A Control Architecture for Lightweight Virtual Networks

Sean Rooney

IBM Zurich Research Laboratory
8803 Rüschlikon, Säumerstrasse 4, Switzerland
sro@zurich.ibm.com

Abstract. As an IP network is supportable over multiple different layer 2 networks, so a virtual network is supportable over multiple different resource allocation mechanisms. This fact is obscured in the literature as the goal — the privileging of certain user traffic by segregating it from other traffic — is tightly bound with the means of achieving it — e.g., the differentiation of traffic at routers based on the bits in the type of service field of the IP packet. We propose an IP control plane capable of supporting the dynamic creation of lightweight virtual networks and requiring minimal support from network devices. The latter is achieved by minimizing what is required from network elements, while taking advantage of what is available.

1 Introduction

At a fundamental level network policy is an emergent property of differentiating between traffic. Traffic differentiation can be distinguished based on the following three principles:

- the reason for doing it, e.g. QoS, security;
- the granularity at which it is performed; e.g. flow, user, service, site;
- the scope, e.g. between hosts, between edge routers.

Existing solutions address different parts of the problem space defined by this tuple. For example what is commonly called in the literature a Virtual Private Network (VPN), such as the commercial offer proposed in [1], secures traffic in transit between trusted locations; guaranteeing resources between hosts is addressed by the IETF integrated service model (int-serv) [2]; while the IETF differentiated service model (diff-serv) [3] allows a guarantee to be given about the forwarding of the aggregate of traffic across a router for a given service.

The diversity of ways in which policy is defined for, distributed to and applied on different network devices makes the configuration of the network complex and reasoning about its overall behavior extremely difficult; this is likely to inhibit the introduction of new desirable policies. We propose a single abstraction within which different solutions at different protocol layers can be coordinated. This abstraction interconnects applications resident on various edge elements such as

A. Ambler, S.B. Calo, and G. Kar (Eds.): DSOM 2000, LNCS 1960, pp. 13–24, 2000.

servers, edge-routers and hosts, in such a way that their communication can be distinguished from other traffic carried over the same physical infrastructure. As the abstraction isolates traffic from one user group from others it effectively virtualizes the physical network and is properly termed a virtual network. The qualifier *lightweight* is used to denote that these virtual networks are created quickly and make minimal requirements from the physical network over which they are overlayed.

Lightweight Virtual Networks (LVN) are currently being used for supporting network policy within a framework for the dynamic addition of new content providers to an Application Service Provider's (ASP) infrastructure [4]. The LVN is the means by which the ASP's distribution network is partitioned between the diverse clients and the location at which QoS guarantees are enforced.

We motivate the feasibility of the approach by describing a working implementation of a control architecture for creating LVNs.

2 LVN Control Plane Requirements

A Lightweight Virtual Network (LVN) is an overlay network allowing a community of information producers and consumers to exchange information such that at the network nodes their traffic can be distinguished from other traffic and can be treated according to some community specific policy. The constraints on the LVN control plane are such that it must:

- *potentially be end-to-end*: often policy, for example security, needs to be enforced everywhere or not at all.
- *involve both layer 2 and 3 devices*: the distinction between routers and switches is becoming blurred. Routers are forwarding at or near switching speeds, while Ethernet switches, for example, are becoming more sophisticated in regard to the policy they can support.
- *make use of existing technology and run in a heterogenous environment*: a successful solution must build on the large range of existing technologies for supporting diverse types of policy.
- *make minimal requirements of the network nodes*: while use will be made of any technology for supporting policy resident at a node, the solution must make as few demands as possible on the network nodes. Equipment as diverse as a simple Ethernet hub and an MPLS Router should fit into the same framework.
- *be dynamic and self sustaining*: LVNs are created dynamically without the intervention of a human operator.

The policy to be applied to traffic on a given LVN must be distributed to all appropriate nodes along with a label. Each data unit belonging to the traffic of that LVN carries the label enabling the node to identify the appropriate policy to use when forwarding the unit.

