RZ 3211 (# 93257) 03/06/00
Computer Science/Mathematics 18 pages

Research Report

A Control Architecture for Lightweight Virtual Networks

Sean Rooney

IBM Research
Zurich Research Laboratory
8803 Rüschlikon
Switzerland

A Control Architecture for Lightweight Virtual Networks

Sean Rooney

IBM Research, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland

Abstract

This paper proposes an IP control plane for supporting policy based overlays. It argues that the appropriate abstraction with which to associate policy is a lightweight virtual network. The term lightweight denotes that these virtual networks are created quickly through signaling and require minimal support from network devices. The latter is achieved by minimizing what is required from network elements, while taking advantage of what is available; this is termed *the principle of low expectations*.

The paper also motivates the feasibility of the solution by describing the design and implementation of a control architecture supporting such virtual networks. Within this infrastructure the virtual networks are implemented at the IP level by a set of related private IP subnets, while at lower layers a range of techniques for supporting policy is used. The infrastructure may be thought of as a virtual network control plane within which various technologies from a range of protocol layers are knitted together.

1 Introduction

The term policy based networking is becoming increasingly popular in both the research and product literature. A network policy is simply a way of determining who has access to which network nodes and how their traffic should be treated at those nodes. While policy has always existed within IP networks, this was obscured by the fact that it was implicit (e.g. all authorized users treated equally), simple (e.g. a single policy applied to the entire network) and modified only infrequently.

The interest in policy based networking arises from the increasing complexity of the policy that IP networks are required to support. This complexity is due to:

- increased requirements for resource guarantees, either because of the introduction of continuous media services or simply because the user's business requires bounded delays;
- the need to dynamically create private location independent user communities whose traffic is isolated and protected from the rest of the network;
- the fact that a given physical element may have to support multiple different policies for different services and/or user communities;
- the desire to take advantage of service specific knowledge within the network, for example shaping the constraints on the distributions, e.g. loss sensitivity, as a function of the nature of the information, e.g. video, data.

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

- 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], is concerned with securing 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.

This paper argues that 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 abouts its overall behaviour extremely difficult. This is likely to inhibit the introduction of new desirable policies.

This paper proposes a single abstraction within which different solutions at different protocol layers can be coordinated. For example, this abstraction allows the very different semantics of QoS present in ATM [4], Ethernet [5] and differentiated service [3] to be handled within a single policy.

The term *lightweight virtual network* is used to describe this abstraction. The *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. For reasons of brevity, lightweight virtual network, virtual network and VNs are all considered synonymous.

As an example, a service provider offering a corporate video on demand service for distributing internal company presentations could use the lightweight virtual network to: bound the amount of traffic generated by the service; to privilege certain presentations and/or certain users over others; to secure at the network level that only authorized users see sensitive material.

While the lightweight VN presented in this paper is a generalization of the VPN concept, in practice network operators of access networks are unlikely to be able to control the Internet Service Provider's core network. In this case, the traffic in the VNs is transported transparently between the enterprise's site across the core network (using say MPLS [6] tunnels) without the ISP being VN-aware.

This paper motivates the feasibility of the approach by describing the infrastructure necessary for creating such virtual networks. The constraints on the infrastructure 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 layer 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 of the network nodes. Equipment as diverse as a simple Ethernet hub and an MPLS Router must fit into the same framework. This is only achievable by allowing the equipment itself to make the decision as to how to support the VN.
- *be dynamic and self sustaining*: VNs are created dynamically without the intervention of a human operator, possibly across multiple administrative domains.

The infrastructure is a virtual network control architecture across which requests for the creation of virtual networks are carried. The VN control architecture was implemented in order to gain experience and as a proof of concept. It also serves as a useful platform for other network control experiments in the laboratory. Figure 1 shows an overview of the VN control layer and how it relates to the physical one.

The paper describes the following topics:

- the nature of the virtual networks within the control architecture;
- the way that diverse network technologies with different capabilities can be used to support the virtual networks;
- the means by which virtual network control messages are propagated;

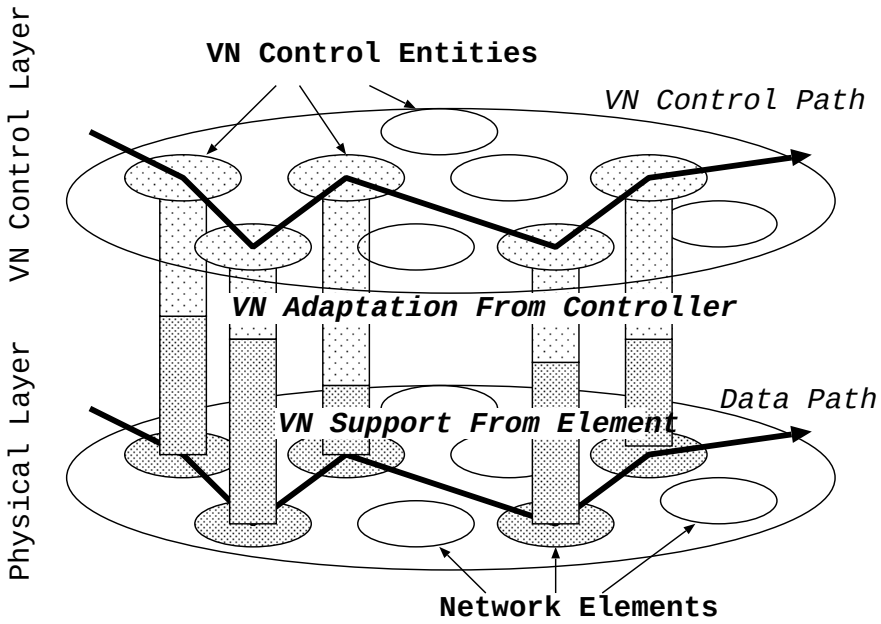

Figure 1: Overview

- experience with using the VN control architecture to support applications with network requirements.

