NP | VNC | High | High | Low |
| VNC | VNC | High | Medium | High |

TABLE II

EXAMPLE SET OF RESILIENCE CLASSES

| Resilience Class | RC1 | RC2 | RC3 | RC4 |
|---|---|---|---|---|
| Resilience req. | High | Medium | Low | None |
| Recovery time | $< 100ms$ | $< 0.5s$ | $< 5s$ | N/A |
| QoS | Equivalent | Temp. reduced | Reduced | None |

in terms of scalability, flexibility and overhead combined with a target environment. A programmable virtual network is a two layered network managed by different administrative entities. Therefore, the end-to-end resilience can be provisioned in a number of different ways. Resilience provisioning consists of two parts: end-user signaling to establish resilient end-to-end paths and provisioning of recovery mechanisms.

Whether the VNC or the NP should provision one, both or none of these parts is an open issue that depends on the target environment. Table I shows the possible provisioning models and how they relate to scalability, flexibility and overhead. As shown, NP provisioning of recovery mechanisms is efficient in terms of overhead since the same recovery mechanisms will be shared by all VNCs. In the NP/NP model, a full mesh of virtual links have to be established between a VNC's virtual nodes. Thus the number of virtual links is bounded by $O(n^2)$, where n is the number of virtual nodes, resulting in low scalability. The other two models scale better, the number of virtual links is bounded by $O(n)$ as an effect of the customized routing. The NP/VNC model offers highest flexibility. In this model, the VNCs can choose to implement their own recovery mechanisms or buy them from the NP. If a VNC choose to provision its own recovery mechanisms, it is essential that the virtual links used during normal and recovery operation are physically disjoint. Otherwise it might not be possible to recover from a single link or node failure.

When either the NP or a VNC provision both recovery mechanisms and end-user signaling, there is no practical difference compared to a non-overlay infrastructure. Methods described in the literature [6], [7], [8] can be applied directly. However, when end-to-end resilience provisioning is split between the NP and its VNCs, the situation becomes more complicated. The interface between the NP and the VNCs must be very clear and fulfill a number of requirements, as discussed below.

### A. Split end-to-end resilience provisioning

Here, the NP implements a set of recovery mechanisms but no end-user signaling; that responsibility is effectively pushed to the VNCs. However, the NP must provide resilience hooks to the VNCs. The hooks are used by the VNC to specify the desired resilience level when mapping end-user traffic onto its virtual links. Each VNC must be able to select an appropriate set of resilience hooks on a link by link basis since they may have different resilience requirements for different links.

Generally, finer resilience granularity implies higher realization costs in the network. There is a clear trade-off between resilience granularity and realization cost. The need for fine granularity indicates that the actual implementation of recovery schemes should be located in the network layer [2], [7], that is in IP or MPLS. Since provisioning of resilience differentiation requires some grade of traffic engineering, the underlying network should be based on MPLS or a protocol with similar traffic engineering properties.

The authors of [6] propose a definition of resilience levels in terms of resilience classes (RCs). A RC specifies a maximum service disruption time for a set of expected failures such as all single link and node failures. A RC may also specify the allowable reduction of QoS during recovery operation. A network provider offers a limited set of well defined resilience classes to its VNCs, for example the set of resilience classes shown in Table II. In the table, RC1 offers the highest resilience guarantees with a maximum service disruption time of 100 milliseconds and no QoS reduction. RC2 guarantees to recover within 0.5 seconds with possible temporary QoS reduction. Resilience class RC3 offers recovery within five seconds and essentially permanent QoS reduction (until normal operation is resumed). Finally, class RC4 has no recovery guarantees nor does it promise to provide any QoS constraints. These RCs can be extended and further subdivided if so desired. The authors also present resilience signaling extensions to the two dominating QoS models proposed by the Internet Engineering Task Force (IETF): Differentiated Services (DiffServ) and Integrated Services (IntServ) [9]. How many resilience classes and what resilience levels to choose depend on the network provider's situation and expected VNC demands. A programmable virtual network architecture should not restrict the number of supported resilience classes.