Network policy is the coordination of the diverse forwarding functions of a set of network devices to attain some overall objective, for example security through

packet filtering, guaranteed service quality through priority based forwarding. No attempt is made to define policy precisely as this will necessarily evolve as network elements obtain new capabilities.

A LVN label can be something implicit, e.g. an IP source address, or explicit, e.g. an MPLS label [5]. The label may be present only at the IP layer or also be usable by layer 2 devices such as Ethernet and ATM. It may have significance for the network as whole, e.g. a VLAN [6] identifier, or have only local significance, e.g. an ATM VCI. Using implicit labels for identifying policy does not require modifying the format of the data units. They can therefore be carried by nodes that do not support the LVN infrastructure. However, they are more restrictive as they simply use parts of the format of the data unit for a purpose for which they were not intended. Explicit labels can be used exclusively for the purpose of identifying policy, but require changing formats of well established data units (or creating new encapsulation for them) with all the inherent problems. Moreover, there is no general agreement about what such a label should be like, e.g. MPLS, IPv6.

2.1 Design Choice

The chosen solution for the infrastructure is to use implicit labeling; no changes are required in the format of data units and no modification in the forwarding function of the nodes are required.

As the LVNs are supported both end-to-end and across multiple network layers, the implicit label chosen has to be meaningful to a large range of different network devices. The TCP/IP protocol suite is sometimes described in terms of an hour glass shape: many different protocols exist above and below the IP layer, but are unified at the IP layer. Although, in theory, layer 2 protocols should be oblivious to IP addressing, in practice they are often IP aware. For example an Ethernet switch may limit a broadcast to a VLAN containing only hosts within a given IP subnet. It achieves this by maintaining a MAC address/IP address mapping; this mapping is obtained by sniffing ARP requests and replies. If this type of sniffing is not supported then there is still the potential to use policy to enhance the IP address/Physical Address resolution, for example, a given set of IP addresses are resolved to both an Ethernet MAC address *and* a specific VLAN priority tag.

IP's ubiquity makes the IP address a good candidate for the implicit label within the prototype implementation.

Multiple LVNs should be able to coexist on the same host and the presence of LVNs on a given host should not affect its normal operation. Therefore, the LVN is not associated with the IP address of the host itself, but rather with a dedicated *private* IP address; the host has one such private address per LVN. Including socket identifiers as well in the label allows the discrimination to be carried right to the application layer, e.g. two instances of the same application on the same host may communicate with a given server using different LVNs.

At the IP layer a LVN appears as a set of private IP subnets. These subnets are created dynamically across the physical network and the participating end-

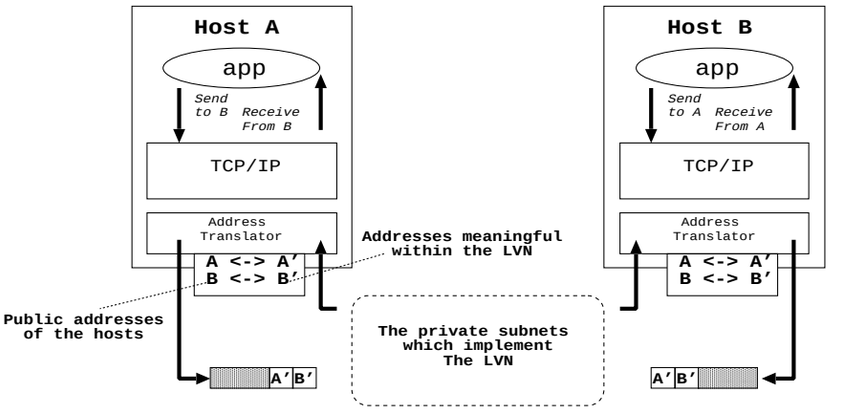

Fig. 1. Public/Private Mapping

points in the LVN are informed of the private-to-public mapping they should use in order that their traffic be carried across that LVN. The private IP address is the label which allows each node to determine the policy to use for forwarding that traffic. These private IP addresses are not exchanged by routers and therefore addresses conforming to RFC 1918 could be used, however as the LVN exists only over a well defined set of sites, any address space convenient for those sites is usable. Different administrative domains negotiate at LVN creation in order that the address space allocated to the LVN during its lifetime is used by it alone across all the involved domains. How this is achieved is explained in Section 3. The addresses used by a LVN are available for reuse after it is deleted from the network.