Throughout the paper, the generality of the chosen solutions is discussed and the extensions and revisions needed to support other solutions and technologies examined. The paper attempts to distinguish between the general nature of the problem and the particular solution chosen for the implementation of this prototype.

While initially the focus has been on policy related to resource allocation, other policy related to charging, error recovery, security etc. must also be considered within a complete model and the reader should be aware that while often points are motivated using resource allocation as an example, the scope of this work is more general.

2 Nature of Virtual Networks

This section gives a general description of the nature of virtual networks. Then the choices made for supporting these networks are described and motivated. Finally, the consequences and generality of these choices are discussed.

2.1 Description

Within this paper a Lightweight Virtual Network (VN) is defined as: 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.

In consequence the policy to be applied to traffic on a given VN must be distributed to all appropriate nodes along with a label. Each data unit belonging to the traffic of that VN

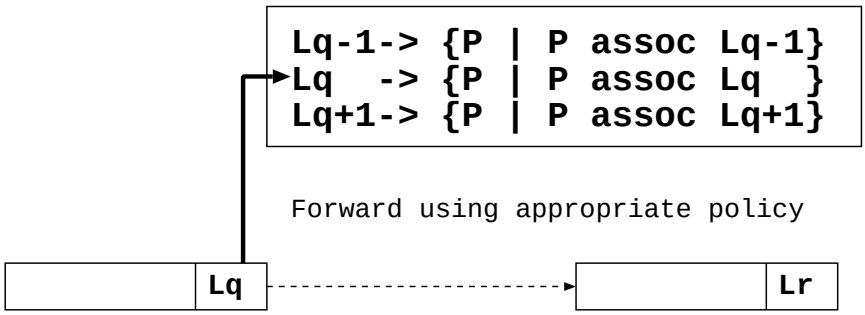

Figure 2: Label/policy association at network element

carries the label enabling the node to identify the appropriate policy to use when forwarding the unit.

Such a definition is general enough to encompass the diverse concepts of VPN defined in [7, 8, 9]. What distinguishes these various approaches are:

- the nature of the policy;
- the nature of the label;
- how and to which nodes the label/policy association is distributed.

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. [10] gives a conceptual network policy hierarchy; the use of policy in this paper corresponds to what [10] terms Node level and Device level policy. Figure 2 shows the label/policy association at a single network element.

No attempt is made to define precisely how policy is defined as this will necessarily evolve as network elements obtain new capabilities. However, network policy is an emergent property of the network elements forwarding function, for example while many diverse propositions for pricing have been made, at some level they all can be reduced to stop sending/forwarding packets after price/congestion has exceeded a given point. So any network policy must fit into the schema described by Figure 2.

Section 5 describes how the policy/label association is distributed. The rest of this section describes the choice of labels used for identifying VNs within the infrastructure.

A VN label can be something implicit, e.g. an IP source address, or explicit, e.g. an MPLS label[6]. 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 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 VN 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.

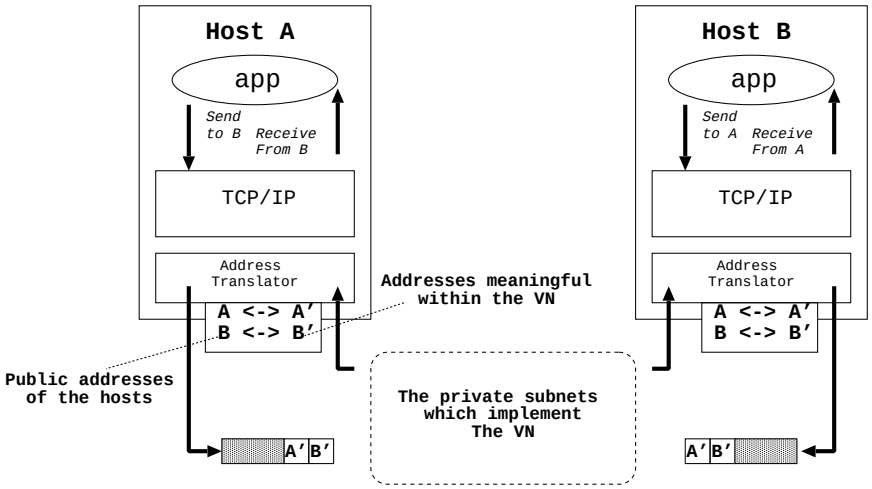

Figure 3: Public/Private Mapping

2.2 Design Choice

The chosen solution for the prototype was 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 VNs 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 VNs should be able to coexist on the same host and the presence of VNs on a given host should not affect its normal operation. In order to achieve this within the prototype, the VN 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 VN. 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 VNs.