When a network failure such as a link or node failure occurs it may result in redirection of a number of flows. How the redirection is carried out and how long it takes depends on the recovery mechanism that protects the affected flows. In ordinary networks, this would not impose any problem. In the case of programmable virtual networks however, the redirected flows are owned and controlled by VNCs, not the network provider. It is of vital importance that the redirection is handled transparently to the VNCs. This is a direct implication of the resilience guarantees offered to the VNCs. The guarantees imply that the NP takes full responsibility to maintain operation in case of a link or node failure. Hence, the redirection must be handled in the physical network, transparent to the VNCs.

Another very important aspect of resilience differentiation in programmable virtual networks is security. The integrity and

privacy of PVNs must be maintained during network recovery as well as during normal operation. Packets must never leak out of or into a PVN. The consequences of misrouted traffic between PVNs are potentially much worse than in non-overlay infrastructures since layer-three addresses can be reused in different PVNs.

VNCs will have access to a PVN in which the virtual links are provided with resilience hooks. The VNCs can apply any resilience signaling protocol such as the extended DiffServ and IntServ protocols proposed in [6] to provision end-to-end resilience. A VNC may choose to implement its own recovery schemes even though it could buy resilience classified hooks from the network provider. In such cases VNC should be signaled when a virtual link goes down or perceives degraded QoS.

A summary of the architectural guidelines for resilience differentiated programmable virtual networks are presented below. The architecture:

- must be based on a network infrastructure that offers traffic engineering capabilities.
- must support a limited but sufficiently large set of resilience classes.
- must provide VNCs with hooks to a suitable subset of the supported resilience classes.
- must provide VNC-transparent traffic redirection during recovery operation.
- must uphold PVN integrity and privacy.
- should notify the VNCs when a virtual link failure or QoS de-gradation occurs.

## III. Resilience differentiation in the PVN architecture

### A. PVN Architecture

The Programmable Virtual Network architecture [5] is an MPLS-based approach that introduces programmability in a carrier grade environment. It basically provides VNCs with a full set of virtual MPLS network resources including labels, links and nodes. The main goals of the Programmable Virtual Network architecture are:

- **Scalability** achieved through a fast data transit path and efficient active packet identification and extraction.
- **Flexibility** to support a wide range of applications, including those with very high processing requirements, without compromising scalability.
- **Migration path** through gradual provisioning of processing capability and expansion of active network applications over legacy infrastructure.

Fig. 1 shows the Programmable Virtual Network node architecture with two Active Processors. The Active Processors virtualise all resources exposed to the VNCs and enforce policies in the MPLS abstraction layer. The Executional Environments (EEs) represent the virtual nodes. An EE has access rights to a set of virtual links and a contiguous block of virtual labels. The label switched router (LSR) is an MPLS switch that conforms to the MPLS standard. The Active Processors are connected to the LSR via a set of label switched paths (LSPs) and a management port. The LSPs carry packets that either require some active processing in the node or originate from EEs in the node's Active Processors. There are two distinct types of flows; the first type carries active packets destined to an EE inside the node, the second type carries transit traffic.

The Programmable Virtual Network architecture uses a two-level label stack. The outer (top) label denotes an MPLS tunnel between two nodes while the inner (bottom) label represents a VNC controlled LSP. The inner label is used by the network provider's control plane to identify both a VNC and the VNC's label. Hence, it is possible to multiplex LSPs controlled by different VNCs onto the same MPLS tunnel. Furthermore, the inner label is used by the forwarding plane to either divert active packets to the corresponding EE or to an outgoing MPLS tunnel. MPLS tunnels start and terminate in LSRs while VNC controlled LSPs start and terminate in EEs.

### B. MPLS-based recovery

MPLS has attracted considerable attention because of its traffic engineering properties [10], [11] and its ability to achieve fast and efficient recovery. A framework for MPLS recovery [7] has been published by the IETF MPLS working group. It discusses issues related to fast and reliable fault detection and recovery mechanisms in MPLS networks. It concludes that MPLS-based recovery can detect a failure and recover traffic on a time scale comparable to synchronous Optical Network (SONET), which is about 50 milliseconds [12]. On top of fast recovery, MPLS provides both high flow recovery granularity and QoS granularity.