Figure 1 shows two hosts with addresses `A` and `B` participating in a LVN. First a private IP subnet (or set of subnets) is created across the nodes that interconnect the hosts. Each of the participating hosts receive mapping: `A`→`A'`, `B`→`B'`, such that `A'` and `B'` are meaningful addresses within the LVN's private subnets. The hosts update themselves such that if `A` wishes to communicate with `B` within the context of the LVN then `A` should send with source address `A'` and destination address `B'`. The network nodes identify the traffic within the LVN using the private address apply the LVN specific policy when forwarding it.

Within the dynamic ASP infrastructure described in [4] instead of mapping public-to-private addresses within the protocol stack, multiple virtual servers, each with their own private IP address are started on the ASP's physical servers on behalf of a content provider. Proxies with public DNS registered addresses running at the edge of the ASP's network are used by end-users to access the content; the proxies communicate with the appropriate virtual servers, and the LVN is used to distinguish the traffic of one content provider from another within the ASP's network.

3 LVN Admission Control

The creation of virtual overlays requires coordination of the participating entities before creation can occur, for example keys must be exchanged if encryption is required, pricing negotiated agreed upon, etc. This is especially true when multiple administrative domains are involved. In consequence, the authorization function is split into two: first, the application communicates to an authorization entity its requirements for the virtual network and obtains from this entity the system wide information that participating nodes will need to know in order to create it; second, the application propagates this information to each of the network nodes which will then make a local decision to accept and forward or reject and rollback.

3.1 Design Choice

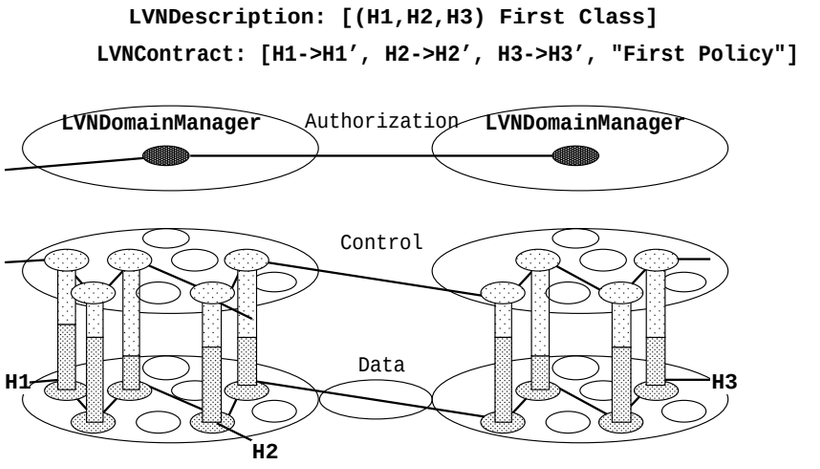

Fig. 2. Authorization

In the current implementation, the authorization entity is named the `LVNDomainManager`, the abstraction the application passes to this entity is the `LVNDescription` and the system wide state it returns the `LVNContract`. `LVNDomainManagers` are federated such that there is one per administrative domain and they intercommunicate in order to obtain the system wide information.

The `LVNDescription` is simply a set of IP addresses of the end-points of the intended LVN (in the case where the LVN is supported from application to application, socket identifiers are included as well) and a policy description.

The `LVNDomainManagers` collaborate together in order to resolve the `LVNDescription` into a `LVNContract` acceptable to all participants. In the current implementation the contract is a set of mapping of public addresses to private ones. More generally the contract would contain additional information such as encryption keys or different LVN label formats, e.g MPLS labels, VLAN identifiers, used to create the end-to-end LVN.

