# On Managing Traffic over Virtual Private Network Links

Leila Lamti Ben-Yacoub
ENST Bretagne
2 Rue de la châtaigneraie, BP 78 35512 Cesson Sévigné - France
Leila.Lamti@enst-bretagne.fr

*Abstract*

*A Virtual Private Network (VPN) is a solution that provides corporate networking between geographically dispersed compagny sites. VPN sites consist of Local Area Networks (LANs) and their interconnection is based on a public network infrastructure. Links connecting VPN sites are allocated with a peak rate and the VPN customer pays for the reserved bandwidth. Our approach assumes that reservation protocols for the establishment of guaranteed links throughout a public network exist and are functional. The goal of this work is to develop a control model able to manage traffic on these links. By maximizing bandwidth utilization, an optimum balance for the allocation of bandwidth on each virtual link can be found. To respond to such needs, we propose to introduce a control architecture that we call* The Hierarchical Bandwidth Manager. *into each LAN of the VPN. This manager is able to cope with best-effort and guaranteed flows so that bandwidth left unused by guaranteed flows is dynamically distributed among best-effort ones. It uses a tree representation of the LAN nodes and regulation of link bandwidth is done in a hierarchical and distributed manner. Each node in the tree respects either an* inter-node bandwidth share *protocol or an* intra-host regulation *protocol. The bandwidth manager relies on the implementation of both protocols in the tree nodes. The Intra-host regulation protocol is implemented through a new mechanism that we call* fair shaper. *An implementation of the* fair shaper *in a Solaris 2.5 operating system and results of the implementation of the inter-node bandwidth share protocol in the* ns *network simulator are also described in this paper.*

*Keywords: VPN, LAN, Traffic shaping, Packet scheduling*

## 1 Introduction

Efficient link sharing among multiple sessions with QoS has been analyzed in several research studies [6], [17]. Heterogeneity is one of the main problems since multiple traffic classes with different QoS requirements are multiplexed on the same path. Integration of applications with different QoS needs and traffic characterizations in a local network in general and at the network outlet in particular is an important problem to solve. It is even more critical in the presence of known or predicted resource contentions. With the advent of constrained traffic (real-time, jitter or delay constrained, ... ), the interest in the specification and development of new mechanisms able to cope with such traffic characterizations has increased.

Major networking communities have contributed in the field of QoS provisionning, especially with the specification of Traffic Management for ATM networks [18]. These specifications allow flexible bandwidth allocations at high speeds and a scalable architecture, in the context of local area networks (LANs). The IETF community has also contributed in this field by designing and developing a new architecture for integrated services in the Internet.

Both communities have identified the main categories of services to be concerned with the integration and have defined different service classes. These service classes are designed to provide certain QoS guarantees for traffic flows. We can distinguish at least two types of services: A guaranteed service and a Best-Effort service. Examples of the guaranteed service are the CBR and VBR transfer capabilities of the ATM network. On the other hand, we find two categories of commitments defined as guaranteed services by the IETF: the guaranteed service class and the controlled load service class. The guaranteed service class provides the users with an assured amount of bandwidth, firm end-to-end delay bounds, and no queuing loss for flows that conform to the parameters negotiated at the connection set up. The controlled load service class does not provide the users with any firm quantitative guarantees. It simply ensures that the users will get a service as close as possible to the one received by a best-effort flow in a lightly loaded network. Best-Effort services are those that we currently experience on the Internet backbone. Applications are expected to use one of these classes according to their needs.

The same issue for QoS provisioning in the context of Virtual Private Networks (VPNs) has emerged. A VPN is a concept that allows a customer to interconnect geographically dispersed sites throughout a public network infrastructure. It is likely to be used by customers instead of networks constructed using private lines. Site interconnection may be of interest to a large number of international companies. It provides a cost-effective response to two major requirements in current corporate networks.

- The evolution of information processing systems deployed in LANs and the need to interconnect such systems has increased the need for reliable and high performance links.
- A VPN network, if well designed, offers an enterprise a reliable and easy to manage communication infrastructure.

Due to a lack of standardization, VPNs were more marketable products than well-defined telecommunication services. Currently, more attention is being paid to the VPN concept and several research projects as well as international standardization activities are involved in this field. We should note that in the VPN context, efficient link sharing becomes a clear and important need.