A number of different recovery schemes have already been developed [13], [14]. Each recovery mechanism has its own properties in terms of recovery time, QoS preservation, resource utilization and overhead. Which recovery scheme to choose for a given resilience class is an open issue, although some guidelines are given in [6]. If more than one resilience class is provided by a network provider, it is likely that more than one recovery scheme will be implemented in the network.

### C. Resilience extensions

To extend the PVN architecture with resilience differentiation, care must be taken so the architecture's original design goals, as outlined in III-A, remain fulfilled. The resilience extension targets the two- layered approach where the network provider supplies resilience hooks to the VNCs who in turn provide the end-user signaling. How the VNCs provision the end-user signaling is beyond the scope of this paper.

The Resilience Module in Fig. 1 represents the major part of the architecture extension. It is inserted as a middleware layer between the management ports of the Active Processors and the LSR. It contains three major functional blocks. First, it provides a set of recovery engines, each implementing one or more recovery mechanisms. Second, it provides a recovery redirection block that offers redirection support to the recovery engines while hiding eventual redirections from the management block. The label information base (LIB) in legacy

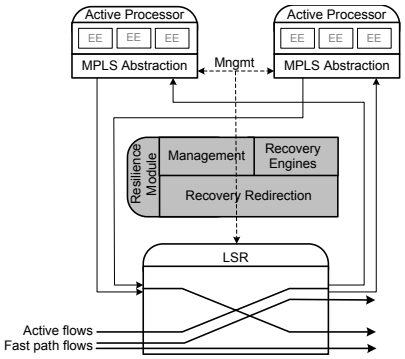


Fig. 1. Extended PVN node architecture with two Active Processors

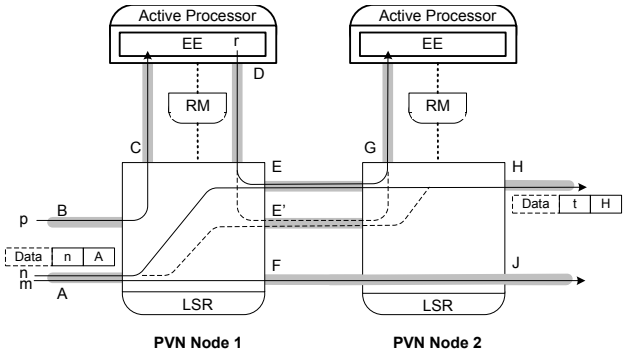


Fig. 2. MPLS Tunnels and VNC controlled LSPs

LSRs is often designed to support fast packet forwarding and hence not very flexible. Therefore, the resilience redirection layer implements an extended variant of the LIB called ELIB. The ELIB provides the additional functionality required in the control plane. Note that the forwarding plane still uses the LIB. Finally, the Resilience Module provides a management block. This block process all configuration requests initiated by the EEs.

A VNC buys virtual links and interfaces from the network provider. A virtual link connects two EEs, or virtual nodes, with each other and a virtual interface is used to transmit and receive data over a virtual link. Instead of purchasing one virtual interface per virtual link, as in the original architecture, a set of virtual interfaces is purchased. Each virtual interface corresponds to a specific resilience class.

The abstraction layer in the Active Processor maps the virtual interfaces to a tunnel protected by a suitable recovery engine, which typically provides the same resilience class as the virtual interface. However, it can be mapped to a an engine that provides higher resilience.

Fig. 2 shows a simple subset of a Programmable Virtual Network with two programmable nodes. The set of gray-colored MPLS tunnels, marked with capital letters, are configured in the nodes by the network provider. Another set of LSPs, marked with small letters, are configured by the VNC residing in the displayed EEs. LSPs *C*, *D* and *G* can be considered as tunnel extensions to reach the appropriate Active Processors but are not conceptually part of the tunnels.

Packets following LSP *p* are regarded as active packets as they are diverted to an EE. Likewise, packets on LSP *r* are considered active since they originate from an EE. Note that packets may be active in one programmable node while inactive in others.