When a `LVNDomainManager` receives a `LVNDescription`, it determines which addresses are local to the domain and which foreign. For any foreign address the `LVNDomainManager` contacts (via a directory service) the appropriate `LVNDomainManager`. If no such entity exists then the LVN creation can still continue but the policy can only be applied within the local administrative domain. The `LVNDomainManager` must decide if this is appropriate. For example, in the current implementation the `LVNDomainManager` will continue the LVN creation but the public addresses for the foreign hosts will be mapped to themselves. Figure 2 shows the patterns of interactions between entities required for the creation of LVNs across multiple domains involving three hosts with addresses `H1`, `H2` and `H3`.

4 LVN Adaptors

Both layer 2 and layer 3 devices can offer support in regard to supporting policy. For example an IP router might support diff-serv priority based queuing on bits in the IP header, an IEEE 802.1q [6] enabled Ethernet switch priority switching based on frame tags, and an ATM switch continuous bit rate cell forwarding based on the virtual circuit identifier. All might be required in order to give an end-to-end performance guarantee. However, the label, the attachment to the policy and the means of disseminating the association are all very different. In order to be as general as possible the infrastructure supporting the virtual networks should be able to handle multiple different technologies.

The principle adopted is that the underlying node should try to do as *much as it can* to support the virtual network, but its exact behavior is a function of the technology and the precise capabilities of the element. This is called the *principle of low expectations*. By adopting such a weak semantic a wide variety of technology can be coordinated within the same framework. The LVN control plane guarantees connectivity for the traffic of the LVN, but gives no precise guarantee as to how the associated policies are enforced. The behavior of the LVN control plane may be thought of as a type of 'best-effort policy support', it only guarantees not to make the situation worse for supporting the policy on its associated network elements, it does not guarantee to make it any better.

4.1 Design Choice

In the prototype a `LVNController` entity is associated with each node within the physical network. The controllers support a common interface for LVN control independent of the nature of the underlying element. Within each controller

is an adaptor that maps generic LVN operations onto a particular technology and then further onto a particular instance of that technology, i.e. the model of switch, router, etc. So for example, while a router and an Ethernet switch both support a `createLVN` operation taking a `LVNContract` as argument, their effect is very different: the router updates its routing tables to reach the newly created private IP subnets with an appropriate forwarding class, while the Ethernet switch might support the LVN through the creation of a IP rule based VLAN with associated priority.

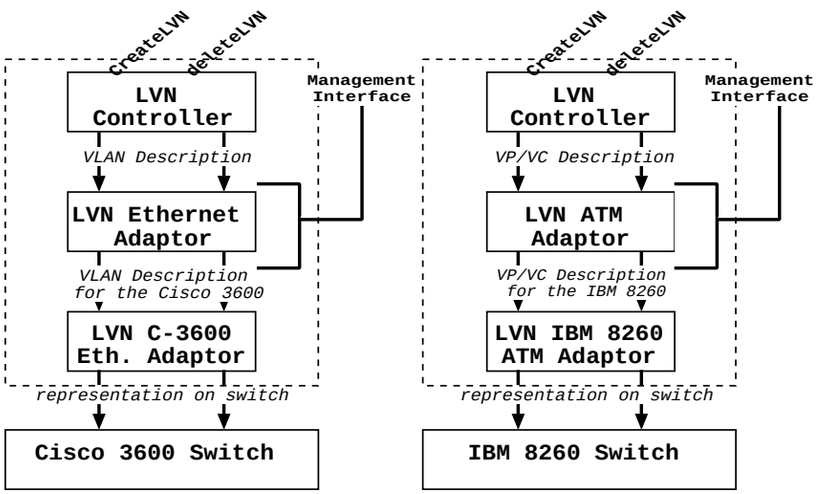

Fig. 3. Example of Adaptors

Figure 3 shows an example of two adaptors for two different technologies and pieces of equipment.

The `LVNController` may be thought of as a convenient abstraction in which to collect together a variety of architectural entities currently supported by network elements. For example, RSVP [7], COPS [8] the MPLS Label Switching Router [5] etc., would all be constitutions of the controller. The controller supplies an interface for LVN control general enough and with weak enough associated semantics such that it is supportable over a range of technologies, but the actual implementation of the operations is performed using existing technologies. The adapter is nothing other than a specification of how the mapping is performed between the `LVNController` interface and a network element with a given set of capabilities.