Within the IETF community, a significant amount of work is being done for the specification of the VPN service deployment over the IP backbone [10] for the elaboration of a QoS framework for IP-based VPNs [4]. ATM backbones may also support VPN service deployment and in the literature, several papers propose the use of ATM Virtual Paths for the provision of VPN services [9], [23].

Supporting different classes of traffic within a VPN allows differentiation between traffic such as data versus voice or video flows, or any delay and loss sensitive streams. There are multiple ways in which VPN QoS may be managed in the network. In our study, we focus on a solution where the resources (VPN links capacities) are managed locally at the edges. Different QoS can be provided by appropriately scheduling access to the VPN link at the customer edge. Another solution could be based on core scheduling, i.e. the provider network is responsible for the scheduling decisions within the core network to share the VPN resources between flows.

We think that managing resources locally at the customer edge allows high multiplexing gains with minimum customer-network interactions. Also, such a solution is easier for the provider (the provider only guarantees a maximum capacity on a link and it is the responsibility of the customer to efficiently control link sharing). It offers a minimum implementation cost from the provider point of view. Even if this solution requires the customer to support a control model able to manage traffic over VPN links, it is more realistic than solutions that assume the support by the provider equipments of new VPN-related control functionalities.

A VPN may be built over an ATM network or over the Internet backbone. In both IP-based and ATM-based implementations, VPN nodes are connected to each other through virtual links. These virtual links are usually allocated with a peak rate and the VPN customer pays for the reserved bandwidth even when he does not use it. In this paper, we focus on virtual link bandwidth allocation. We assume that within a LAN, bandwidth is less scarce than at interconnection points and focus our study on probable network bottlenecks that appear in equipment interconnecting heterogeneous bandwidth networks that are also called *edge routers*.

Since traffic characterization is a difficult process, customers tend to over-allocate bandwidth on virtual links in order to prevent congestion situations which leads to an inefficient network utilization. To avoid such inefficiency, a reasonable solution would be to try to enhance statistical multiplexing on each virtual link. In order to obtain differentiated and customized Quality of Service (QoS) per flow, an adequate and sufficient amount of bandwidth should be allocated. Also, the statistical multiplexing should be coupled to an efficient control of the bandwidth distribution between flows sharing the same virtual link. Traffic has to be managed so that each flow at least receives the amount of bandwidth it has been agreed to use.

The work we present here proposes a distributed link sharing algorithm. Our proposal relies on two important principles: flow isolation and traffic shaping. Flow isolation is necessary since sensitive applications with stringent QoS requirements could malfunction due to a lack of bandwidth or network congestion (caused by an unfairness in bandwidth share or by greedy applications). On the other hand, traffic shaping, when performed, ensures that the overall multiplex of flows sharing the same virtual link does not exceed the maximum allocated capacity. Without traffic shaping, non conforming packets may be sent and discarded by the policing functions of the public network.

To meet these requirements, we propose to introduce in each LAN of the VPN a distributed control model that we call The Hierarchical Bandwidth Manager. It uses a tree representation of the LAN nodes and the regulation of link bandwidth is done in a hierarchical and distributed manner. Control algorithms are performed by each node in the hierarchy in order to implement the 2 above-mentioned features. Each node in the tree conforms to either an inter-node bandwidth share protocol or an intra-host regulation [14].

The inter-node bandwidth share protocol is a dynamic hierarchical bandwidth allocation scheme. The top level of the hierarchy is the VPN link edge device; the next level consists of subnet routers and the last level consists of individual hosts. Starting from an initial set of bandwidth shares, each parent monitors the bandwidth utilization of each child, and periodically reallocates the excess bandwidth among the children hosts that are not under-using their shares. The intra-host regulation protocol uses a rate regulator for guaranteed traffic and best-effort traffic is kept in a separate queue. Whenever no guaranteed traffic is eligible to be sent, best-effort traffic, if any, is served.