In the prototype implementation a VN is identified by a `VNId` issued by authorization authority. While this identifier uniquely identifies the VN within the control path it is not carried by every packet in the data path. Instead it is resolved into an implicit label defined by the source and destination IP address and socket identifier. The resolution is explained in Section 3.

At the IP layer a VN appears as a set of private IP subnets. These subnets are created dynamically across the physical network and the participating end-points in the VN are informed of the private-to-public mapping they should use in order that their traffic be carried across that VN. 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 the addresses defined in [11] could be used, however as the VN exists only over a well defined set of sites, any address space convenient for those sites is usable. Different administrative domains negotiate at VN creation in order that the address space allocated to the VN during its lifetime is used by it alone across all the involved domains. How this is achieved is explained in Section 3. Note that the addresses used by a VN are available for reuse after it is deleted from the network.

The implementation requires that the private-to-public address mapping at hosts can be dynamically updated. Currently, this is achieved through modifying the state of the protocol stack. Enhancing the Dynamic Host Configuration Protocol (DHCP) [12], such that an appropriate private IP address was dynamically allocated to a host, would be an alternative, although it would restrict the host to only belonging to one VN at a time.

Figure 3 shows two hosts with addresses `A` and `B` wishing to participate in a VN. 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 The hosts update themselves such that if `A` wishes to communicate with `B` within the context of the VN then `A` should send with source address `A'` and destination address `B'`. The network nodes then identify the traffic within the VN using the private address and can then apply the VN specific policy when forwarding it.

Multi Protocol Label Switching (MPLS) [13] allows IP routers to forward IP packets quickly by using a label carried by the packet. The entire header of the packets does not need to be examined, only the fixed length label. To make use of MPLS's support for fast packet forwarding and traffic differentiation the VN labels must be resolved into MPLS labels on entering an MPLS cloud, for example in the current implementation the private IP address might be resolved into an MPLS label offering an equivalent class of service to that specified for the VN. The means by which MPLS distributes MPLS labels within the MPLS cloud remains unchanged.

If a wide variety of LAN and WAN equipment were MPLS aware, then it would make sense to replace the VN label with that of MPLS, i.e. the mapping between the two would then be the identity function. The label would serve a dual purpose; helping MPLS to perform fast traffic forwarding while allowing other (possibly non MPLS enabled) network nodes to identify the appropriate policy to apply to the associated virtual network.

Section 7 discusses some propositions for supporting VPNs using MPLS.

3 Virtual Network Admission Control

3.1 Description

This paper proposes lightweight virtual networks created on fast timescales without the need for human intervention (although the possibility of human intervention is not excluded either). Potentially even end-user applications that need some network policy, for example a video conferencing or distributed game, might initiate the creation of virtual networks. Some entity

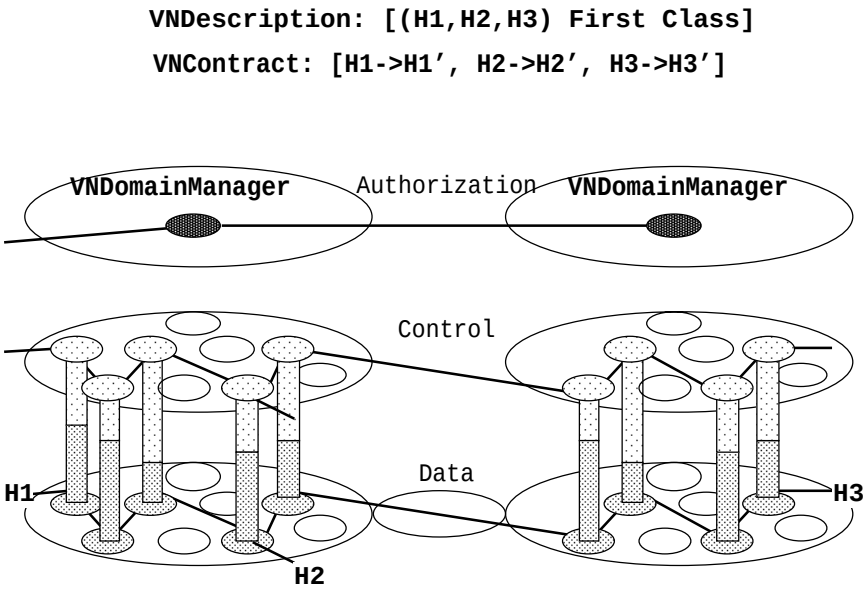

Figure 4: Authorization

must therefore authorize the creation before it takes place. The authorization entity decides both if the application is entitled to create the desired VN and whether the network is capable of supporting it.

In network technologies supporting the notion of end-to-end connections, e.g. ATM [4] the authorization function is termed Call Admission Control and is normally distributed across each network node across which the call is carried, i.e. each node makes a local decision to either accept the call and forward the message upstream or block the call and send some repudiation message down stream.