The exact nature of the mapping is set and adjusted by the network operator via management. For example, a new enterprise specific policy ('The-quarterly-report-broadcast-policy') can be added to an LDAP enabled directory along

with the precise mappings to be used for a range of technologies. When the `LVNController` encounters an unknown policy in the `LVNContract` it communicates with the directory in order to obtain the appropriate mapping.

5 Propagating LVN Control Messages

To create an LVN, the entire set of specified edge elements and some set of network elements interconnecting them must be made aware of the policy and the associated label. The normal network is used to carry the signaling messages that instigate the creation of the virtual networks over which the data will flow. From the point of view of the LVN control plane, the public network is the signaling channel — the path that signaling messages take will be determined by normal IP routing. While it would be possible to define a new signaling protocol for achieving this task, reusing existing protocols is preferred.

RSVP [7] is a signaling protocol designed for the creation of IP flows. Adopting RSVP as the means of LVN creation has the advantage of using a well known and established protocol. RSVP soft state model allows the LVNs to have the desired self sustaining property, the use of IP routing to determine the topology of the virtual network means that support from the network elements is minimal and the fact that RSVP does not specify the precise form of the flow specification allows for arbitrary policy/label associations to be carried in RSVP messages.

However, RSVP as defined in [7], does not match exactly the needs of the LVN control plane as RSVP uses IP routing to determine the next hop of the control message, within the LVN control plane layer 2 devices must also receive the RSVP messages. Moreover, RSVP is designed for applications with a small number of senders and a large number of receivers, the advertisement of a flow (via a Path-Message) is performed by an information producer, and the creation of the flow initiated by an information receiver. In consequence RSVP is inherently asymmetric; the LVN described in this paper should satisfy a larger class of applications. For example, within a ASP traffic is carried both to and from the servers. The final problem is that the flows that RSVP creates are trees, with the sender as root and the receivers as leaves (there may be several senders in which case it is a forest). The LVN is (by definition) a graph.

5.1 Design Choice

In the prototype an enhanced RSVP daemon is run as part of the `LVNController`. In order to allow RSVP message to be propagated to both layer 2 network nodes and routers, the forwarding function of the daemon is a function of the nature of the device it controls. Routers use their routing tables in order to identify the next router to forward a message to in order to reach a given end-point. They then use physical topology information to determine which is the next layer 2 hop to take them to that router, the message is then forwarded to that layer 2 entity with the next router as subtarget.

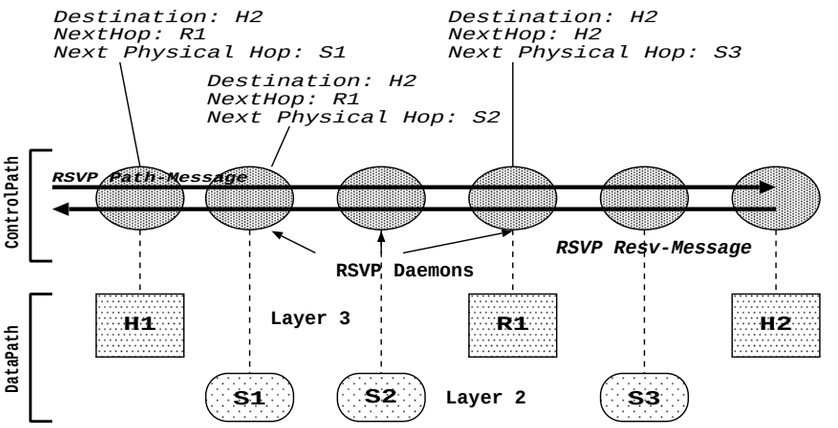

Fig. 4. LVN Creation Using RSVP Messages

The RSVP daemon of layer 2 entities only attempts to find the next hop to reach the subtarget rather than the end-point, they do this using physical topology information, so for example an Ethernet switch needs to know the next Ethernet switch to cross to get to an IP router. The view of the physical network that a `LVNController` is required to have is restricted to the boundaries of the set of layer 2 technologies to which it is attached. Layer 2 topology discovery is technology specific. For implementation purposes a simple Ethernet discovery system has been created. In this implementation each Ethernet `LVNController` registers its port/MAC association with a `LVNDomainManager` and the `LVNDomainManager` correlates the information to obtain the view of the entire Ethernet. This method could easily be replaced by appropriate layer 2 discovery mechanisms if available in the environment.