The paper is organized as follows: Section 2 introduces the concept of virtual sub-networking and describes some alternatives associated with this concept. In section 3, we explain how the proposed manager may be combined with VPN implementation and present its principle. In section 4, we introduce two new protocols called the inter-node bandwidth share protocol and the intra-host regulation protocol that constitute the control components of the hierarchical bandwidth manager. We explain how these protocols dynamically enforce bandwidth regulation on LAN virtual links. We also present a novel algorithm called the fair shaper used to schedule packets in hosts and aimed at implementing the intra-host regulation protocol. In Section 5, we explain how the inter-node bandwidth share protocol can be implemented in existing equipment. We also present some implementation details of the fair shaper in a multi-tasks environment. The results of implementation of both protocols in the ns network simulator [22] show that bandwidth regulation goals are reached. We show how it offers a reasonable bandwidth utilization in an end-system. Section 8 resumes and concludes this paper.

## 2 Virtual Private Networks: State of the art

Each enterprise site represents a LAN. The purpose of a VPN service is to interconnect these LANs through a public network infrastructure. As mentioned above, a VPN service may either be offered over the IP backbone or through a public ATM network. In the next paragraph, we present the related connectivity models proposed in the literature.

### 2.1 ATM-based implementations

A VPN can be built using native mappings onto the ATM switched backbone, e.g. VPNs using the LAN emulation over ATM (LANE) [7], or Multi-protocol over ATM (MPOA) [8] protocols. The key factor is that ATM technology facilitates the VPN service based on the Virtual Path (VP) concept. A VP is a virtual connection between 2 end-points and it supports multiplexing several Virtual Channels (VC) within the same VP.

To construct a VPN based on an ATM backbone, it is possible to establish end-to-end VPs between any two sites in the enterprise network. Another solution is to concatenate two or more VPs between VPN sites and ATM nodes [9]. In the literature, we can distinguish two main types of ATM-based VPN architectures: the End-to-end VPN (EEVPN) and the Broadband VPN (BVPN). In the following we briefly present each of them.

- The EEVPN solution: All customer sites are interconnected with a fully meshed VP network. The customer is the only responsible for multiplexing multiple sessions (VCs) within a VP. The network operator polices the peak rate allocated to the VP. This strategy has the advantage of allowing the user to manage its VPs internally without interacting with the network operator. The main drawback is bandwidth waste over many VPs and lack of statistical multiplexing between different source-destination pairs. The main advantage of the EEVPN strategy is that no extra function is needed in the public network. A typical EEVPN architecture is represented in Figure 1.

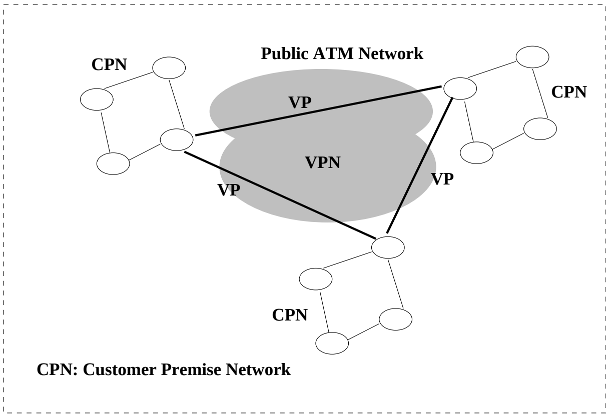


Figure 1: An example of VPN architecture

- Broadband Virtual Private Networks: The BVPN is a strategy that has been proposed in [23] by Walters, et al. This strategy consists in connecting ATM nodes to each other and to user nodes through a mesh of VPs (c.f. Figure 2).

