Topology Abstraction Service for IP-VPNs

Ravishankar Ravindran, *Senior Member*, *IEEE*, Changcheng Huang, *Senior Member*, *IEEE*, and Krishnaiya Thulasiraman, *Fellow*, *IEEE*

Abstract—VPN service providers (VSP) and IP-VPN customers have traditionally maintained service demarcation boundaries between their routing and signaling entities. This has resulted in the VPNs viewing the VSP network as an opaque entity and therefore limiting any meaningful interaction between the VSP and the VPNs. The purpose of this research is to address this issue by enabling a VSP to share its core topology information with the VPNs through a novel topology abstraction (TA) service which is both practical and scalable in the context of managed IP-VPNs. TA service provides tunable visibility of state of the VSP's network leading to better VPN performance. A key challenge of the TA service is to generate TA with relevant network resource information for each VPN in an accurate and fair manner. We develop three decentralized schemes for generating TAs with different performance characteristics. These decentralized schemes achieve improved call performance, fair resource sharing for VPNs, and higher network utilization for the VSP. We validate the idea of the VPN TA service and study the performance of the proposed techniques using various simulation scenarios over several topologies.

Index Terms—IP virtual private networks, managed IP-VPN service, topology abstraction

1 Introduction

IP-VPN [1] service is typically backed by strong service layer agreements (SLA), such as QoS guarantees, reliability, and data security requirements. Nagarajan [1], Knight and Lewis [2] discuss the requirements of a L3 PPVPN service from a VPN service provider's (VSP) perspective. Recently, most of the research in this area has focused on traffic engineering of VPN traffic over VSP's network to achieve load balancing and maximize network utilization. These studies assume the availability of a site-to-site traffic matrix. This mode of VPN provisioning is known as the PIPE model [3]. Given the traffic matrix, a VSP can then determine the network paths by solving a multicommodity optimization problem adapted to a VPN context with business constraints and optimization objectives to serve the VPN's and VSP's interests (as in [3]). Estimating site-to-site traffic matrix specification is often a difficult task. To address this problem HOSE model was proposed in [4]. In this mode of provisioning, the VPNs specify traffic demands as aggregate ingress and egress traffic estimates for each VPN site. Given the per site ingress-egress aggregate traffic requirement, [5], [6] are efforts that further enhance the HOSE model by proposing new provisioning schemes that realize aggregate site demands in the core network. More recent work [7] builds over the Hose-based provisioning model based on the aggregate access capacity of the CE to the PE connectivity. The advantage of the proposed technique is that core traffic engineering is oblivious to the eventual traffic matrix distribution of the site-to-site VPN traffic hence allowing significant fluctuations without reengineering the traffic in the VSP's network.

The PIPE and the HOSE modes of VPN provisioning are generally good for IP-VPNs whose traffic specification changes at longer time scales. With increasing deployment of short lived and bandwidth intensive multimedia applications in enterprise VPNs, an approach designed to enable the VPN sites to negotiate SLA and seek services dynamically is required. In this paper, a novel framework that enables dynamic VPN service (DVS) in the context of L3 PPVPN using topology abstraction (TA) is proposed. Earlier research on enabling DVS was limited to the study of link capacity sharing dynamically among VPNs. Initial efforts, such as those in [8], [9], proposed solutions which allow dynamic resizing of the preestablished virtual circuits as a function of varying traffic demands of the VPNs, and thereby improving the statistical multiplexing gain and the overall utilization of the core network. A more recent line of research in the existing literature, which enables DVS, is the work related to programmable VPNs; this work allows VPNs to have access to a subset of the router and network resources that can be optimized in order to satisfy the interests of the customer's applications. Isaacs and Leslie [10] proposed a dynamic programmable VPN architecture that allows the spawning of dynamic VPN networks with dedicated router and link resources built over logical switch partitions called *switchlets* that are controlled by an open control interface. These programmable VPN architectures and solutions enabling dynamic VPN creation assume availability of access to physical router and link resources through an open programmable interface. For this reason, employing this approach is not possible when strict trust issues exist, as in the case of a VPN enterprise and a transport infrastructure owned by

[1] In the paper, the terms IP-VPN and L3 PPVPN have been used interchangeably.

- *R. Ravindran is with the Huawei, 2330 Central Expressway, Santa Clara, CA, 95050. E-mail: ravi.ravindran@huawei.com.*
- *C. Huang is with the Department of Systems and Computer Engineering, Carleton University, 1125 Colonel By Drive, Ottawa, ON K1S 5B6. E-mail: huang@sce.carleton.ca.*
- *K. Thulasiraman is with the School of Computer Science, University of Oklahoma, 200 Felgar Street, Room 114, Norman, OK 73019. E-mail: thulasi@ou.edu.*

Manuscript received 26 Jan. 2011; revised 26 Nov. 2011; accepted 9 Dec. 2011; published online 11 Jan. 2012.
Recommended for acceptance by C.-Z. Xu.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2011-06-0355.
Digital Object Identifier no. 10.1109/TPDS.2012.27.

1045-9219/13/$31.00 © 2013 IEEE Published by the IEEE Computer Society

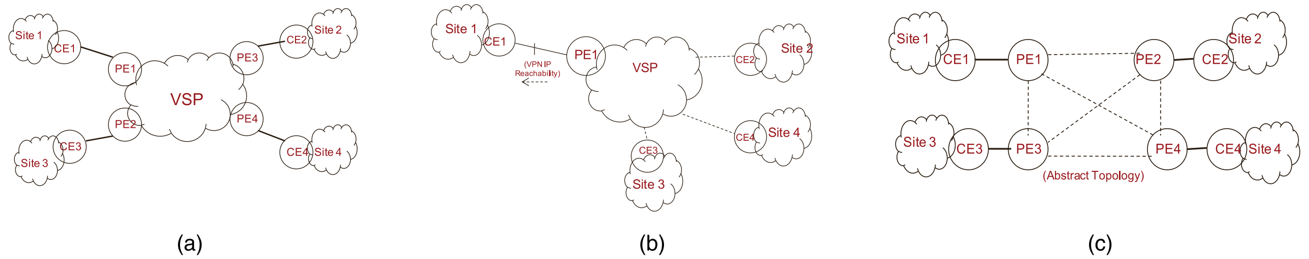


Fig. 1. (a) VSP providing managed IP-VPN service, (b) Current IP-VPN service only enables IP reachability, (c) VSP providing topology abstraction service.

a commercial VSP. Another drawback with this mode of VPN management is similar to that of the PIPE model which requires prior knowledge of the demand matrix of the VPN, which is then mapped to the appropriate switch and link resources in order to spawn VPNs dynamically and enable the customers to manage it.

In this paper, we propose a service that enables VSP to share its core topology and link state information in a scalable manner with its IP-VPN customers. This service is enabled by the novel application of topology abstraction, which is along the lines of the topology aggregation concept applied for hierarchical routing, but is new under the context of IP-VPN service. This TA service is studied in the context of managed dynamic VPN framework which is amenable for realization in a VSP's network. The VPNs use the abstracted topology and QoS metric information to seek services on demand from the VSP. This mode of provisioning prevents wastage of any committed resources, which may occur as a result of poor utilization in a dynamic call request scenario when well-known models of provisioning based on the PIPE or the HOSE model are used; it also enables statistical multiplexing of VPN demands in the core of the VSP's network.

TA service provides VPNs more information than reachability information enabled today and therefore more options in terms of how they route their traffic. This can improve VPN call performance significantly. The challenge is how to create TAs for different VPNs so that the VSP core network is shared efficiently and fairly. We propose three decentralized schemes to address this challenge. There is a tradeoff among these schemes in terms of VPN call performance and VSP network utilization. We will demonstrate that, compared with the traditional VPNs which do not deploy TA service, the VPNs with good TAs show significant call performance improvements. Applying TA could also lead to certain tradeoffs as seen in our simulation results. While TA allows VPNs to have very good call performance, we noted a tradeoff with respect to network utilization. But the reduction in the average utilization is not found to be very significant.

This paper is organized as follows: Section 2 provides a general discussion and sets the objective for the TA service. Section 3 shows how TA as a service can be realized by a current standardized IP-VPN framework. Section 4 introduces graph theory notations used in this paper. Section 5 proposes three abstract topologies and related SLA parameters required to generate service differentiation in the context of the TA service. Section 6 defines the VPN topology abstraction (VPN-TA) problem and proposes three abstraction algorithms that result in TA service with different performance characteristics. Section 7 discusses the simulation results studying the benefits of the TA service and the three abstraction schemes proposed to address the problem of generating fair TA abstractions to VPNs.

2 Topology Abstraction Service for IP-VPNs

A VSP today only has the ability to disseminate reachability information to all the CE nodes of a given VPN using the constructs defined in [1]. This is shown in Fig. 1a, where the VSP provides managed IP-VPN service to a VPN. Fig. 1b shows the opaque nature of the service today, here, the VPN sites whose gateways are represented as CE1-CE4 are logically connected to the other sites through their IP reachability property enabled by the VSP without having any view of the state of the core network. Fig. 1b shows this reachability with respect to Site-1 whose gateway CE-1 receives the reachability updates from border node PE-1, hence allowing virtual connectivity for Site-1 to the remaining sites.

This information suffices the need of most of the IP-VPNs today, as the QoS for the VPN traffic is prenegotiated with the VSP, using the PIPE or HOSE mode of traffic specification and guaranteed by provisioning appropriate core resources to handle the traffic. The assumption here is that, these traffic specifications are expected to only vary in long time scales. Such long time scale variations give a VPN sufficient time to renegotiate the existing SLA to include the future capacity requirements. However, with the increasing number of multimedia applications in enterprises used to support applications such as conferencing, collaboration, and data management involving backups and restoration, the VPN customers, today, require significant capacity for a number of small windows of time in order to meet the needs of these bandwidth intensive applications. In current approaches, if the capacity requirements between the sites exceed existing negotiated resources based on the demands of applications requiring dynamic resources the VPN have to go through the process of renegotiating existing SLAs. Even if the VPN customers were able to predict the worst case capacity requirements and renegotiate the existing SLA, this solution has two unwanted drawbacks. First, the VPN will have to pay for the capacity negotiated even if it does not use it beyond the intended period of use. Second, the VSP who commits the resource for a given VPN could experience poor network utilization and possible loss of revenues. This paper is about how a VSP can make use of this opportunity and abstract the underlying core network information in a fair and efficient manner and provide this to the VPNs so that they can make informed requests based on the information related to the resource availability. An example of this is shown in Fig. 1c. Here, the core has been abstracted as a fully meshed graph. The links of the abstract topology are virtual links that may be associated with abstracted static and/or dynamic QoS metrics.