The Resilience Module (RM) in node 1 supervises tunnel *E*. The Resilience Module is configured to switch all traffic in tunnel *E* to tunnel *E'* if a physical link or node failure occurs along tunnel *E*. Observe that tunnels *E* and *E'* can be routed over several intermediate LSRs including other programmable nodes. The switch-over time depends on what recovery engine that is used. Fig. 3 shows a snapshot of the ELIB and LIB configurations on PVN node 1 and PVN node 2 in Fig. 2. When a tunnel failure occurs and the corresponding recovery engine initiates a redirection request, two things take place in the redirection layer. First, the corresponding entry in the ELIB is modified to point to the new desired tunnel *E'* as shown in Fig. 3. Second, all entries in the LSR's LIB that point to tunnel *E* are re-mapped to tunnel *E'*. Whether tunnel *E'* is pre-established or dynamically signaled and allocated when a failure occurs depends on the recovery mechanism implemented by the supervising recovery engine. When a recovery engine detects that a tunnel is down or perceives degraded QoS it signals this to the management block. The management block forwards the signal to all EEs that have a virtual interface mapped to the failing tunnel.

When the Management block in the Resilience Module receives a configuration request from an EE it translates the virtual interface to a tunnel (label) before the request is forwarded to the redirection layer. The redirection layer then checks if there is any active redirection for that tunnel, in which case a substitution is made before the actual LIB configuration is carried out in the LSR. To illustrate this procedure, suppose that tunnel *E* has been re-directed to E' when the setup request of LSP *n* is made in PVN node 1, see Fig. 2 and Fig. 3. First, the management block translates the virtual interface to tunnel *E*. Next, the redirection layer configures its ELIB to swap label n with label q and forward the packets over tunnel *E*. However, tunnel *E* is currently redirected to tunnel *E'*, so *E* is substituted with *E'* before the request is forwarded to the LIB. In PVN node 2, no special precautions are required, thus both the ELIB and the LIB are configured to swap label q with label t and forward over tunnel *H*. Consider the incoming packet in PVN node 1 shown in Fig. 2. The top label *A* is popped and the inner label *n* is swapped with label *q* before the packet is forwarded on tunnel *E'* according to PVN node 1's LIB configuration, as shown in Fig. 3. when the packet arrives in PVN node 2, its top label *E'* is popped and its inner label *q* is swapped with label *t* before it is forwarded on tunnel *H*. The ELIB is never used to perform any forwarding actions in the LSRs.

## IV. Guideline conformance

The guidelines presented in Section II are all met by the proposed resilience differentiation extension:

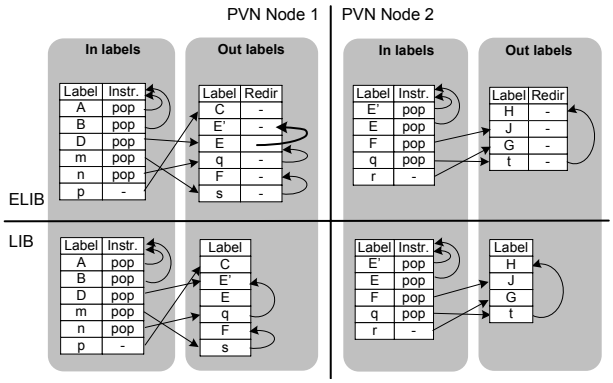


Fig. 3. ELIB and LIB configurations

- The PVN architecture is based on MPLS and therefore offers a sufficient level of traffic engineering.
- The network provider can use multiple recovery engines simultaneously to meet any desired number of supported resilience classes. The extended architecture can even support resilience classes with recovery times as low as 50 milliseconds.
- The virtual interfaces provide the necessary resilience hooks for a VNC to assign end-user flows to different resilience classes.
- The Recovery redirection block provides the required VNC transparency during recovery operation.
- The VNC identification is based on the inner label, hence it is independent of the outer label (tunnel). This guarantees that the packets belonging to different VNCs do not leak either into or out of the PVNs.
- When a recovery engine detects tunnel failure or QoS degradation it signals this to the Management block in the Resilience Module, which in turn makes sure that the affected EEs are notified.

The proposed extension also meets the Programmable Virtual Network architecture's main goals as outlined in Section III-A. The extension leaves both the fast transit path and active packet identification and extraction principles unaltered. It does, however, introduce a larger set of MPLS tunnels: one extra tunnel for each peering PVN nodes and supported resilience classes. A consequence is that the size of the LIB increases and the maximum number of possible independent MPLS tunnels decreases. This fact should be considered when determining the resilience granularity. The ELIB is only used in the control plane, hence it does not affect the forwarding performance in the LSRs.