The authorization of end-to-end VNs using implicit labeling cannot be achieved in this way because system wide information must be obtained before the creation is initiated. In particular there must be agreement about the public→private mapping to use. While this might be seen as artifact of the implementation, in general the creation of VPNs requires coordination of the participating entities before creation can occur, for example keys must be exchanged if encryption is required, pricing negotiated and 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 networks nodes which will then make a local decision to accept and forward or reject and rollback.

3.2 Design Choice

In the current implementation, the authorization entity is termed the `VNDomainManager`, the abstraction the application passes to this entity is termed the `VNDescription` and the system wide state it returns the `VNContract`. `VNDomainManagers` are federated such that there is one per administrative domain and they intercommunicate in order to obtain the system wide information.

The VNDescription is simply a set of IP addresses of the end-points of the intended VN (in the case where the VN is supported from application to application, socket identifiers are included as well) and a policy description. As the focus has been on resource allocation, for the moment only one policy has been implemented: a class of service. This policy permits the resources to be allocated in such a way that certain VNs will be privileged over others in times of network congestion.

The VNDomainManagers collaborate together in order to resolve the VNDescription in to a VNContract 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 VN label formats, e.g MPLS labels, VLAN identifiers, used to create the end-to-end VN.

When a VNDomainManager receives a VNDecription, it determines which addresses are local to the domain and which foreign. For any foreign address the VNDomainManager contacts (via a directory service) the appropriate VNDomainManager. If no such entity exists then the VN creation can still continue but the policy can only be applied within the local administrative domain. The VNDomainManager must decide if this is appropriate. For example, in the current implementation the VNDomainManager will continue the VN creation but the public addresses for the foreign hosts will be mapped to themselves. Figure 4 shows the patterns of interactions between entities required for the creation of VNs across multiple domains involving three hosts with addresses H1, H2 and H3.

In the current implementation the communication between a client and its VNDomainManager uses a proprietary protocol. However, it would be possible to use the standard Common Open Policy Service Protocol (COPS) [14] for obtaining the VNContract. A host wishing to create a VN would start by emitting a COPS-Request message to the VNDomainManager (the location where policy for a given administrative domain is decided). This message would contain the required VNDescription in the client specific part of the message. The host plays the role of the COPS-PEP and the VNDomainManager the COPS-PDP. The VNDomainController would reply with a COPS-Decision message, containing in the client specific part the VNContract.

4 Virtual Network Adaptors

Section 2 explained why within the current infrastructure a simple implicit label based on the source and destination IP address was chosen. Section 5 explains how the label/policy association is disseminated. This section explains within the infrastructure how the policy is attached to the label for a set of diverse technologies.

4.1 Description

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 [5] enabled Ethernet switch priority switching based on frame tags, and an ATM switch continuous bit rate [15] 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.

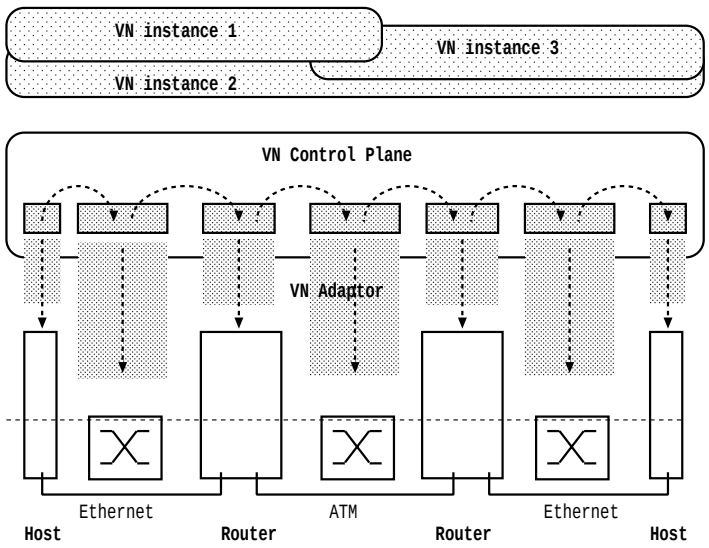

Figure 5: The VN Control Plane

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*.

It is only by adopting such a weak semantic that a wide variety of technology can be coordinated within the same framework. The VN control plane guarantees connectivity for the traffic of the VN, but gives no precise guarantee as to how the associated policies are enforced. The behavior of the VN control plane may be thought of as a type of 'best-effort policy support', it only gurantees not to make the situation worse for supporting the policy on its associated network elements, it does not gurantee to make it any better. In some cases the element will not be able to do anything at all, for example legacy IP routers do not support anything in the way of service differentiation — even in such cases however, it is still useful to know the policy that the node is expected to support, regardless of whether it is enforceable or not.

4.2 Design Choice

In the prototype a `VNController` entity is associated with each node within the physical network. The controllers support a common interface for VN control independent of the nature of the underlying element. Within each controller is an adaptor that maps generic VN 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 `createVN` operation taking a `VNContract` as argument, their effect is very different: the router might update its routing tables to reach the newly created private IP subnets with an appropriate priority, while the Ethernet switch might support the VN through the creation of a IP rule based VLAN with associated priority.

Figure 5 shows schematically the relationship between the VN Control plane, the control adaptation layer and the physical network.