Applying TA to share the core resource information with the VPNs serves multiple purposes for the VSP: 1) It allows the VSP to share the core resource information in a scalable manner, without having to divulge its sensitive and private core topology information; 2) TA can be used by the VSP as a tool to abstract its core network based on its local policies to any degree of granularity, correctness, and associate it with one or many aggregated QoS metric information it desires. This property of TA allows the VSP to negotiate the properties of the TA based on VPN's requirements and arrive at appropriate SLAs. 3) TA will also significantly reduce the signaling cost for the VSPs, particularly during high load conditions when the signaling will be terminated at the VPN's end itself.

TA information can be used by a VPN in many ways such as: 1) To adapt its current QoS requirements (for, e.g., by modifying the application layer parameters) to what the provider can offer at any given point of time. This allows applications (such as [11]) with dynamic bandwidth requirements to improve their decision based on availability of resources, which is expected to yield good probability of success in terms of both successfully requesting as well as rejecting the demand locally; 2) Human and information resources within a VPN are typically spread among the VPN sites, thus the VPN has to deal with the problem of routing its intersite traffic to meet the enterprise needs. TA information shall allow the VPN to traffic engineer their intersite traffic more efficiently by adapting to the varying state of the core network; 3) In case a VPN is multihomed (i.e., a VPN site connected to multiple VSPs), and in a situation where it may avail the TA service from more than one of them, it allows the VPN to be opportunistic in choosing the provider that can best suit its needs for its given requirements.

Based on the above discussion, we now summarize the objectives of the TA service definition for the IP-VPNs.

- Enable a VSP to share core topology information as a service with appropriate SLA definition to achieve service differentiation.
- One important requirement of this service is to have the ability to realize it as an extension to the existing managed IP-VPN solutions like [12], [13], and thereby, minimize the changes to the control and transport plane logic and the cost of realizing this service in current IP-VPN frameworks.
- The service should enable the VPNs to use the TA information to perform intelligent route computation, and leverage it to seek resources on demand in order to meet its QoS requirements for the duration of the call request.
- In a dynamic call request scenario, the service should also result in good VPN call performance, while maximizing the network utilization.

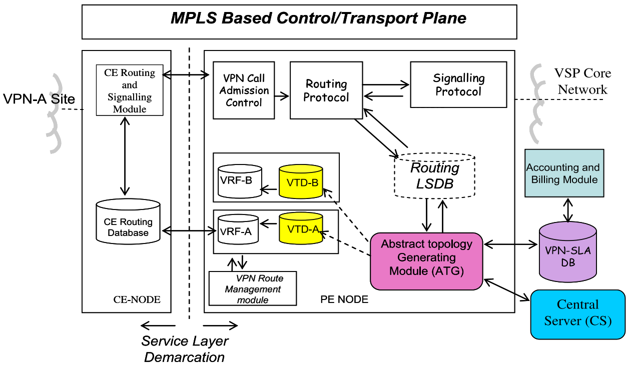

MPLS Based Control/Transport Plane
VPN-A Site
CE Routing and Signalling Module
CE Routing Database
CE-NODE
VPN Call Admission Control
Routing Protocol
Signalling Protocol
VSP Core Network
VRF-B
VTD-B
VRF-A
VTD-A
Routing LSDB
Abstract topology Generating Module (ATG)
VPN Route Management module
PE NODE
Accounting and Billing Module
VPN-SLA DB
Central Server (CS)
Service Layer Demarcation

Fig. 2. Managed dynamic VPN framework.

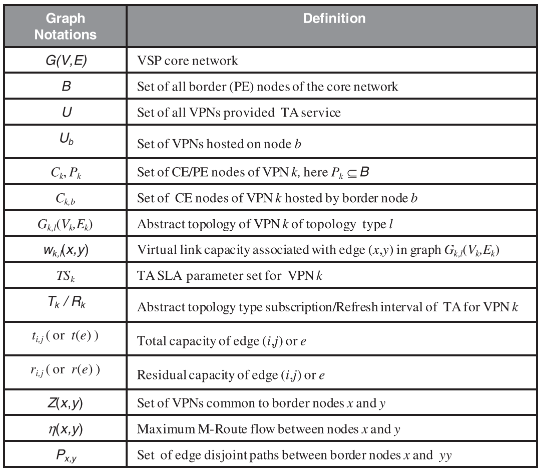
3 Managed Dynamic VPN Framework

In this section, we introduce the managed dynamic framework (MDVF) for implementation in a VSP's network which realizes the previously mentioned functions as part of the TA service to the VPNs. In particular, we propose extensions to a well-known IP-VPN solution [12] in order to realize it. We add new functional components to this solution to realize the MDVF for the VSP enabling the TA service. Fig. 2 shows the components of our framework.

The proposal in [12] uses VPN specific virtual routing and forwarding (VRF) instance tables within PE routers to logically separate VPN contexts, both from a control and transport plane perspective. The additional software modules required to realize the MDVF as part of the control plane logic are shown as colored components in Fig. 2.

The *abstract topology generation module* (ATG), is the most important module in the context of the TA service that executes the core capacity sharing algorithms locally in the PE nodes with or without the assistance of the *central server* (CS). A central server is a centralized module, and its involvement depends on whether the abstractions are generated in a decentralized or in a centralized manner. The ATG module makes decisions on how to share and expose the resources of the core network among the VPNs, while taking into consideration the SLA parameters negotiated between the VPN and the VSP. One key VSP policy is related to fairness; this policy decides the criteria for logically partitioning the core resources from which the TAs are generated. The notion of fairness, as applicable to TA service, will be discussed later. The ATG module in the PE nodes also maintains timer based interrupts during which the abstract topologies are refreshed regularly and updated to the VPN specific *VPN abstract topology database* (VTD). The VTD has a VPN scope, and as a result, it can be considered as an extension of the VRF, which in the current solution [12] is the database of VPN site-to-site routing information or can be implemented as a stand-alone database. The updates to abstract topologies from the above discussed ATG module are stored in this database.

The correctness of the TA information is associated to the *abstract topology refresh interval* metric, a parameter negotiated between the VPN and the VSP. These updates to the VPN CE node can be sent over as part of the LSA updates of a protocol, such as OSPF. Rosen and Psenak [14] propose the use of OSPF as the peering protocol between the CE and PE node for BGP/MPLS based IP-VPN solution [12] to exchange



Graph Notations	Definition
$G(V,E)$	VSP core network
B	Set of all border (PE) nodes of the core network
U	Set of all VPNs provided TA service
U_b	Set of VPNs hosted on node b
C_k, P_k	Set of CE/PE nodes of VPN k, here $P_k \subseteq B$
$C_{k,b}$	Set of CE nodes of VPN k hosted by border node b
$G_{k,l}(V_k,E_k)$	Abstract topology of VPN k of topology type l
$w_{k,l}(x,y)$	Virtual link capacity associated with edge (x,y) in graph $G_{k,l}(V_k,E_k)$
TS_k	TA SLA parameter set for VPN k
T_k / R_k	Abstract topology type subscription/Refresh interval of TA for VPN k
$t_{i,j}$ (or $t(e)$)	Total capacity of edge (i,j) or e
$r_{i,j}$ (or $r(e)$)	Residual capacity of edge (i,j) or e
$Z(x,y)$	Set of VPNs common to border nodes x and y
$\eta(x,y)$	Maximum M-Route flow between nodes x and y
$P_{x,y}$	Set of edge disjoint paths between border nodes x and yy

Fig. 3. Graph theory notations.

reachability information. This recommendation to use OSPF can also be extended to flood the abstract topology information as link state updates as part of TA service to the VPNs. The abstract topology information, once populated in the client routers, can be used by the VPNs to compute end-to-end paths traversing the VSP's core network and check on the availability of the desired QoS.

The *VPN SLA database* (VPN-SLADB) is a database for the SLA parameters of all the VPNs subscribing to the TA service. This may be implemented as a centralized database that will be accessible to all the PE nodes or as a distributed database local to a PE node managed by the operator.

4 Graph Theory Notations

The graph theory notations used in this paper are summarized in Figs. 3 and 4 depict a VSP providing TA service to two VPNs.

With reference to the notations in Fig. 3 and example network in Fig. 4 where a VSP is providing IP-VPN service to two VPNs (green and red). VSP's core topology is represented as a graph $G(V,E)$. Each directed link $e = (i,j) \in E$ of the network has a capacity represented as $t_{i,j}$ (which we also denote as $t(e)$). This is the total capacity of the link. On the other hand, the capacity available on a link e at any given time is called residual capacity, denoted as $r_{i,j}$ (or $r(e)$). B represents the set of all border PE nodes in graph $G(V,E)$. U represents the set of all VPN customers subscribing to the TA service. Each border node $b \in B$ may support multiple VPN instances identified as U_b. For a VPN instance $k \in U$, we represent the sets of corresponding CE and PE nodes as C_k and P_k, respectively. The set of CE nodes corresponding to a VPN instance $k \in U$ hosted on a border node b is represented as $C_{k,b}$. The "P" nodes in Fig. 4 are the provider nodes, which only participate in routing or switching the traffic.