If a layer 2 network element is not discovered then it is transparent to the control path and no RSVP is forwarded to it. This does not mean that data for the LVN will not cross that network element, but only that no policy can be enforced on it. For example, if a host is connected to a router across an Ethernet hub and the hub does not support a `LVNController` then it is not discovered and the RSVP message from the host is forwarded directly to the router. This is simply another application of "the low expectations principle".

Among the set of end-points specified in the contract one is nominated as the initiator of the LVN creation. In the current implementation the element that requests the creation is considered the initiator, but another end-point might be chosen (perhaps based on its location in regard to all other end-points).

The initiating end-point starts by emitting RSVP Path-Messages, passing in the message the `LVNContract`. Each `LVNController` that receives a message, examines the contract to determine if it can support it and if so reserves the required resources. If it is the `LVNController` of a network node it then forwards

the message to the next `LVNController` upstream, if on the other hand it is the `LVNController` of an end-point specified in the contract and it is willing to support the virtual network creation, it replies by sending reservation messages downstream. A `LVNController` receiving a first reservation message for a given virtual network activates the policy on the node, i.e. creates a VLAN on the Ethernet switch, a virtual circuit on an ATM switch and updates the routing table on the router.

Figure 4 shows the pattern of communication for creating a LVN between two hosts `H1`, `H2`, interconnected at the IP layer by a router `R1` and at layer 2 across three switches `S1`, `S2` and `S3`.

Implicitly a tree shaped LVN is created with the initiator as the root of the LVN and the others end-points as leaves. Although reachability is guaranteed, the route taken may be extremely inefficient, i.e. all communication between leaves must travel across their nearest common ancestor in the LVN tree. The `Path-Message` that an end-point receives contains the `LVNContract`, therefore the receiving end-point has access to the complete set of other end-points participating in the virtual network, allowing it to locally decide if there is a better route between itself and one or more of the other non-sender nodes rather than simply going through the sender. If it decides that a better route exists it starts sending out `Path-Messages` carrying the same contract as it received. The `LVNController` will not forward `LVNContracts` over interfaces where they are already supported; normal routing will ensure that the better route to the other hosts are included as part of the virtual network. In summary, a tree is created and then if better paths between the leaves exist, the leaves are joined in order to make a graph.

The LVN policy is soft state and is maintained by the periodic reception of RSVP messages carried in the public network. If the RSVP messages are carried along a different path due to routing changes, then the shape of the LVN is modified accordingly. This permits the LVNs to dynamically adapt to the current state of the network.

6 Related Work

Virtual Network Research: [9] introduced the notion of a *switchlet*, i.e. a switch partition that allows multiple control architectures to be supported over the same physical switch, [10] describes the Tempest framework in which switchlets are used. Several other research group have made similar proposals [11,12]. That work is weak in its: applicability to technologies such as Ethernet, which have little support for traffic seperation; in the interoperation of multiple technologies. The work described here addresses these issues by using IP as the glue and applying a type of 'best-effort' partitioning mechanism. While the Tempest allows network operators to associate distinct control systems with their set of switchlets, the LVN is simply created using one of a set of predefined policies; i.e. it is a lowest common denominator. An LVN could be supported using

switchlets and this would be the preferred adaptor if the environment supported it.

IP QoS: Diff-serv [3] allows IP routers to apply a certain QoS policy to an aggregate of traffic associated with a given service. Traffic at a router is distinguished using the Type Of Service (TOS) field in the IPv4 header. Diff-serv is normally only enabled at edge routers. The policy/label association is typically manually set by a network operator directly on the router or added to a directory and distributed to the routers via a directory access protocol. The work described in this paper differs from diff-serv in that it attempts to use more general policies than just those concerning QoS, associate them with other granularities than traffic aggregations, apply them potentially from application to application and do all this dynamically. Moreover, the work in this paper attempts to make *direct* use of the sophisticated support offered by layer 2 devices for supporting policy. However, the LVN control plane makes use of diff-serv if enabled at the routers within its domain.