The proposed resilience extension has no negative effects on flexibility. In fact, the extension enhances flexibility. An even wider range of services and applications can be supported, because the Programmable Virtual Network architecture now provides resilience differentiation to the VNCs. Finally, the migration path is not affected by the resilience extension. However, recovery mechanisms that require forwarding functionality beyond the MPLS standard might not be supported. A legacy LSR may provide the required extra forwarding functionality but it is not required to provide anything more than the MPLS standard specifies.

## V. Conclusion

New services and applications create an increasingly large diversion of network resilience requirements. Programmable virtual networks aim to improve fast and easy provisioning of new services and applications but have, so far, left resilience-related issues unexplored. This paper discusses those issues and provides a set of guidelines that a resilience differentiated programmable virtual network architecture should conform to. An extension to the Programmable Virtual Network architecture was proposed as a case study to show how the guidelines could be met while upholding the architecture's original design goals. In future work, we hope to implement the proposed architecture extension along with a set of recovery engines in the CRC Smart Internet testbed.

## Acknowledgement

The support of the Co-operative Research Centre for Smart Internet Technology (http://www.smartinternet.com.au) for this work is hereby acknowledged.

## References

[1] D. Awduche, A. Chiu, A. Elwalid, I. W. and X. Xiao, "Overview and principles of internet traffic engineering," May 2002, IETF RFC 3272.
[2] P. Demeester, M. Gryseels, A. Autenrieth, C. Brianza, L. Castagna, G. Signorelli, R. Clemente, M. Ravera, A. Jajszczyk, D. Janukowicz, K. Doorselaere, and Y. Harada, "Resilience in multilayer networks," *IEEE Communications Magazine*, vol. 37, no. 8, pp. 70–76, August 1999.
[3] S. Rooney, J. E. van der Merwe, S. A. Crosby, and I. M. Leslie, "The tempest, a framework for safe, resource assured, programmable networks," *IEEE Commuications magazine*, vol. 36, pp. 42–53, October 1998.
[4] T. Brunner and R. Stadler, "Service management in multiparty active networks," *IEEE Commuications magazine*, vol. 38, no. 3, pp. 144–151, 2000.
[5] T. Nguyen, P. Boustead, and F. Safaei, "An architecture for carrier grade programmable networks," in *Proceeding of GLOBECOM*, 2002.
[6] A. Autenrieth and A. Kirstadter, "Engineering end-to-end IP resilience using resilience-differentiated QoS," *IEEE Communications Magazine*, pp. 50–57, January 2002.
[7] V. Sharma, F. Hellstrand, B. Mack-Crane, S. Makam, K. Owens, C. Huang, J. Weil, B. Cain, L. Andersson, B. Jamoussi, A. Chiu, and S. Civanlar, "Framework for MPLS-based recovery," October 2002, work in Progress, Internet Draft, draft-ietf-mpls-recovery-frmwrk-08.txt.
[8] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proceedings of the 18th ACM Symposium on Operating Systems Principles SOSP*, October 2001, pp. 131–145.
[9] X. Xiao and L. Ni, "Internet QoS: The big picture," *IEEE Network Magazine*, vol. 13, no. 2, pp. 8–18, March/April 1999.
[10] D. Awduche, "MPLS and traffic engineering in IP networks," *IEEE Communications Magazine*, vol. 37, no. 12, pp. 42–47, December 1999.
[11] X. Xiao, A. Hannan, B. Bailey, and L. Ni, "Traffic engineering with MPLS in the Internet," *IEEE Network Magazine*, pp. 28–33, March 2000.
[12] "MPLS resilient and scalable interoperability event," *MPLS world congress 2003*, February 2003, white paper.
[13] C. Huang, V. Sharma, K. Owens, and V. Makam, "Building reliable MPLS networks using a path protection mechanism," *IEEE Communications Magazine*, vol. 40, no. 3, pp. 156–162, March 2002.
[14] M. Kodialam and T. Lakshman, "Dynamic routing of bandwidth guaranteed tunnels with restoration," in *Proceedings of INFOCOM*, vol. 2, 2000, pp. 902–911.