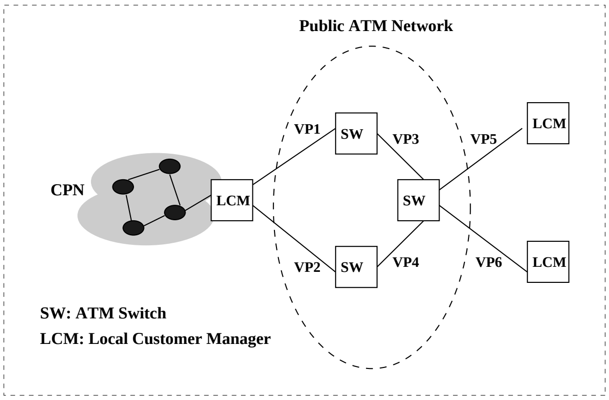


Figure 2: The BVPN strategy

According to this strategy, a logical private customer network is embedded in the public ATM network. This private network is composed of Local Customer Managers (LCM) that connect the user site to the nearest ATM node. Like in the EEVPN strategy, a mesh of VPs is allocated to the user with peak bandwidth. The difference is that the VP network is not a fully meshed one and that a VC is not confined to a single end-to-end VP, but may cross a set of VPs. This architecture relies on the ability of ATM nodes to keep information on VP/VC mapping for the routing function. The main advantage of this strategy is obviously the gain in statistical multiplexing between VPs with different source-destination pairs. However, a main drawback is that the network should support the VC cross-connect function and also, policing should be performed at each VP segment (not only at the network entrance). Another problem caused by this strategy is internal congestion in ATM nodes. In fact, since the VC traffic pattern may change, some output VPs may be overloaded when the number of supported VCs increases. To illustrate this situation, suppose that traffic from VP5 and VP6 are initially balanced (see figure 2). Because of traffic fluctuations, it is possible that during a given period, all traffic from VP5 and VP6 goes out to VP4. This will cause internal congestion of the switch on output VP4. The contention can be resolved in 2 ways:

1. Fast Reservation Management (FRM)

   This approach is based on a reservation procedure for traffic acceptance and bandwidth allocation. When a burst arrives at the LCM, this latter generates a reservation cell and sends it to the nearest ATM switch on the proper VC. If sufficient resources are available on the path from the source to the destination, a booking procedure is invoked and the burst is accepted. Otherwise, it is rejected. The main advantage of this strategy is that it polices VCs individually and supports dynamic bandwidth allocation. Nevertheless, it has high round-trip time and processing overhead. It also supposes that the public ATM network supports extra functions to deal with burst reservation cells. More details on this strategy can be found in [21].

2. Intranode Policing (INP)

   In this policy, the ATM switch keeps a common counter for all VCs routed from one VP to another. Counters are chosen so that total traffic from all VCs routed to an output VP is always less than the VP maximum capacity. It is as if an Intranode VP (I-VP) is created within the ATM switch and the policing function is based on these I-VP parameters. The drawback of static I-VP is poor performances when traffic pattern changes frequently. This may be corrected by allowing users to change I-VP parameters but such a solution requires the existence of a signaling mechanism.

For both FRM and INP, the routing function must be performed at network initialization. The bandwidth allocation function concerns only VPs. In INP, the policing parameters are chosen to limit the peak bandwidth on I-VP while for FRM, VC peak bandwidth is chosen at each set up of a connection. Finally, INP and FRM strategies, unlike EEVPN, need a fairly high important involvement on the part of the ATM network. In addition to the signaling mechanism that should be supported, ATM nodes should introduce new hardware and software to deal with INP and FRM requirements.

### 2.2 IP-based VPNs

The IETF community is now interested in the implementation of some VPN solutions over the IP backbone. An IP-based VPN framework [10] mainly focuses on 2 main possible VPN implementations:

- Virtual Leased Lines (VLL): The two endpoints of the VPN are connected by an IP tunnel which emulates a dedicated connection or a physical leased line. This model can use for example the IP-in-IP tunneling protocol (defined in [19]) (other examples can be found in [10]). The IP backbone is used as a link layer technology, i.e., the traffic sent trough the tunnel is opaque to the underlying IP backbone. This solution has the same advantages and the same drawbacks as the EEVPN solution since both of them rely on a simple implementation. Figure 3 shows an example of a VLL implementation.