As part of the TA service, each VPN is served with an abstract topology of type l. We discuss different types of abstract topologies in Section 5.1. For a given VPN $k \in U$, we represent the abstract graph as $G_{k,l}(V_k,E_k)$. V_k includes the subset of PE nodes P_k, the set of CE nodes C_k, and the set of virtual nodes Y_k whose meaning vary with the type of TA. Each of the nodes in the set P_k and C_k map to a border PE and a CE node, whereas Y_k are logical node(s) virtualizing VSP's core which does not map explicitly to any of the core or border nodes. There are two types of links comprising the set of edges E_k in the abstract topology. The first is the access link, which connects a C_k node to a P_k node, and the other is the set of virtual links connecting a P_k node to a virtual node in Y_k or another P_k node. Broadly speaking, the virtual link $e \in E_k$ can be associated with a vector of abstracted QoS metrics. *Here, we restrict ourselves with one abstracted metric, namely, bandwidth.* Hence, for a VPN k provided, with an abstract topology of type l, we denote the abstracted capacity associated with the virtual link connecting nodes x and y as $w_{k,l}(x,y)$. This bandwidth represents the capacity exposed by the VSP between the pair of nodes $(x,y) \in V_k$. The remote access link connectivity information flooded as part of the abstraction will carry the available capacity of the physical link corresponding to that remote access link. A VPN k is also associated with a set of SLA parameters, and this is represented as TS_k.

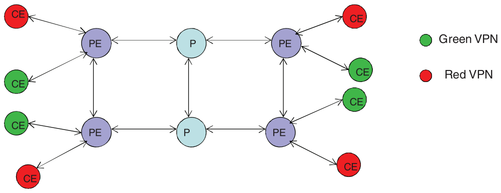


Fig. 4. VSP providing TA service to two VPNs.

5 TA Service SLA Definition and Parameters

To enable a VSP to use TA service for generating service differentiation among the VPN customers, we propose a new set of topology abstraction SLA (TA-SLA) parameters, which allows the VSP to customize the properties of the TA service to the requirements of the VPN. We next discuss the elements of the TA-SLA.

Abstraction topology type parameter. This parameter represents the type of abstract topology generated by the VSP for a VPN. The VSP uses this parameter to generate an abstract graph with a certain granularity before sharing it with the VPN. The optimizing objective for any form of TA is to minimize the complexity with respect to the granularity of the abstraction, while at the same time, maximizing the accuracy of the topology metric information that is being abstracted. In this paper, we have considered three forms of abstract topologies: source-star abstraction (SSA), star abstraction (SA), and simple node abstraction (SNA), which are also the most well-studied forms of abstractions in the context of hierarchical routing literature. We discuss the properties of these TAs in the context of a VPN service later on in the section. We use the notation T_k to denote the TA type subscribed by VPN k. In this case, the parameter would be one of SSA, SA, or SNA types. $G_{k,l}(V_k,E_k)$ will denote the abstract topology l corresponding to VPN k, where the l represents one of the abstract topologies SSA, SA, or SNA.

Abstract topology link metric. This parameter identifies the choice of QoS metric information that is required to be associated with the virtual links of the abstract topology chosen as part of the abstraction topology type parameter. It

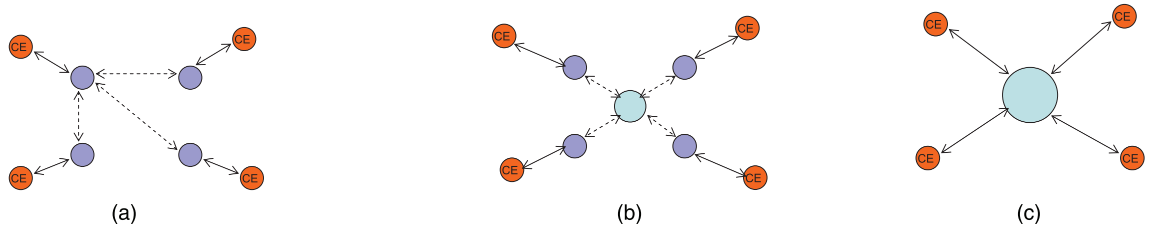


Fig. 5. (a) source-star abstraction topology, (b) Star abstraction topology, (c) Simple node abstraction topology.

is not necessary that all the VPNs subscribe to the same set of QoS metric information. The choice of QoS metrics associated with the abstract topology could be determined by the purpose for which the TA service will be used for. The problem with abstracting link metric information to IP-VPNs can be classified into two categories: single metric abstraction and multiple metrics abstraction. Here, we deal only with the abstraction of the available capacity associated with the links of the core VSP network. The virtual link metric associated with the nodes x and y for a VPN k of the abstracted graph $G_{k,l}(V_k, E_k)$ is represented as $w_{k,l}(x, y)$.

Abstract topology refresh interval metric. The importance of this parameter is linked to the fact that the VSP provides periodic updates about resource availability by flooding the TA periodically in order to synchronize the core state information with that of the VPN's view of the core network. From this, we can see intuitively that the choice of this parameter decides the correctness of the TA information at any given point of time in relation to the link state of the VSP's core network. In a dynamic call request scenario, the ideal refresh rate (inverse of the refresh interval) desired by a VPN is expected to be a function of the mean arrival rate of bandwidth requests, dynamic nature of the state information in the core network, which will depend on factors like the load offered by all the VPNs and time of day, and the tolerance for the control overhead by the CE routers due to the frequency of update of abstraction information from the PE nodes. The abstract topology refresh rate negotiated by the VPN is also influenced by both the abstraction topology type and the abstract topology link metric chosen by the VPN. The abstract topology refresh interval metric for a VPN k is represented as R_k.

5.1 Topology Abstraction Types

A number of topology abstraction types can be generated from a given VSP core network. Not all types of topology are created equal. The criteria for selecting a special type of abstract topology depend on the objectives of TA. The goal of a service provider is to provide service differentiation to VPNs while hiding the details of the core network. Clearly, simple abstract topologies that can provide service differentiation are ideal candidates. In this paper, we will consider three abstract topologies with increasing complexity and available information: simple node abstraction, star abstraction, and source-star abstraction. Figs. 5a, 5b, and 5c show the abstractions that are being considered. We next elaborate on the different TA types, referring to Fig. 4, which shows a VSP providing TA service to two VPN clients. The two headed arrow in the figure represents a pair of oppositely directed links. The straight arrow in the figures represents a physical link, while the dashed one represents a virtual link.

Source-star abstraction topology. This type of abstraction, illustrated in Fig. 5a, has been adapted from [15]. In this case, a VPN is provided an abstraction that is a source-rooted tree. Here, the root of the tree is the PE node computing the abstraction for the corresponding VPN. The other nodes of the source-star topology are the PE nodes P_k on which VPN of the same type are hosted. In the case where a CE node has adjacency with more than one PE node, multiple source-star abstractions would have to be provided to the CE node in order to take advantage of the routing diversity due to multiple PE connectivity. For the case of single CE-PE connectivity and for a given VPN k, the total message complexity of updating a CE is $O(|P_k| + |C_k|)$.

Star abstraction topology. This abstraction is shown in Fig. 5b. In this scheme, the abstract topology contains all the border nodes hosting the VPN, i.e., the set of nodes P_k, and virtualizes the remaining network as a virtual node v. The virtual link edges of the abstract topology correspond to the connectivity between the border nodes and the virtual node. For a given instance of VPN k, the spoke of the star topology connects a node $u \in P_k$ to the virtual node v bidirectionally. The bandwidth of a virtual link can be assigned in multiple ways; one way is to assign to each virtual link connecting a node $u \in P_k$ to the spoke node v the bandwidth that is the average of the bandwidths of all the virtual links (u, w) in the source-star abstraction topology with $w \in P_k$ and $w \neq u$; the equations to compute this virtual link capacity are given in (1) and (2). In this equation, $w_{k,SA}(u, v)$ represents the capacity associated with the directional virtual link $(u, v) \in E_k$ corresponding to abstract graph $G_{k,SA}(V_k, E_k)$. Summations are over all nodes in P_k. Another approach, but an aggressive one, to compute this virtual link capacity is to set this to the maximum of the capacities of virtual links (u, w) in the source-star abstraction topology with $w \in P_k$ and $w \neq u$. For any given VPN k, the total message complexity to flood a star abstraction topology is $O(|P_k| + |C_k|)$

$$w_{k,SA}(u, v) = \sum_{w} \frac{(w_{k,SSA}(u, w))}{(|P_k| - 1)} \quad \forall w \in P_k, w \neq u \qquad (1)$$

$$w_{k,SA}(v, w) = \sum_{l} \frac{(w_{k,SSA}(l, w))}{(|P_k| - 1)} \quad \forall l \in P_k, l \neq w. \qquad (2)$$

Simple node abstraction topology (SN). This is the least granular of all the abstraction topologies. Here, the core network is completely virtualized as a single node, as shown in Fig. 5c. The VPN CE nodes C_k in this case are provided only the updates of the available bandwidths on the remote VPN access links while the set of nodes P_k are virtualized as a single node. The complexity of updating a VPN with this form of abstraction is $O(|C_k|)$.

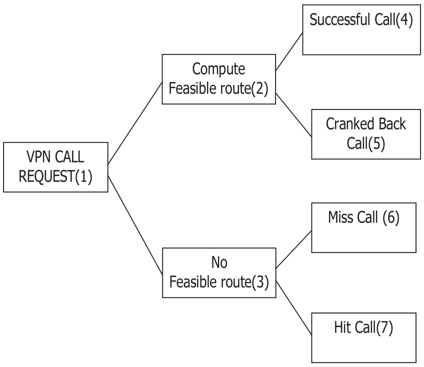


Fig. 6. VPN call request scenarios.

As discussed above, among the three types of topology, the SSA type provides the finest granularity. Once a SSA type topology is generated, the other two topologies can be generated from the SSA topology through information aggregation which results in information loss and differentiated services. Our later discussions will therefore focus on generating SSA type TAs with fair resource sharing among VPNs.

5.2 Topology Abstraction Types

One of the significant benefits of abstracting VSP's core network and QoS information is to improve the call performance of the VPNs that require dynamic bandwidth service, while minimizing the overhead for sharing such information with the VPNs. Three call performance metrics have been defined to study the efficiency of the TA service, which are success, crankback, and misscall ratios. These metrics are explained with respect to Fig. 6 which shows the different possibilities of a bandwidth request from a VPN.