The adaptor can add state to the `VNContract` which may or may not be meaningful to the next hop. For example, on an ATM switch the description of the created virtual connection

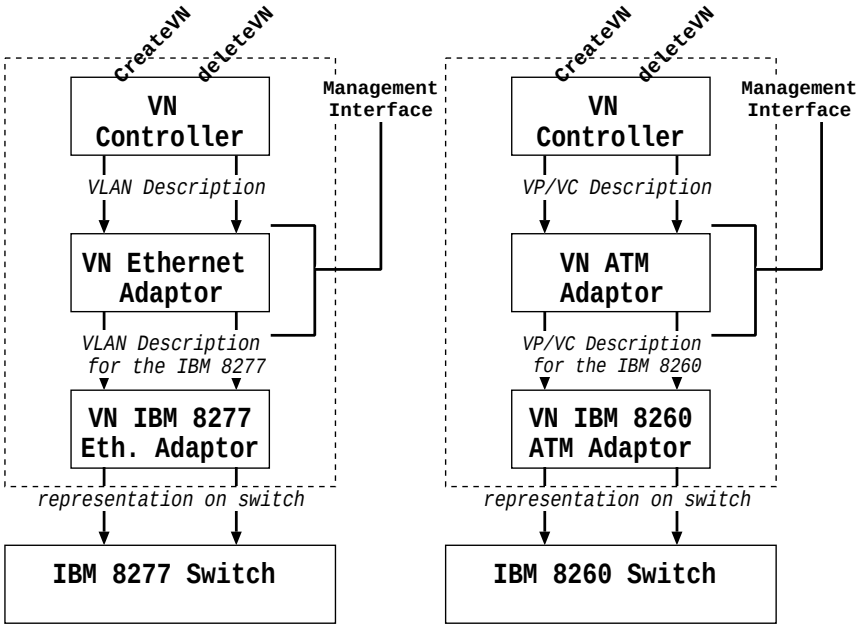

Figure 6: Example of adaptors

is added so that the next hop can make use of it to create the entire virtual circuit.

The `VNControllers` run either as integral part of the entity if the entity supports enhancements to its control plane or separately if its doesn't. For example, the `VNController` is integrated into the Linux operating system (as a loadable Linux module) but runs as an application on Windows NT.

Figure 6 shows an example of two adaptors for two different technologies and pieces of equipment.

At first sight, the addition of 'yet another' piece of software to network elements — the `VNController` — might seem to complicate further the already complex control plane. In reality, the `VNController` is just a convenient abstraction to collect together a variety of architectural entities currently supported by network elements. For example, RSVP [16], GVRP [5], MPOA [17], COPS [14] the MPLS LSR [13] etc., would all be constitutions of the controller. The controller supplies an interface for VN 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 `VNController` interface and a network element with a given set of capabilities. The exact nature of the mapping can be set and adjusted by the network operator using a management interface, e.g. SNMP. For example, a new class enterprise specific policy ('The-quarterly-report-broadcast-policy') might be added to the `VNContract` and the set of adaptors altered to give it a meaning.

5 Propagating VN control messages

5.1 Description

To create a virtual network, the entire set of specified hosts 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 VN 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 [16] is a signaling protocol designed for the creation of IP flows. Adopting RSVP as the means of VN creation has the advantage of using a well known and established protocol. RSVP soft state model allow the VNs 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 [16], does not match exactly the needs of the VN control plane. The problems are as follows:

- while RSVP uses IP routing in order to determine the next hop of the control message, within the VN control plane layer 2 devices must also receive the RSVP messages;
- 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 VN described in this paper should satisfy a larger class of applications.
- 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 VN is (by definition) a graph.

It is legitimate to question whether it is better to retrofit an existing protocol to the needs of the infrastructure or reinvent an entirely new protocol for the policy propagation. However, the overriding principle is to knit together existing capabilities rather than invent new ones. It therefore makes more sense to propose a small delta on RSVP, rather than an entirely new proprietary protocol.

5.2 Design Choice

In the prototype an enhanced RSVP daemon is run as part of the `VNController`. 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 host. 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.