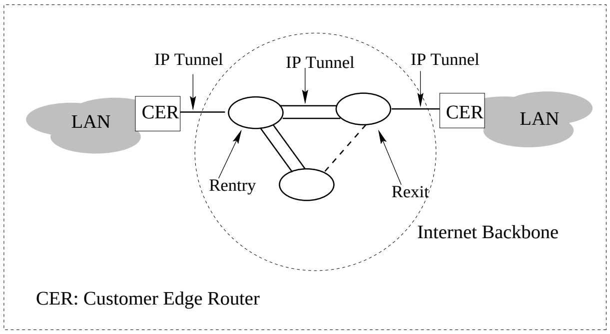


Figure 3: A VLL example

- Virtual Private Routed Network (VPRN): In a VPRN, each customer edge router connects to one or more Internet Service Provider (ISP) edge routers. A VPRN consists of a mesh of IP tunnels between ISP edge routers. Attached to these routers are customer edge routers connected via one or more **stub** links. The customer edge router sends all traffic for all non local VPRN destinations to the ISP edge router. The burden of tunnel establishment and management is out-sourced to the ISP edge routers. These routers should support appropriate forwarding mechanisms for ongoing and outgoing traffic. Each stub link (at the customer edge router) should be configured with the identity of the particular VPRN to which it belongs. The customer edge router needs to learn about destinations reachable via each stub link. It must know the set of non-local VPRN destinations reachable via the default VPRN route. Each ISP edge router should disseminate access information to all ISP edge routers that have members in the VPRN. A central database could be used to maintain a listing of the address prefixes associated with each point. The main advantage of a network based VPRN is the simple configuration of the customer edge router. However, such a solution requires ISP routers to deal with VPN routing information exchange and to implement appropriate forwarding functions. It also relies on a new VPN identifier field to be added in IP headers that will be used by the routing functions. Figure 4 describes an example of a VPRN implementation.

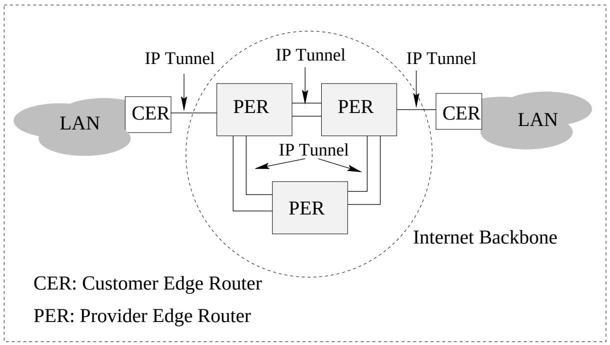


Figure 4: A VPRN example

## 3 The Traffic Management Requirements

It follows from the above-mentioned VPN connectivity models that, from a traffic management point of view, we can represent VPN interconnection points by means of a unique model. In order to make a clear presentation of this model, we first give some definitions that will be used in the rest of this paper:

### 3.1 Definitions

- Customer Edge Device: This is the edge device on the customer side that connects the LAN to the provider network. It consists of the IP router that connects the LAN to the Internet backbone for the VLL or the VPRN solutions. For the EEVPN and BVPN solutions, it consists of the ATM switch that connects the LAN to the public ATM network.

- Provider Edge Device: This is the edge device at the provider side to which the customer edge device is connected. Like for the customer edge device, it may be an IP router or an ATM switch.

- Virtual link: Depending on the VPN implementation, it may consist of the virtual link that connects 2 customer edge devices in the VPN or the one that connects a customer edge device to the provider edge device. For example, in an EEVPN solution, a virtual link represents the VP that connects two customer edge devices of two LANs within the VPN; in a VPRN solution, it represents the IP tunnel that connects the customer edge device to the provider edge device.

Figure 5 presents the generic VPN connectivity model that we will use for the rest of the paper and resumes all the above-mentioned definitions.

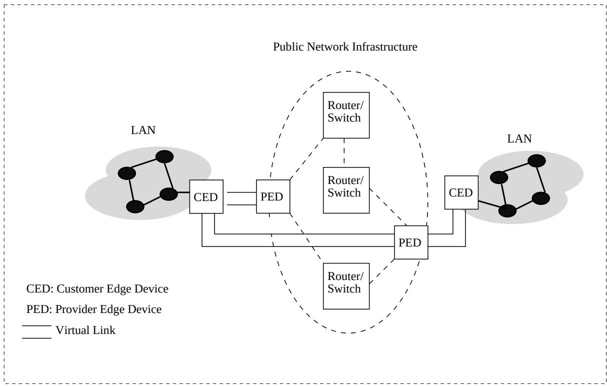


Figure 5: The generic VPN connectivity model

### 3.2 Requirements

In order to obtain differentiated and customized Quality of Service (QoS) for flows belonging to a VPN, an adequate and sufficient amount of bandwidth should be allocated on each virtual link over which flows are multiplexed. Since bandwidth over-allocation on virtual links leads to an inefficient utilization of network resources, a reasonable solution would be to try to enhance statistical multiplexing over each virtual link. The high peak-to-mean ratios in the volume of LAN outgoing traffic allow high multiplexing gains.