Starting from the root, denoted as (1), a VPN call request could have two possible outcomes. The first outcome is when a CE node finds that the requested capacity is not available and terminates the call locally. This is identified as (3). The second outcome is that the CE node finds that the requested capacity is available and sends the request to the corresponding PE node (2). For the case (3), we could have two possibilities. The first possibility is that the call has been wrongly terminated due to stale abstraction information, in which case we count it as a "Miss" call (6). The alternative possibility is that the call has been correctly terminated because of the VSP's inability to find the required path in the core network. We label such a call as a "Hit" call (7). For the case where a path is deemed to exist by the VPN, there can be two outcomes. The first is that the call does go through to the destination and the CE node receives a positive acknowledgement from the PE (4). The other is that the call crankbacks because of the inability of the PE node in the VSP to route the call (5). Crankbacked calls also waste signaling processing resources for the VSP. Success ratio measures the percentage of correct decisions in terms of both successfully computing a path (with successful signaling) and rejecting the call locally using the abstraction provided by the VSP. Crankback ratio measures the percentage of calls that resulted in successful path computation locally but is cranked back by the VSP. Misscall ratio measures the proportion of calls which were terminated wrongly at the VPN's end, in spite of sufficient resources in the core network to satisfy the call. For each VPN, success ratio is to be maximized while crankback and miscall ratio are to be minimized.

To generate the TA for the VPNs, we assume the availability of the entire core network topology, locally in a border node. In addition, the algorithms also require VPN specific information, which would include information about the VPNs hosted on each border node, along with the residual capacity of the VPN access link connecting the PE and the CE nodes. The core topology information is readily available as part of link state protocols, such as OSPF or ISIS, which are the protocols of choice in most VSP networks. The VPN specific information required by the abstraction schemes can be obtained from a TA service specific database populated by appropriate protocol extensions that include VPN membership information and the associated TA-SLA parameters discussed earlier. This information can be either locally available at the PE node or accessed centrally from a centralized database.

An important factor to consider during the abstraction process is the TA-SLA parameters, particularly the abstraction topology type parameter to offer service differentiation. To enable this service differentiation, we assume that the VSP adopts a model where the goal is to always expose the available capacity differently to different types of TA while exposing the available capacity equally among all the VPNs that have subscribed to the same abstract topology type parameter T_k .This fairness policy is applied locally by each border node b, while generating abstract topologies for the set of VPNs U_b hosted by it. The criterion we adopt can be stated as follows: For a pair of VPNs $(i, j) \in U_b$ with the same abstract topology type parameter l, with abstract graphs $G_{i,l}(V_i, E_i)$ and $G_{j,l}(V_j, E_j)$ and with a common pair of nodes x and y, the VSP imposes the rule that the exposed capacity associated with virtual link connecting the nodes x and y should be equal, that is $w_{i,l}(x, y) = w_{j,l}(x, y)$. This ensures that two VPNs with the same TA type will be exposed the same amount of bandwidth for each virtual link connecting the nodes x and y in the abstract topology.

This paper explores decentralized mode of TA generation, here each PE node generates TA for each of the hosted VPN using the latest topology and link state information and the fairness criterion as discussed above.

6 Topology Abstraction Problem and Abstraction Algorithm

The challenge faced during the abstract topology generation process is the fact that the process of generating TAs may lead to oversubscription of the available core resource, which could result in poor VPN call performance and high signaling cost for VSPs under high load conditions. Oversubscription results because of two reasons: first due to the decentralized approach of TA generation wherein each border node generates TAs for the VPNs without coordinating with one another; second, the degree of oversubscription also depends on the abstraction algorithm that is used to generate the abstraction for each VPN by a border node. The VPN-TA problem stated next is to address the problem of oversubscription arising due to the second factor in a decentralized context of TA generation. The objectives of this problem are the following:

- Provide the VPNs with an accurate representation of available capacity considering the case of having to satisfy simultaneous VPN calls during high load condition. This objective also correlates with the objective of maximizing the call performance of the VPNs.
- Maximize utilization of the VSP core network.
- Generate fair abstraction which aligns with the fairness policy presented in Section 5.2.

VPN topology abstraction (VPN-TA) problem. *Given a set of VPNs U_b hosted on the border node b, each VPN instance $i \in U_b$ is to be provided with an abstract topology $G_{i,l}(V_i, E_i)$. The objective is to device a methodology to allocate virtual capacities to the links in E_i so that the VSP maximizes the probability of each VPN making a correct decision of successfully computing or rejecting a path locally in the context of the TA service.*

We propose three algorithms for the above problem. All the three abstraction schemes have been proposed to maximize the call success ratio. The three schemes vary in their nature from being aggressive or conservative in terms of associating virtual capacity to the link in the abstract topology. The nature of these algorithms leads to different performance results varying in tradeoffs between the three call performance metrics, i.e., the success, crankback, and misscall ratio which we introduced in Section 5.2.

As the VPN-TA problem is defined in a decentralized context, the algorithms presented in the following sections are in the context of a PE node b which executes these algorithms to generate the required abstractions. Also, to make the presentations easy to follow, all discussions and algorithms are with respect to source-star abstraction since the star and the simple-node abstractions can be derived from the source-star abstraction as discussed in Section 5.1.

6.1 Maximum Capacity Abstraction Algorithm

The maximum capacity algorithm is aggressive because the maximum network resource is advertised to all clients making each client think it can use all the capacity. In this scheme, the VSP sets the virtual link capacity $w_{k,SSA}(b_1, b_2) \in$ corresponding to a pair of border nodes (b_1, b_2) to the capacity of the widest path (defined next) from b_1 to b_2. For a given pair of border nodes $(b_1, b_2) \in P_k$ hosting a VPN k, let $P = \{p_1, p_2, \dots, p_l\}$ be the set of paths from b_1 to b_2 Let $C(p_i)$ be the bottleneck capacity of each path. We define as $\omega(b_1, b_2) = max_{i=1\dots l}(C(p_i))$ the maximum capacity of a path from b_1 to b_2. This path is called the widest path from b_1 to b_2 . Once the capacity of this path is computed, we set the virtual link capacity $w_{k,SSA}(b_1, b_2)$ of VPN k to $\omega(b_1, b_2)$.

We can obtain the widest path capacity by computing a maximum capacity tree rooted at border node b. We refer to this tree as $T(V_k, E_k)$ for VPN k rooted at b. This tree spans all the nodes of P_k. This tree can be obtained in $O(|V|^2)$ time by applying a modified form of the Dijkstra's shortest path algorithm [16]. Wang and Crowcroft [16] give an algorithm to compute a shortest widest path in a graph, in our case the edge cost metric is 0. A tree thus computed enables one to generate the set of virtual links from node b to other nodes in P_k. This step is iterated over all the nodes in P_k to generate the trees rooted at the respective nodes. The worst case complexity for a border node to compute all these rooted trees is $O(|U| * |B| * |V|^2)$ assuming that every border node hosts all VPNs in U.

```
Maximum Capacity Path Algorithm
Input: $G(V,E)$, $U_b$ .
Output: $G_{k,SSA}(V_k,E_k)$, $\forall k \in U_b$ .
begin
Step1: identify the set of PE nodes $P_k$ belonging to VPN $k$ ;
Step 2: For each $x \in P_k$ ,
        compute the widest path tree $T(V_k, E_k)$ rooted at node $x$;
      For each $(x,y) \in E_k$ ,
        set $w_{k,SSA}(x,y)$ = the capacity of the path between nodes $x$ and $y$ in $S(V_k, E_k)$ tree rooted at $x$;
end;
```

Fig. 7. Maximum capacity abstraction scheme for VPN topology abstraction problem.

As noted earlier, this approach is very aggressive since the maximum capacity between a pair of border nodes is advertised to all the VPNs having that pair of border nodes in common. This aggressive mode of capacity sharing may work well during low load conditions, when there is a high probability that the capacities associated with the virtual links are in sync with the state of the core network. However, this approach may result in poor call performance at high load conditions when the abstraction at the VPN's end may not be synchronized with the state of the core network.

Note that, for this scheme (Fig. 7), we don't apply the fairness policy as all the subgraphs $S(V_k, E_k)$ are computed using the same core topology, which results in TA for VPNs that satisfies the fairness policy defined in Section 5.2.

6.2 Mixed Bound Abstraction Algorithm

The maximum capacity abstraction algorithm may not perform well in high load conditions. This is particularly true when the aggregate VPN demand request between a pair of border nodes (during the interval between two abstract topology refreshes) from multiple VPNs exceeds the capacity of the maximum capacity paths between the pair of border nodes. To address this, we propose an algorithm where, instead of a single value, we represent virtual capacity as a pair of values in the form of upper and lower bounds. Providing such bounds is expected to give more realistic information on the available capacity in the VSP's network and help in handling the issue of simultaneous bandwidth requests received by a border node during high load conditions. This approach uses flow-based algorithms in order to compute the bounds.

Upper bound on advertised capacity. The goal of providing an upper bound on advertised capacity to a VPN is to ensure that the total amount of resources it seeks does not exceed the advertised upper bound during the interval between two consecutive abstract topology refreshes. For a VPN k and virtual link corresponding to edge $(x, y) \in E_k$, let $M_{k,i}(x, y)$ be the ith bandwidth request between two consecutive refresh update interval instances, and let $\delta_{k,SSA}(x, y)$ be the upper bound on advertised capacity. If there are n requests from a VPN k in the interval R_k then

$$\sum_{i=1}^{n} M_{k,i}(x, y) \leq \delta_{k,SSA}(x, y). \tag{3}$$

For a VPN $k \in U_b$ and $x, y \in P_k$, $\delta_{k,SSA}(x,y)$ is the computed upper bound capacity advertised for the virtual link connecting nodes x and y. Let $Z(x,y)$ be the set of all VPNs that share the border nodes x and y. According to the capacity sharing policy, equal sharing implies that the total flow advertised must be at most $|Z(x,y)| * \delta_{k,SSA}(x,y)$. Let $\alpha(x,y)$ be the maximum-flow [17] possible between border nodes x and y, then the upper bound $\delta_{k,SSA}(x,y)$ on the advertised capacity must satisfy the following:

$$|Z(x,y)| * \delta_{k,SSA}(x,y) \leq \alpha(x,y) \tag{4}$$

$$\delta_{k,SSA}(x,y) \leq \lfloor (\alpha(x,y)/|Z(x,y)|\rfloor. \tag{5}$$

Basically, the upper bound calculated above distributes the maximum flow achievable from the border node x to y equally among all the VPNs in the set $Z(x,y)$.