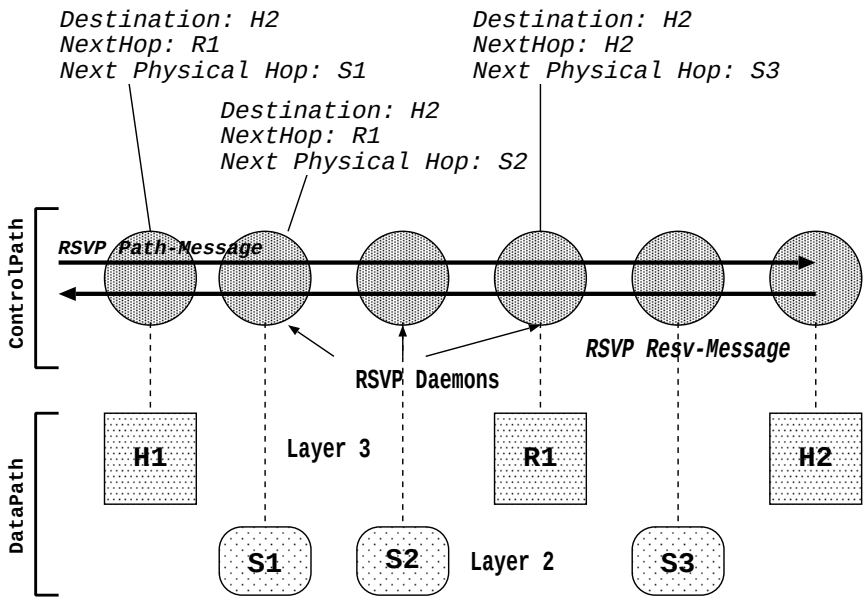

Figure 7: VN Creation using RSVP Messages

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 `VNController` 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 (although a working group [18] is trying to define a generic framework). For implementation purposes a simple Ethernet discovery system has been created based on each Ethernet VNController registering its port/MAC association with a `VNDomainManager` and that entity correlating 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 VN 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 `VNController` 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 VN creation. In the current implementation it is the host on which the applications that requests the creation that is considered the initiator, but another host might be chosen (perhaps based on its location in regard to the other end-points).

The initiating host starts by emitting RSVP Path-Messages, passing in the message the `VNContract`. Each `VNController` that receives a messages, examines the contract to determine if it can support the contract and if so reserves the required resources. If it is the `VNController` of a network node it then forwards the message to the next `VNController` upstream, if on the other hand it is the `VNController` 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.

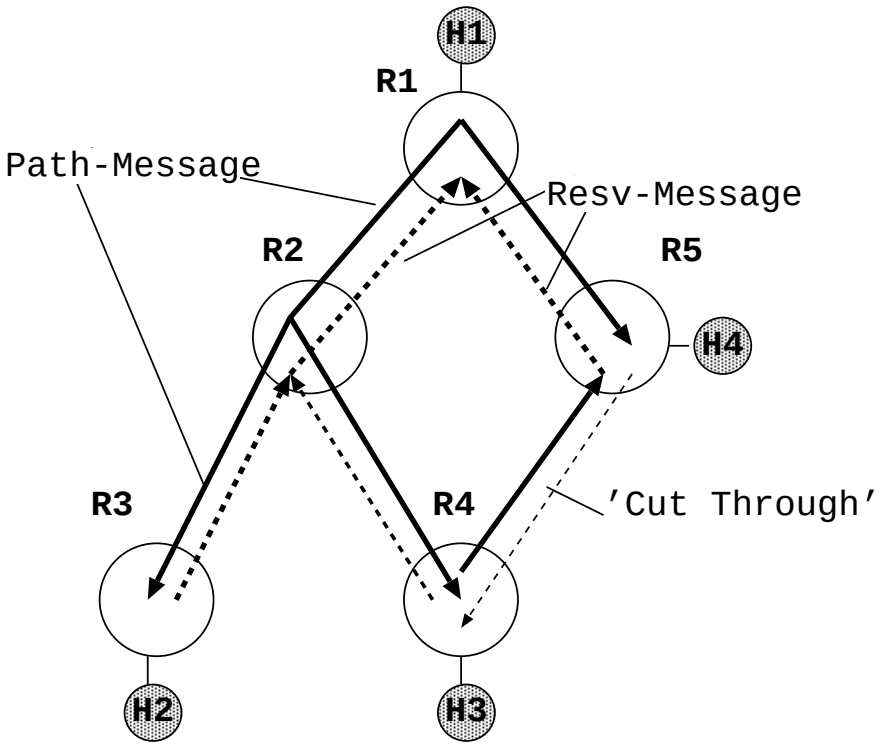

Figure 8: Optimizing the virtual network topology

A `VNController` 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, updates the routing table on the router.

Figure 7 shows the pattern of communication for creating a VN 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 VN is created with the initiator as the root of the VN and the others hosts 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 VN tree. The `Path-Message` that a host receives contains the `VNContract`, therefore the receiving host has access to the complete set of other hosts participating in the virtual network. A host can make a local decision if there is a better route between itself and one or more of the other non-senders nodes than simply going through the sender and can start sending out `Path-Messages` for the same contract. The `VNController` will not forward `VNContracts` over interfaces to which they are already supported; normal routing will ensure that the better route to the other hosts are included as part of the virtual network.

Figure 8 shows a VN created using RSVP between four hosts `H1`, `H2`, `H3` and `H4`. In the initial phase, `H3` and `H4` have `R1` as nearest common ancestor. However, `H3` recognizes that there is a better route to `H4` and performs a 'cut through'. 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 VN 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 VN is modified accordingly. This permits the VNs to

dynamically adapt to the current state of the network.

6 Experience with Applications

This section motivates the description of the infrastructure given in previous sections, by explaining how lightweight virtual networks are created in the test network to support a video on demand service.

The topology of the test network is arranged such that there are two Ethernet island interconnected via ATM. Each Ethernet islands contains a small number of 100 MB/s Ethernet switches (e.g. IBM8277, IBM8371) and are connected to the ATM network via AIX workstations acting as routers. Attached to the Ethernet switches are a collection of hosts running Linux or Windows NT.

Over this physical network runs the infrastructure required to support multiple video distribution networks. The Real-Network family of products is used as the video producers and consumers.