MPLS Based VPNs: Much work has been done on using MPLS [5] to instrument VPNs. For example, [13] describes how MPLS tunnels can be used to support IP based VPN between MPLS enabled routers capable of supporting VPNs; it terms such routers VPN Border Routers (VBRs). The VBR implements a virtual router for each VPN they support which in its turn services a separate forwarding table. The VBR forwards traffic from an enterprise across appropriate MPLS tunnels to a remote VBR at the appropriate site. VBRs are configured such that they have an interface address in the enterprises address range. VBRs for the same enterprise discover each other and exchange routing information using normal routing protocols.

The architecture in [13] is complementary to the work described in this paper. Traffic from multiple LVNs could be aggregated over a single MPLS VPN, or a dedicated VPN could be assigned to a LVN. The LVN-RSVP messages would be carried transparently over the MPLS tunnels to the correct sites. To integrate the two approaches the VBR must be capable of supporting the dynamic addition of new private subnets to and from the enterprise.

7 Conclusion

The diverse support for policy across different network technologies and in different scopes makes implementing the policy end-to-end problematic. We have proposed a unifying abstraction — a lightweight virtual network — with which a set of different policies may be associated across a range of technologies. Such lightweight networks are created dynamically without the intervention of human operators and have minimal requirements for the policy support of the underlying physical network. The virtual network control plane takes advantages, if available, of any support, for example diff-serv, ATM P-NNI, offered by the underlying devices without requiring their presence. This paper has motivated the feasibility of the approach by describing an infrastructure which supports the creation and management of such lightweight virtual networks. Current work is

using LVNs as part of an integrated approach in supporting resource guarantees within an Application Service Provider's distribution network.

References

1. Cisco, "Virtual Private Networks(VPN)," *Cisco System Inc. marketing information*, 1999.
2. R. Braden, D. Clark, and S. Shenker, " Integrated Services in the Internet Architecture: an Overview," *Internet RFC 1633*, June 1994.
3. D. Black, S. Blake, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Service," *Internet RFC 2475*, May 1998.
4. S. Rooney, "The ICorpMaker, a Dynamic Infrastructure for ASPs," *Proceedings of IEEE Workshop on IP Operations & Management*, Sept 2000.
5. E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," *draft-ietf-mpls-arch-06.txt*, August 1999.
6. IEEE/ISO/IEC, "Virtual Bridged Local Area Networks," *ISO Publication*, July 1998. Draft Standard: IEEE Standard for Local and Metropolitian Area Networks, P802.1Q/D11.
7. L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: a new resource ReSerVation protocol," *IEEE Network*, vol. 7, pp. 8–18, September 1993.
8. J. Boyle, R. Cohen, D. Durham, S. Herzog, R. Rajan, and A. Sastry, "The COPS Common Open Policy Service Protocol," *Internet RFC 2748*, January 2000.
9. J. van der Merwe and I. Leslie, "Switchlets and Dynamic Virtual ATM Networks," in *Integrated Network Management V*, pp. 355–368, Chapman & Hall, May 1997.
10. S. Rooney, J. van der Merwe, S. Crosby, and I. Leslie, "The Tempest, a Framework for Safe, Resource Assured, Programmable Networks ," *IEEE Communications Magazine*, vol. 36, pp. 42–53, October 1998.
11. A. Campbell and al, "The Genesis Kernel: A Virtual networking operating system for spawning network architectures," *Openarch'99*, March 1999.
12. W. Ng, A. Jun, H. Chow, R. Boutaba, and A. Leon-Garcia, "MIBlets: A Pratical Approach to Virtual Network Management," in *Integrated Network Management VI*, IFIP & IEEE, Chapman & Hall, May 1999.
13. L. Casey, I. Cunningham, and R. Eros, "A Framework for IP Based Virtual Private Networks." IETF draft-casey-mpls-vp-00.txt, November 1998. Work in progress.