Lower bound on advertised capacity. In computing the lower bound $\gamma_{k,SSA}(x,y)$, the goal is to try to satisfy with a high probability any single path bandwidth request $M_{k,i}(x,y)$ during the refresh interval period R_k from a VPN. The lower bound has to be computed carefully. A conservative lower bound estimate would lead to a VPN wrongly terminating the calls locally, while an overly optimistic estimate could lead to requests being blocked resulting in crankback of calls. Ideally, a lower bound $\gamma_{k,SSA}(x,y)$ should be set, so that the worst case simultaneous requests, which would be the arrival of $|Z(x,y)| * \gamma_{k,SSA}(x,y)$ capacity requests, can be satisfied. Let $P_{x,y}$ be the set of edge disjoint paths between border nodes x and y. Let $C_{x,y}$ be the minimum of bottleneck capacities of all the paths $p_i \in P_{x,y}$, then a lower bound can be obtained as follows:

$$|Z(x,y)| * \gamma_{k,SSA}(x,y) \leq (|P_{x,y}| * C_{x,y}) \tag{6}$$

$$\gamma_{k,SSA}(x,y) \leq \left\lfloor (|P_{x,y}| * C_{x,y}/|Z(x,y)|) \right\rfloor. \tag{7}$$

So, this approach would require computing all the edge disjoint paths between x and y Also, the lower bound computed using (7) may turn out to be very conservative because of a very poor bottleneck capacity in the set of paths $P_{x,y}$. So, we propose to use the flow obtained from an M-Route flow to compute the lower bound. The concept of an M-Route flow was first proposed by Kishimoto and Takeuchi [18] in an effort to solve the problem of communication channel survivability in the case of $(M-1)$ link failures. Kishimoto and Takeuchi [18] define and propose solutions to handle both link and node failures. Since the goal of the VPN-TA problem is to avoid contention for link resources among the VPNs hosted on a given border node, we limit our discussion to the edge disjoint version of this problem. An edge disjoint $M-$ Route flow between a source node s and destination node d can be defined using the cost of an elementary M-Route flow defined next.

An elementary M-Route flow from a node s to a node d is defined as a flow of one unit along each of M edge disjoint paths from node s to node d. The corresponding set of paths is called an M-path. An M-Route flow is a flow that can be expressed as a nonnegative linear sum of the elementary M-route flows.

The value of an M-Route flow is the sum of the values of the flows in all the paths defining the M-Route flow. Kishimoto and Takeuchi [18] also proposed an algorithm to compute a maximum M-Route flow which can be found in maximum of $(\mathrm{M}-1)$ runs of the max-flow algorithm [18] with a net complexity of $O(M * |V|^3)$. For a given value of M, let the value of M-Route flow from border node x to border node y be given as $\eta(x,y)$.

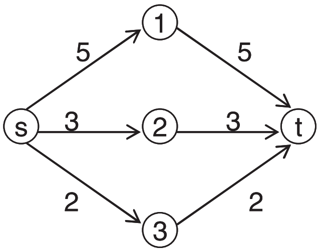


Fig. 8. M-Route flow example.

It is difficult to give a lower bound estimate that represents the single path capacity using an M-Route flow, since the flow represents an aggregate flow, which may not split equally among the M-paths as illustrated in Fig. 8. Here, the 2-Route flow possible from s to t is 10. This is obtained by augmenting two units each along paths $\{(s{-}{>}1{-}{>}t),(s{-}{>}3{-}{>}t)\}$ and three units each along $\{(s{-}{>}1{-}{>}t),(s{-}{>}2{-}{>}t)\}$. But, there are no paths between s and t that can handle two requests of five units simultaneously. So, since the split of such an M-Route flow is not known from the aggregate M-Route flow value itself, an absolute upper bound for $\gamma_{k,SSA}(x,y)$ cannot be obtained; however, we can state with high probability that M simultaneously requests each value less than or equal to $\lfloor \eta(x,y)/M \rfloor$ can be handled. Hence, the lower bound for the capacity of virtual link (*x*,*y*)could be set to

$$\gamma_{k,SSA}(x,y) = \lfloor \eta(x,y)/M \rfloor. \tag{8}$$

Now, we face the question of choosing an appropriate value for M. For an $M-$Route flow to exist, one can observe that M is upper bounded by the maximum number of edge disjoint paths $|P_{x,y}|$ between x and y. That is, $M \leq |P_{x,y}|$. Suppose we select $M = |P_{x,y}|$ while finding the value of $\eta(x,y)$. Then, $\eta(x,y)/|P_{x,y}|$ will be a pessimistic estimate of the capacity available between nodes x and y, if $|Z(x,y)| \ll |P_{x,y}|$.On the other hand it will be an aggressive estimate if $|Z(x,y)| \gg |P_{x,y}|$. It will be a very good estimate if $|Z(x,y)| \leq |P_{x,y}|$ and they are very close to each other. Hence, to accommodate these situations we set the lower bound for the capacity of the virtual link (*x*,*y*) as:

$$\gamma_{k,SSA}(x,y) = \lfloor \eta(x,y)/Z(x,y) \rfloor, \tag{9}$$

where $\eta(x,y)$ is the value of an M-route flow with $M = |P_{x,y}|$. The lower bound computed from M-Route flow in (9) is less conservative than the lower bound given in (7). We show this by comparing the lower bound values obtained by applying (7) and (9) with reference to the graph in Fig. 8, assuming $M = |Z(x,y)| = 2$, and for border nodes (s,t). Using (9), the lower bound $\gamma_k(s,t)$ is five units, compared to three units if (7) is used. This observation can be further generalized as:

$$\lfloor \eta(x,y)/Z(x,y) \rfloor \geq \left\lfloor (|P_{x,y}| * C_{x,y}/Z(x,y) \right\rfloor. \tag{10}$$

```
Mixed Bound Abstraction Algorithm
Input: $G(V,E)$, $U_b$ .
Output: $G_{k,SSA}(V_k,E_k)$, $\forall k \in U_b$ .
begin
Step1: identify the set of PE nodes $P_k$ belonging to VPN $k$ ;
        for all $x, y \in V_k$ , set $\delta_{k,,SSA}(x,y)=0$ and $\gamma_{k,,SSA,}(x,y)=0$;
Step 2: for each border node $x,y \in V_k$
          calculate max-flow $\alpha(x,y)$ between nodes $x$ and $y$;
          set $\delta_{k,SSA}(x,y) = \lfloor (\alpha(x,y)/|Z(x,y)| \rfloor$;
          set $M=$ $|P_{x,y}|$ (maximum number of disjoint
          paths between $x$ and $y$ );
          calculate $M$-Route flow $\eta(x,y)$ between nodes $x$
          and $y$;
          set $\gamma_{k,SSA}(x,y)= \lfloor (\eta(x,y)/|Z(x,y)| \rfloor$;

end;
```

Fig. 9. Mixed bound abstraction algorithm.

Assuming that the maximum number of disjoint paths between any two pair of border nodes $x, y \in B$ is upper bounded by $|P_u|$, the complexity of the mixed bound algorithm is dominated by the computation of the M-Route flow for each pair of border nodes. For the complexity analysis we assume the worst case maximum flow algorithm complexity of $O(|V|^3)$ [18]. This results in an overall complexity of $O(|U| * |B|^2 * |P_u| * |V|^3)$.

Note that, for this scheme (Fig. 9), we don't include a separate step to ensure fairness; as the flows determined by applying max-flow and the M-Route flow algorithms between a pair of border node is equally shared with the VPNs in the set $Z(x, y)$.

6.3 Steiner Tree-Based Abstraction Scheme

Though the mixed bound scheme is expected to perform better than the maximum capacity abstraction scheme, the approach could perform poorly in certain situations. This may be because the maximum flow and M-Route flow may split the exposed capacity across multiple paths, and as a result, may not necessarily guarantee a single path in order to accommodate the capacity requested by the VPN. Other drawbacks include the complexity of the algorithm and implementation changes required in order to realize it in practice, which is discussed in the next section. In view of these drawbacks, we propose a new algorithm, namely, the Steiner tree-based virtual capacity computing algorithm.

Given a graph $G(V, E)$ and a subset V' of V, assume that each link in E is associated with a cost. A Steiner tree of G with respect to V' is a minimum cost tree of G that contains all the nodes of V'. Note that the Steiner tree may also contain some other nodes that are not in V'. The goal of the proposed scheme is to build a tree for each VPN with the goal of minimally oversubscribing the residual link capacity and to use it to generate abstract topologies. The two main objectives of the tree computation process are to maximize the amount of capacity used for abstraction and, at the same time, minimize the capacity shared between the trees constructed for the different VPNshosted on a border node. The objectives of the Steiner tree construction and the VPN-TA problem are correlated, since minimizing the sharing of the link capacity among the VPN abstract topologies will result in more accurate abstractions, which are expected to improve VPN call performance and network utilization. We achieve the first objective by computing a minimum cost Steiner tree on the graph with link costs inversely proportional to the residual capacities of the links. Since the tree needs to be computed over a subset of border nodes P_k for a given VPN $k \in U_b$, a Steiner tree graph (on the border nodes of the VPN) is computed with the objective of minimizing the tree cost. In order to achieve the second objective of minimizing the capacity shared between the trees, we associate a weight with each link and increment the weight each time the link is included in the tree graph computed for a VPN.