The virtual networks have different policy in regard to the service quality they are allocated. A video feed carried over a virtual network with a low priority suffers more when congestion is introduced into the network, than one with a higher priority. The assumption is that in a real system the priority would be some function of a cost users are prepared to pay.

While at the moment there are no real users of the video on demand service (although the intention is to turn it into a real service for distributing videos of internal presentations) others use the network both for experimental and production purposes. The virtual networks described in this paper are dynamically created across the physical network without disturbing these other users.

In order to test the infrastructure, a management application creates the virtual network as well as observes the current networks in place. Creation involves selecting a group of host addresses and a service level. This is used to create a `VNDescription` and thereby a `VNContract` as explained in Section 3. Figure 9 shows the management view of the network after two virtual networks have been created from the same server to two different hosts.

The contract is then passed to one of the hosts (the first one mentioned in the description) using the extended RSVP, the pattern of communication between the end-points is then as described in Section 5.

The adaptors described in Section 4 for the different elements do the following:

- on the IBM8271's and IBM8371 Ethernet switches the controller examines the contract to determine if the public IP subnet it supports is mentioned. If so a new IP based VLAN is created for the new corresponding private IP subnet. For switches that support IEEE 802.1q the VLAN is given an appropriate priority.

- on the AIX 4.3 router the routing tables are updated such that the routers know how to forward packets on the new private IP subnets. The routes to the new private subnets must be via the same physical interfaces as the corresponding public ones as they are simply an overlay. A set of PVCs each with a different traffic class interconnect the two AIX boxes across the ATM network. The router chooses an appropriate PVC appropriate for the demanded service level, in order to forward traffic to its homologue. No controller is run for our ATM switches as this is part of the production network and experimental systems are not allowed to change their state.

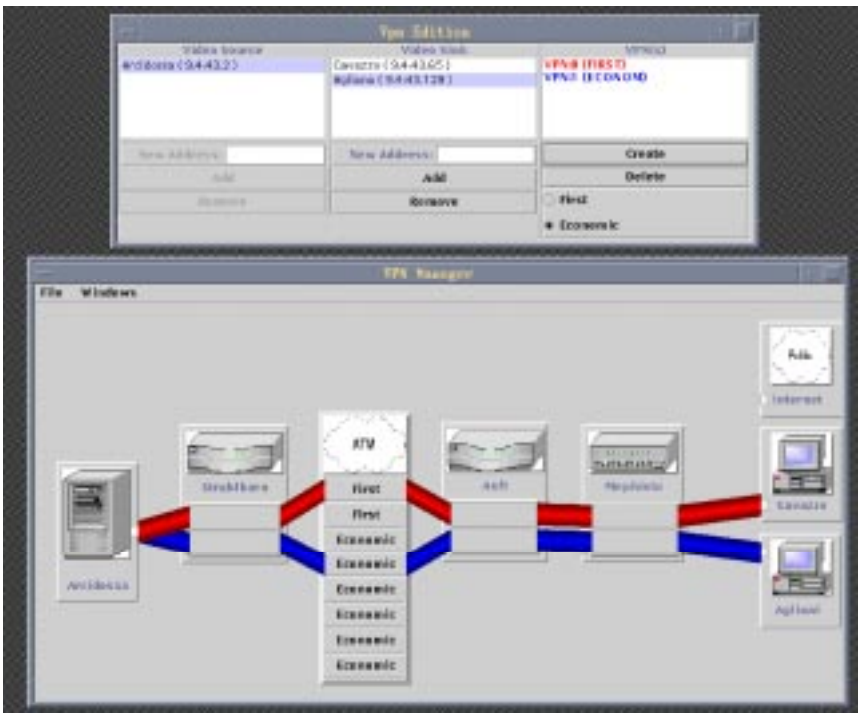

Figure 9: Screen dump of management application for the prototype

- on the Linux boxes the modified kernel updates its protocol stack using the public → private mapping carried in the contract. When Windows is used the addresses must be predefined, as Windows does not offer the same flexibility in regard to dynamic address modification.

7 Related Work

Diff-serv [3] defines a architecture which 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 Type Of Service (TOS) field in the IPv4 header. For example, all video traffic might be marked with the appropriate bit pattern allowing it to be privileged over data traffic. Diff-serv is normally only enabled at edge routers. The policy/label association is typically manually set by a network operator on the router directly 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 that just 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 VN control plane can make use of diff-serv if enabled at the routers within its domain. For example, within the current implementation the `VNDomainManager` could select an appropriate TOS pattern for the VN and this would then be propagated to all participants in the `VNContract`. Dynamic updating of the significance of TOS patterns could be achieved by adding the routers policy directory to the conceptual `VNController` entity of the router.

[7] defines a VPN as a dedicated IP service offered to an enterprise in order to allow address isolation, data privacy and intra VPN routing. It 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 support a virtual router for each VPN they support which services a separate forwarding table. The VBR forwards traffic from an enterprise across appropriate MPLS tunnels to a remote VBR at the appropriate site. VBR 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 [7] is complementary to the work described in this paper. Traffic from multiple VNs could be aggregated over a single MPLS VPN, or a dedicated VPN could be assigned to a VN. The VN-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 either directly or by using a label (perhaps the 16 bit VPN-ID dedicated label proposed by [7]) telling it how to tunnel to a router within the enterprise that can.