We next discuss the abstraction algorithm, and highlight the steps required in order to compute the trees for each VPN. As in previously discussed algorithms, the trees are generated iterating over the set of VPNs U_b sequentially. The link weight variable to measure the number of times a link has been used in the trees constructed so far is denoted as $w(e)$. Initially, this link variable is set equal to a value of 1. Another cost variable $c(e)$, used to compute the Steiner tree is initialized as a function of $w(e)$ and the link residual capacity $r(e)$. We set $c(e)$ to $w(e) * (t(e))/(r(e))$; here $t(e)$ is the total capacity of the link e. The reason for such initialization is to ensure that the links with the more residual capacity would have a lower cost in comparison to links with higher residual capacity.

We next compute the Steiner tree, $ST(V_k, E_k)$, for a VPN k, with edge cost initialized to $c(e)$. For each VPN, Steiner trees are computed with respect to all the border nodes in the set P_k as the root node. After the subgraphs are computed for each of the VPNs, we set the capacity of the virtual link $(x, y) \in E_k$ of the fully meshed abstraction graph to the bottleneck capacity of the path connecting the root node x to the border node y in the Steiner tree $ST(V_k, E_k)$. Before continuing with the computation of subgraphs for the next VPN, the weight of the link variable $w(e)$ corresponding to $e \in E_k$ in $ST(V_k, E_k)$ is incremented by 1, and the edge cost $c(e)$ recomputed. This increased edge cost dissuades the future subgraph computations from using the links that are part of the already previously computed VPN subgraphs; it also enables links with lesser cost, i.e., more residual capacity and least used graph edges, to be considered during the tree computation process.

Note that in this scheme we apply the fairness criterion. For this, we iterate over all pairs of border nodes in the set $B_b = \cup_{t \in U} P_k$. Note that B_b is the set of all PE nodes of the VPNs hosted on the border node b. For each pair of border nodes $(b_1, b_2) \in B_b$, we determine the set of VPNs sharing this pair of border nodes and subscribing to the same abstract topology type parameter l. We represent this set as $Z_l(b_1, b_2)$. For this set of VPNs in $Z_l(b_1, b_2)$, for all $z \in Z_l(b_1, b_2)$ we set the virtual link capacity $w(b_1, b_2)$ in the corresponding abstract graph to the minimum of the capacities of virtual links (b_1, b_2) corresponding to the VPNs in the set $Z_l(b_1, b_2)$. For the case where there are no VPNs sharing a particular combination of border node pair, the set $Z_l(b_1, b_2)$ will be null.

Steiner tree computation is a strongly NP-complete problem. Literature provides good heuristics to compute Steiner trees. We use the nearest node Steiner tree algorithm for directed graphs from [19] for tree computation whose complexity to compute a Steiner tree rooted at a border node to the remaining border nodes (assuming $P_k \equiv B$) is $O(|B| * (|E| + |V| log(|V|)))$. The complexity of deriving a fully meshed abstract topology for a given VPN set U is

```
Steiner Tree Topology Abstraction Algorithm
Input: $G(V,E)$, $U_b$ .
Output: $G_{k,SSA}(V_k,E_k)$, $\forall k \in U_b$ .

begin

begin
for all edge $e \in E$ in $G(V,E)$,
        set $w(e) = 1$;
        set $c(e) = w(e)*(t(e))/(r(e))$;

(Iterate the remaining steps for each VPN $k \in U_b$.).
Step1: identify the set of PE nodes $P_k$ belonging to VPN $k$ ;

Step 2:Pick a border node $x \in P_k$ and compute the Steiner
        tree $ST(V_k, E_k)$ rooted at each border node, span-
        ning all the nodes in $y \in P_k$ and $y \neq x$ ;

Step 3: for each $(x,y) \in E_k$
        let $P = \{x, e_1, \ldots e_k, y\}$, $e_i \in ST(V_k, E_k)$; (Note: $P$ is the
path from $x$ to $y$ in the Steiner tree rooted at $x$);
        set $w_{k,SSA}(x,y) = \min\{r(e_1),\ldots,r(e_k)\}$; (Note: $w_{k.SSA}(x,y)$
is the bottleneck capacity path between nodes $x$ and $y$ in
$S(V_k, E_k)$);
        for each border node $x \in P_k$ ,
          for each Steiner tree $ST(V_k, E_k)$ rooted $x$,
            for each edge $e \in E_k$ in $ST(V_k, E_k)$ and $e \in G(V,E)$,
              set $w(e) = w(e) + 1$; set $c(e) = w(e)*(t(e))/(r(e))$;
end;
(Apply the fairness criteria.)
begin
Step 4: Determine set $B_b$ ;
          Determine set $Z_l(b_1, b_2)$, and
          Calculate $bw = \min_{z \in Z_l(b_1,b_2)} \{w_{z,SSA}(b_1,b_2)\}$ ;
          for each VPN $x \in Z_l(b_1, b_2)$
              set $w_{x,SSA}(b_1, b_2) = bw$ ;
end;
end;
```

Fig. 10. Steiner tree abstraction scheme for VPN topology abstraction problem.

$O(|U| * |B| * (|B| * (|E| + |V|log(|V|))))$. The pseudo code for this scheme is shown in Fig. 10.

7 Simulation and Performance Evaluation

In this section, we present results of simulations studied under different scenarios with the following objectives.

Objective 1. Demonstrate the usefulness of the TA service as a way to share the core topology and capacity information with the VPNs.

Objective 2. Study the fairness of VPNs in the context of VPN TA service and effect of the type of topology abstraction on core network utilization.

Objective 3. Evaluate the performance of the three decentralized TA generation schemes proposed in Section 6.

7.1 Performance Metric Definition

We introduced the call performance metrics in Section 5.2, which are success, crankback, and miscall ratios. These metrics are tracked centrally for each VPN. Following are the definitions for these metrics in relation to our simulation.

Success ratio. The success ratio is a measure of a VPN making a right routing decision using the abstraction provided to it by the VSP. This includes successful calls and the hit calls. The availability of the requested capacity is verified by recomputing the path with the exact state of the network

$$Success\ Ratio = \frac{Number\ of\ calls(\text{correctly accepted} + \text{calls correctly rejected})}{Total\ number\ of\ call} \cdot$$

Crankback ratio. A call would crank back, if there were no feasible path with the requested capacity in the core. The crankback ratio is defined as the ratio of the number of calls that have been cranked back to the total number of path requests made by the VPN

$$CrankBack\ Ratio = \frac{Number\ of\ calls\ cranked\ back}{Total\ number\ of\ calls}.$$

Misscall ratio. The misscall ratio is the ratio of calls that have been wrongly terminated locally at the VPN's end (even though there is enough resource to accommodate the call) to the total number of calls originated by the VPN; ideally, a successful TA service implementation should have a miss call ratio of zero

$$MissCall\ Ratio = \frac{Number\ of\ wrongly\ rejected\ calls}{Total\ number\ of\ calls}.$$

Average network utilization. This metric refers to the ratio of total link capacity utilized by active VPN bandwidth requests (aggregate utilized link capacity) to the total link capacity (aggregate link capacity)

$$Average\ Network\ Utilization = \frac{Aggregate\ utilized\ link\ capacity}{Aggregate\ link\ capacity} \cdot$$

7.2 Simulation Setup

The simulation study is conducted using OPNET [20], a well-known discrete event simulator. The topology used for studying the different scenarios is a 22-node random topology based on well-known Waxman's random graph [21] model with an average node degree of 4, $\alpha = 0.150$, $\beta = 2.2$; this topology is shown in Fig. 11. In order to validate our results over other standard topologies, we also studied the performance with respect to two other European networks referred from [22]. Considering the correlation in the results for the three topologies, we limit our results to those corresponding to the random graph. All the results discussed have been obtained from running the simulation for 30 independent replications to achieve 95 percent confidence interval for an *absolute error* [23] of less than 1 percent. The number of independent replications is deduced by observing the sample variance over several independent runs, and applying the approximation given in [23, p. 512].

In these topologies, five nodes were arbitrarily chosen as the border PE nodes and the remaining nodes chosen as the core nodes. Each of the PE nodes was configured to handle four different VPNs, three of which subscribed to the TA service. One of the VPNs was not enable to receive any TA

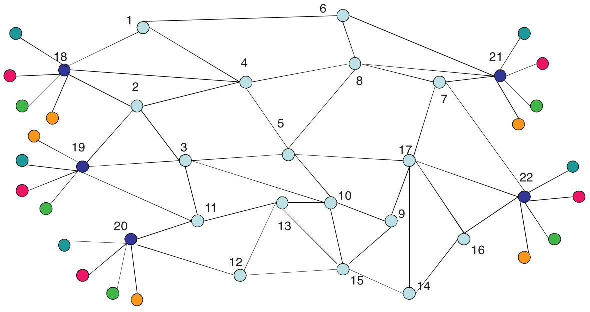


Fig. 11. Simulation topology.

Simulation Parameters	Obj. 1	Obj. 2	Obj. 3
VPN Mean Call Inter-arrival Time (s) (Exponential Dist.)	100	100	100
VPN Mean Call Holding Time (s) (Exponential Dist.)	[10,1000]	[10,1000]	[10-1000]
VPN Bandwidth Request Size (Uniform Distribution)	[1,500]	[1,500]	[1,500]
Core Topology Update Interval (s)	5	5	5
Abstract Topology Refresh Interval (VPN-A) (s)	20	20	20
Abstract Topology Refresh Interval (VPN-B) (s)	20	20	20
Abstract Topology Refresh Interval (VPN-C) (s)	20	20	20
Abstract Topology Type (VPN-A)	SSA	SSA	SSA
Abstract Topology Type (VPN-B)	SA	SA	SSA
Abstract Topology Type (VPN-C)	SNA	SNA	SSA
Abstract Topology Type (VPN-D)	NA	NA	SSA

Fig. 12. Simulation parameters.

service; this was done in order to compare its performance with the other VPNs and study the tradeoff of enabling TA service. For the simulation analysis, bandwidths for the access as well as the core links were initialized to 1,000 units. The bandwidth requests from the VPN client nodes were modeled as Poisson arrivals. The call holding times were assumed to be exponentially distributed. Without implementing a full-fledged flooding mechanism to maintain the link state database, the simulation implements a simple logic of having the link state database of all the PE nodes to be in sync with the state of the network. This is ensured by having the intratopology update interval for route computing engine in the PE nodes set to a value much less than the mean arrival rate of bandwidth requests; this value is 5 s for our simulation analysis (Note: The time interrupts for the events are scheduled during the simulation based on the absolute simulation clock; the units of seconds has been assumed in the simulation discussion for time related metrics to present the discussion in a practical context). Simulation parameter settings that we used are summarized in Fig. 12.

In order to generate the abstraction for the previously stated objectives 1-2, each border node applies the maximum capacity scheme (discussed in Section 6.1). As part of the key TA-SLA parameter initialization, for all the objectives, we assume that the VPNs subscribe to different abstraction types, i.e., SSA, SA, or SNA service. As noted previously, one of the VPNs was purposely not provided any abstraction service. This is indicated as (NA) in our results. This allows us to show the improvement over legacy approaches which do not deploy any TA service. For objective-3 all the VPNs subscribe to SSA type topology abstraction.

Objectives 1-2

Due to space limitations we only summarize below our conclusions based on our simulations for objectives 1 and 2 presented in Figs. 13a, 13b, 13c, and 14 for the case when the load is varied by varying the mean call holding time (H).

- Fig. 13a compares the success ratio of different VPNs subscribing to different TA types. We observed that VPNs with NA and SNA had their performance deteriorate significantly in terms of success ratio with increasing load. For VPNs with either SSA or SA type of abstraction, the significant improvement of the crankback and misscall ratio resulted in very good performance in terms of success ratio at high load. We also observed that among SSA and SA, SA performed relatively poorly. Figs. 13a and 13b together demonstrate the usefulness of the TA service in a dynamic bandwidth request scenario, particularly, at high network load conditions. The overall performance of VPNs with various TA subscriptions decreases with increasing load because of increasing network utilization as shown in Fig. 13b.
- From Fig. 13c, we can observe that the scenario with no abstraction and simple node abstraction resulted in 1-3 percent better network utilization than the VPNs with some form of TA; however, as we discussed earlier, this gain is at the cost of significant deterioration in terms of the call performance metrics (Fig. 13b). The better performance in terms of network utilization achieved by the SNA scenario is expected, as this abstraction is not provided any information about the link state of the core network, resulting in performance similar to the NA case, but with improved call performance statistics. We can also observe that, compared to the VPN which was provided SSA, the VPN with SA resulted in network utilization which is 4 percent poorer than the SSA case. The poorer performance of the SA case correlates with performance degradation of 5 percent observed with respect to the call performance metrics as discussed earlier. From the above observations, we can conclude that providing TA service, while enabling very good call performance to the VPNs, doesn't result in significant decrease in core network utilization.
- Fig. 14 shows the standard deviation (SD) of success, crankback, and miscall ratios for all the VPNs with increasing load. We observe that the SD of the performance metrics of the five VPNs is less than 2 percent, which indicates that the model of sharing resources equally results in the desired objective of achieving fairness to all the VPNs. We also observe that SD increases slightly at higher loads. This is expected because of increasing resource contention and lack of synchronization between the core topology information and the abstractions provided to the VPNs.
- Figs. 15a and 15b illustrate the effect of increasing the abstract topology update interval. SN has not been chosen for comparison as it is expected to perform poorly as noted previously.

The results show that with increasing R_k, the success ratio decreases while crankback ratio, and misscall ratio increases for each of the topology abstractions, which can be

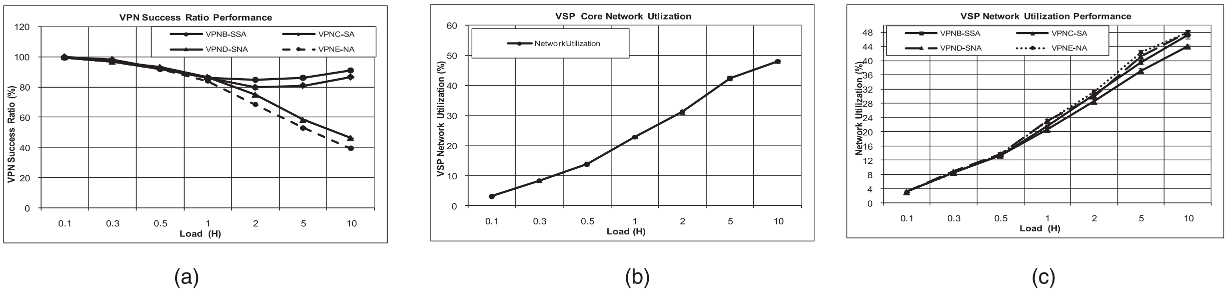


Fig. 13. (a) Objective 1, VPN call performance comparing success ratio with varying mean holding time (H), (b) Objective 1, VSP core network utilization with varying mean holding time (H), (c) Objective 1, VSP core network utilization with varying mean holding time (H).

attributed to increasing lack of the latest state of the core network information with the VPN CE nodes. Observing the maximum and minimum performance level from the graphs for each of the abstraction types in Fig. 15a, we see that the performance of the VPN with SA and SSA drops by 25 percent over the range of refresh intervals. SA performs poorly compared to SSA by 3 percent over the range of refresh interval values. This shows SSA does prove to be of advantage with increasing abstract topology refresh interval. Similar deterioration in performance is also noted with respect to crankback and misscall ratio in Fig. 15b among the two forms of abstractions.

Objective 3: Characteristics of abstraction schemes for TA generation. Due to space constraint, we only summarize the results pertaining to the performance of the abstraction schemes. The performance was analyzed by increasing the mean call holding time (H) and the bandwidth request size (X).

Success ratio. We observed that, among the three schemes, the maximum capacity scheme performs the best. Its success ratio decreased by about 15 percent as the load increased from 0.1 to 10 Erlangs. The next best scheme is the Steiner tree abstraction scheme whose success ratio performance is as good as that of the maximum capacity scheme at over the range of load conditions. The poorest performance is displayed by the mixed bound scheme, whose performance decreased by about 48 percent at the maximum load. The poorest performance of the mixed bound scheme is because of its conservative mode of exposing capacity which divides the available maximum flow and M-Route flow among all the VPNs equally to facilitate the PE node to handle worst case capacity requests from the VPNs simultaneously. This results in the CE nodes terminating calls locally resulting in poor miscall ratio performance which we discuss later. The best success ratio performance of the maximum capacity scheme indicates the usefulness of being aggressive in a dynamic bandwidth request scenario enabled by the TA service. A similar difference in performance among the abstraction schemes is also noted when bandwidth request size (X) is varied. The performance difference again points to the fact that, as the mean bandwidth request size increases, an aggressive scheme, such as the maximum capacity scheme, which tends to expose more resources, results in better successful route computation locally at the CE node in comparison to the other two schemes.

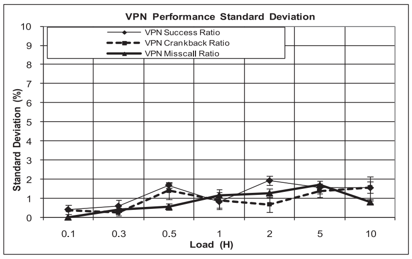
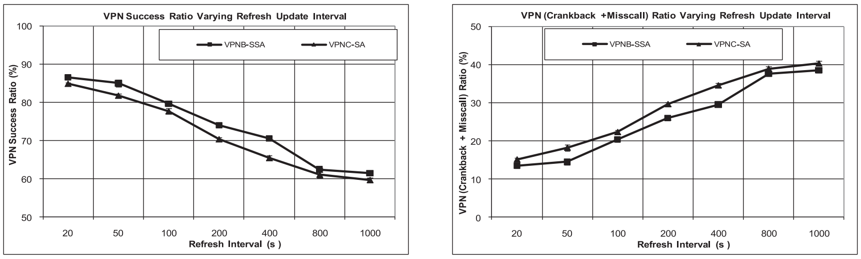


Fig. 14. Objective 2, Fairness performance in terms of standard deviation of call performance metrics with varying mean holding time (H).

Crankback ratio. The VPN crankback ratio for all the three schemes deteriorated with increasing load on the network. This can be attributed to two reasons: the first reason is the oversubscription, which is still an issue for the three schemes, and the second reason is that the increasing load causes a reduction in the correctness of the virtual capacity information associated with the abstraction with respect to the core state of the network. This causes calls to be successfully computed by the CE node locally but cranked back by the PE node due to a lack of resources in the core. Comparing the crankback performance of the abstraction schemes, we noted that the maximum capacity scheme performed poorer than the other two schemes (which illustrates one of the drawbacks of employing an aggressive scheme) despite the fact that it performed the best in terms of success ratio metric. The good performance of mixed bound scheme can be attributed to its conservative approach of assigning lower and upper bounds to the virtual links using the maximum flow and M-Route flow approaches. The performance of the Steiner tree scheme lied between the performances of the other two schemes.

Misscall ratio. With respect to miscall ratio the maximum capacity scheme displayed the best performance of the three algorithms. The next best performance is displayed by the Steiner tree abstraction scheme, which performed poorly when compared to maximum capacity scheme, but gains in terms of crankback ratio. The mixed bound algorithm displays the poorest performance of the three abstraction schemes with respect to the miss call ratio. Its performance is observed to deteriorate by 48 percent over the range of load conditions; the reason being that a lot of calls were rejected locally when the capacity demands of the VPNs exceeded the conservative lower bound of the virtual link capacity associated with the TA.

Core network utilization. It is important to note that although network utilization in the context of the TA service is influenced by the abstraction schemes, it also depends on the optimizing principles used by the core routing algorithm, which in this case is the constrained Dijkstra's algorithm. From simulations we observed that the maximum capacity abstraction scheme results in better network utilization than



(a) (b)

Fig. 15. (a) Objective 3, comparing VSP call performance with varying refresh interval, (b) Objective 3, comparing VSP call performance with varying refresh interval.

the other two abstraction schemes. Of the three schemes, the mixed bound scheme shows the poorest network utilization performance due to its high miscall ratio.