[19] proposes another virtual router based approach using MPLS. Virtual routers discover homologues using ARP over an emulated LAN supported by IP multicast. [20] is another variant on the same theme.

[21] describes an infrastructure in which separate control mechanism can be associated with different VPNs over the same physical infrastructure. The policy associated with the VPN goes beyond simply the forwarding function, for example policy can involve the algorithms used for routing, call admission control and the protocols used for exchanging information between network elements; it terms these collectively a control architecture. While [21] has a strong ATM bias, it actually only requires that the underlying technology can be partitioned. The work described in this paper could be considered as just another control architecture supportable within [21], however [21] is very weak on how the infrastructure knows how and where to build the VPNs, especially across administrative domains. The work described here is simple and makes minimal assumptions about the capacity of the network elements, in consequence it might be described as a candidate for bootstrap control architecture for [21].

[22] describes what the author calls Policy Based Network Management. In essence this is the enhancement of network management platforms such that they can make use of the support for service differentiation of the equipment. It describes the current offers from a set of vendors and suggests that the trend is likely to continue. The fact that vendors are taking advantage of the lack of integration of different technologies in order to promote management solutions demonstrates that there is a problem; the work described in this paper proposes a dynamic self managing control plane rather than a management solution requiring human intervention.

8 Conclusion

This paper has argued that the diverse support for policy across different network technologies and in different scopes makes implementing the policy end-to-end problematic. It has proposed a unifying abstraction — a lightweight virtual network — with which a set of different policies may be associated across a range of technologies. It terms such networks lightweight as they 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 of any support, for example diff-serv, P-NNI, offered by the underlying devices without requiring their presence. This is possible by allowing the network elements themselves to decide how best to support a policy. This paper has motivated the feasibility of the approach by describing an infrastructure which supports the creation and management of such lightweight virtual networks.

9 Acknowledgements

The author is indebted to Christian Hörtnagl and Anthony Bussani for their work in implementing parts of the infrastructure described here.

– –

References

[1] Cisco, "Virtual Private Networks(VPN)," *Cisco System Inc. marketing information*, 1999.

[2] S. S. R. Braden, D. Clark, " Integrated Services in the Internet Architecture: an Overview," *Internet Draft 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] ATMF, "B-ISDN Inter Carrier Interface Specification - Version 2.0 (B-ICI 2.0)," *The ATM-Forum: Approved Technical Specification*, December 1995. af-bici-0013.003.

[5] 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.

[6] E. Rosen, A. Viswanathan, and R. Callon, "A Proposed Architecture for MPLS," *draft-ietf-mpls-arch-00.txt*, August 1997.

[7] L. Casey, I. Cunningham, and R. Eros, "A Framework for IP Based Virtual Private Networks." draft-casey-mpls-vp-00.txt, November 1998. Work in progress.

[8] N. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. Ramakrishnan, and J. van der Merwe, "A Flexible Model for Resource management in Virtual Private Networks," *ACM Computer Communications Review*, vol. 29, pp. 95–108, October 1999. Proceedings SIGCOMM'99.

[9] S. Fotedar, M. Gerla, P. Crocetti, and L. Fratta, "ATM Virtual Private Networks," *Communications of the ACM*, vol. 38, pp. 101–109, Feb 1995.

[10] R. Rajan, D. Verma, S. Kamat, E. Felstaine, and S. Herzog, "A Policy Framework for Integrated and Diffentiated Services in the Internet," *IEEE Network*, pp. 36–41, September/October 1999.

[11] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. de Groot, and E. Lear, "Address Allocation for Private Internets," *RFC 1918*, February 1996.

[12] IETF, "Dynamic Host Configuraion Protocol Options and BOOTP Vendor Extensions," *Internet RFC 1533*, October 1993.

[13] R. Callon, P. Doolan, N. Feldman, A. Fredette, G. Swallow, and V. Viswanathan, "A Framework for Multiprotocol Label Switching," *Internet Draft*, November 1997.

[14] J. Boyle, "The COPS Common Open Policy Service Protocol," *Internet Draft Work in Progress*, February 1999.

[15] S. S. Sathaye, "ATM Forum Traffic Management Specification Version 4.0," in *ATM Forum Technical Committee - Contribution 95-0013*, 1995.

[16] 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.

[17] ATMF, "Multi-Protocol Over ATM (Version 1.0)," *The ATM Forum Technical Committee, af-mpoa-0087.000*, July 1997.

[18] K. S. Jones, "Physical Topology (PTOPO) Working Group," *Internet Draft*, March 1997.

[19] K. Muthukrishnan and A. Malis, "Core IP VPN Architecture." draft-muthukrishnan-corevpn-arch-00.txt, October 1998. Work in progress.

[20] D. Jamieson, B. Jamoussi, G. Wright, and P. Beaubien, "MPLS VPN Architecture." draft-jamieson-mpls-vpn-00.txt, August 1998. Work in progress.

[21] J. van der Merwe, S. Rooney, I. Leslie, and S. Crosby, "The Tempest - A Practical Framework for Network Programmability," *IEEE Networks*, pp. 2–10, May/June 1998.

[22] J. Conover, "Policy Based Network Management," *Network Computing*, November 1999.