8 Conclusion

In this paper, we introduced the problem of using topology abstraction to share VSP's core topology and link re source information as a service to its IP-VPN customers. To realize this in practical networks we defined TA SLA parameters with the goal of generating service differentiation among IP-VPN customers subscribing to the TA service. Our TA generation approach guarantees that all the VPNs will share the network capacities fairly if they subscribe to the same abstract topology type parameter. We defined the VPN-TA that captures the oversubscription issue associated with the decentralized mode of TA generation. These abstraction schemes differ from one another with respect to their degrees of aggressiveness in terms of exposing resources, which result in tradeoffs in terms of VPN call performance metrics and core network utilization. Using extensive simulations, we showed the usefulness of VPN-TA service. From our study of the performance of the abstraction schemes, we conclude that the abstraction schemes demonstrated a tradeoff between the call performance metrics and network utilization. A VSP's choice of the abstraction scheme for decentralized TA generation will ultimately depend on its service objectives. We wish to conclude by drawing attention to the growing importance of the topology abstraction concept in other applications. For example, Kokkinos and Vavarigos [24] use this concept in the context of Grid computing services and Verdi et al. [25] discusses this in the context of interdomain QoS routing.

Our research so far has focused on topology abstraction service with respect to the availability of the bandwidth in the physical network. An interesting direction of future research would be to study robustness of topology abstraction service with respect to other parameters, in particular, robustness with respect to link or node failures in the physical network. A multicriteria optimization approach using sophisticated techniques such as machine learning would be required, Concepts and ideas from recent works in [26], [27] could be of help in this investigation.

References

[1] A. Nagarajan, "Generic Requirements for Provider Provisioned Virtual Private Networks (PPVPN)," RFC 3809, IETF, June 2004.

[2] P. Knight and C. Lewis, "Layer 2 and 3 Virtual Private Networks: Taxonomy Technology and Standardization Efforts," *IEEE Comm. Magazine,* vol. 42, no. 6, pp. 124-131, June 2004.

[3] C.T. Chou, "Traffic Engineering for MPLS Based Virtual Private Networks," *Proc. IEEE INFOCOM,* pp. 110-115, June 2002.

[4] N.G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K.K. Ramakrishnan, and J.E. van der Merwe, "Resource Management with Hoses: Point-to-Cloud Services for Virtual Private Networks," *IEEE/ACM Trans. Networking,* vol. 10, no. 5, pp. 679-692, Oct. 2002.

[5] A. Kumar, R. Rastogi, and A. Siberschatz, "Algorithms for Provisioning Virtual Private Networks in Host Model," *IEEE/ACM Trans. Networking,* vol.10, no.4, pp. 565-578, Aug. 2002.

[6] L. Zhang, J. Muppala, and S. Chanson, "Provisioning VPN in the Hose Model with Delay Requirements," *Proc. Int'l Conf. Parallel Processing,* pp. 211-218, June 2005.

[7] M. Kodialam, T.V. Lakshman, and S. Sengupta, "Traffic-Oblivious Routing for Guaranteed Bandwidth Performance," *IEEE Comm. Magazine,* vol. 45, no. 4, pp. 46-51, Apr. 2007.

[8] D. Mitra and I. Ziedins, "Hierarchical Virtual Partitioning: Algorithms for Virtual Private Networking," *Proc. IEEE GLOBECOM,* vol. 3, pp. 1784-1791, Nov. 1997.

[9] R. Garg and H. Saran, "Fair Bandwidth Sharing in Virtual Networks: A Capacity Resizing Approach," *Proc. IEEE INFOCOM,* vol. 1, pp. 255-264, Mar. 2000.

[10] R. Isaacs and I. Leslie, "Support for Resource-Assured and Dynamic Virtual Private Networks," *IEEE J. Selected Areas in Comm.,* vol. 19, no. 3, pp. 460-472, Mar. 2001.

[11] CISCO Telepresence Service. http://www.cisco.com/en/US/docs/solutions/Enterprise/Video/tpqos.html#wp1045102, 2012.

[12] E. Rosen and Y. Rekter, "BGP/MPLS Virtual Private Networks," RFC 4364, IETF, Feb. 2006.

[13] P. Knight and H. Ould-Brahim, "Network Based IP-VPN Using Virtual Routers," draft-ietf-l3vpn-vpn-vr-03.txt, Internet Draft, IETF, Mar. 2006.

[14] E. Rosen and P. Psenak, "OSPF as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs)," RFC 4577, IETF, Jan. 2006.

[15] T. korkmaz and M. Krunz, "Source-Oriented Topology Aggregation with Multiple QoS Parameters in Hierarchical Networks," *ACM Trans. Modeling and Computer Simulations,* vol. 10, no. 4, pp. 295-325, Oct. 2000.

[16] Z. Wang and J. Crowcroft, "Quality-of-Service Routing for Supporting Multimedia Applications," *IEEE J. Selected Areas in Comm.,* vol. 14, no.7, pp. 1228-1234, Sept. 1996.

[17] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network Fows: Theory, Algorithms, and Applications.* Prentice-Hall, 1993.

[18] W. Kishimoto and M. Takeuchi, "On M-Route Flow in Networks," *Proc. Singapore ICCS/ISITA '92,* 1992.

[19] M.S. Kodialam, T.V. Laxman, and S. Sengupta, "Online Multicast Routing with Bandwidth Guarantees: A New Approach Using Multicast Network Flow," *IEEE ACM Trans. Networking,* vol. 11, no. 4, pp. 676-686, Aug. 2003.

[20] OPNET (Discrete Event Simulator), www.opnet.com, 2012.

[21] B.M. Waxman, "Routing of Multipoint Connections," *J. Selected Areas in Comm.,* vol. 6, pp. 1617-1622, Dec. 1988.

[22] K. Shing Ho and K. Wai Cheung, "Generalized Survivable Networks," *IEEE/ACM Trans. Networking,* vol. 15, no. 4, pp. 750-760, Aug. 2007.

[23] A.M. Law and W. David Kelton, *Simulation Modelling and Analysis,* third ed. McGraw Hill, 2000.

[24] P. Kokkinos and E. Vavarigos, "Resource Information Aggregation in Hierarchical Grid Networks," *Proc. IEEE/ACM Symp. Cluster Computing and Grid (CCGRID),* pp. 268-275, 2009.

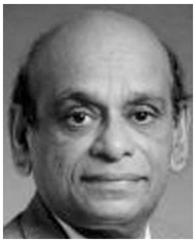
[25] F.L. Verdi, M.F.M Aes, E.R.M Madeira, and A. Welin, "Using Virtualization to Provide Inter-Domain QoS Enabled Routing," *J. Networks,* vol. 2, no. 2, pp. 23-32, 2007.

[26] A. Udenze and K.M. Maie, "Dyna-Routing: Multi Criteria Reinforcement Learning Routing for Wireless Sensor Networks with Lossy Links," *Ad Hoc and Sensor Wireless Networks,* vol. 11, nos. 3/4, pp. 285-306, 2011.

[27] D. Traskas and J. Padget, "A Multi-Agent Systems Approach to Call-Centre Management," *Int'l J. Parallel, Emergent and Distributed Systems,* vol. 26, no. 5, pp. 347-367, 2011.

Ravishankar Ravindran received the PhD degree in electrical engineering from Carleton University, Ottawa, Canada and the MS degree in computer science from the University of Oklahoma, Norman. He is currently a senior researcher at Huawei's Research Center, Santa Clara, conducting research in area of future Internet architectures. Before this, he was a part of Nortel's Advanced Technology group where he conducted research in the areas of Optical Networking, QoS Routing, Control Plane Architectures related to IP/(G)MPLS, 4G Wireless Research, and End-to-End QoE/QoS Engineering for Multimedia Applications. He is a senior member of the IEEE.

Changcheng Huang received the BEng degree in 1985 and the MEng degree in 1988 both in electronic engineering from Tsinghua University, Beijing, China. He received the PhD degree in electrical engineering from Carleton University, Ottawa, Canada in 1997. From 1996 to 1998, he worked for Nortel Networks, Ottawa, Canada where he was a systems engineering specialist. He was a systems engineer and network architect in the Optical Networking Group of Tellabs, Illinois, during the period of 1998 to 2000. Since July 2000, he has been with the Department of Systems and Computer Engineering at Carleton University, Ottawa, Canada where he is currently an associate professor. He won the CFI new opportunity award for building an optical network laboratory in 2001. He was an associate editor of *IEEE Communications Letters* from 2004 to 2006. He is a senior member of the IEEE.



Krishnaiya Thulasiraman received the PhD degree in EE from IIT Madras, India, in 1968 and has been a professor and holds the Hitachi chair in CS at the University of Oklhoma since 1994. His prior appointments include: IIT Madas (EE/CS, 1965-1981) and Concordia University , Montreal (ECE, 1981-1994). His research has been in graph theory, combinatorial optimization, algorithms, and applications in a variety of areas in CS and EE. He has coauthored with M.N.S. Swamy two text books *Graphs, Networks, and Algorithms* (1981) and *Graphs: Theory and Algorithms* (1992), both published by Wiley Inter-Science. He has received several awards and honors: Distinguished Alumnus Award of IIT Madras(2008), Fellow of the American Association for Advancement of Science (2007), 2006 IEEE Circuits and Systems Society Technical Achievement Award. Endowed Gopalakrishnan Chair Professorship in CS at IIT, Madras (Summer 2005), Elected Academician of the European Academy of Sciences (2002), IEEE CAS Society Golden Jubilee Medal (1999), and Senior Research Fellowship of the Japan Society for Promotion of Science (1988). He has held visiting positions at the Tokyo Institute of Technology, University of Karlsruhe, University of Illinois at Urbana-Champaign , Chuo University, Tokyo, University of Waerloo and the National Chia-Tung University, Taiwan. He has been very active professionally: vice president (Administration) of the IEEE CAS Society (1998, 1999), technical program chair of ISCAS (1993, 1999), deputy editor-in-chief of the *IEEE Transactions Circuits and Systems I* (2004-2005), co-guest editor of a special issue on "Computational Graph Theory: Algorithms and Applications" (IEEE Transactions CAS , March 1988) , TPC member of INFOCOM, founding regional editor of the *Journal of Circuits and Computers*, etc. He is a fellow of the IEEE (1990).