The statistical multiplexing should be coupled to an efficient control of the bandwidth distribution between flows sharing the same virtual link. Traffic should be managed so that each flow receives the amount of bandwidth it has been agreed to use. Our approach assumes that reservation protocols for the establishment of virtual links with a fixed capacity throughout a public network exist and are functional. Bandwidth allocation over each virtual link requires the VPN customer to determine the traffic distribution between the VPN sites. Such knowledge is often unknown, especially for new customers. Traffic characterization over virtual links is not studied in this work but is subject of ongoing research studies [16].

The sharing model per link we propose is called a **hierarchical bandwidth manager** and is able to control the access to an outgoing VPN link. The hierarchical bandwidth manager relies on two protocols: an **inter-node bandwidth share protocol**, and an **intra-host regulation protocol.**

The inter-node bandwidth share protocol is a dynamic hierarchical bandwidth allocation scheme. The intra-host regulation protocol uses a rate regulator for guaranteed traffic and best-effort traffic is kept in a separate queue. Whenever no guaranteed traffic is eligible to be sent, best-effort traffic, if any, is served.

### 3.3 The hierarchical link sharing

Our model is inspired from the Hierarchical Link Sharing concept proposed in [6]. According to this concept, each node in a hierarchy is assigned a weight and should receive its allocated bandwidth by means of a set of control functions. A node can be a traffic class (e.g the real-time class), a traffic type (e.g. telnet traffic), etc.

An example of link sharing algorithms is The Hierarchical Fair Service Curve Algorithm by Stoica et al. [20]. Another proposal is The Class Based Queueing CBQ by Floyd et al. [6]. Bennet and Zhang have also presented an overview of The Hierarchical Packet Fair Queueing Algorithms H-PFQ in [1]. These algorithms define a class hierarchy (a tree) for each link. A class represents a collection of flows grouped into organizations, protocol families, types of traffic, etc. Each class receives a minimum bandwidth (expressed as a weight) totally consumed if sufficient traffic is generated. Otherwise, the unused bandwidth (or excess bandwidth) is fairly shared between active classes. These algorithms are proposed as packet scheduling techniques in public network routers. They differ from each other in the definition of the class scheduling technique and in the policy applied in case of excess bandwidth distribution. In case of congestion, some packets may be lost since neither shaping nor policing is performed. The root node of the tree performs packet scheduling on the aggregate traffic. Intermediate nodes in the tree (other than the root node) are used by the packet scheduling algorithm to decide which packet to serve.

In the context of our work, a tree representation of the VPN virtual links can be of interest since we can divide a LAN into administrative groups, give priority to some traffic types and share bandwidth of outgoing links by statically assigning weights to each node. The model we propose is also based on the hierarchical link sharing concept. Nevertheless, our model differs from previously presented models (CBQ and H-PFQ) in 4 ways:

- In our model, intermediate nodes in the tree (other than leaf nodes and root node) do not only represent aggregate flows but also control entities.
- These control entities are aimed at performing traffic shaping as well as flow isolation between guaranteed applications sharing the same virtual link.
- In our model, excess bandwidth is distributed among only best-effort flows since we consider that excess bandwidth is not a guaranteed resource and cannot be used by guaranteed flows.
- The implementation of our model is distributed. Not only does the root node serves packets (like for CBQ since the tree representation is logical) but also the intermediate nodes perform control functions and schedule packets before sending them to the root node.

### 3.4 The traffic management model

During VPN configuration, the root node negotiates with the provider edge device the establishment of a virtual link and the allocation of its peak bandwidth. Throughout this study, we consider a LAN as composed of subnetworks $S_1$, $S_2$, ... $S_n$ (see Figure 6).

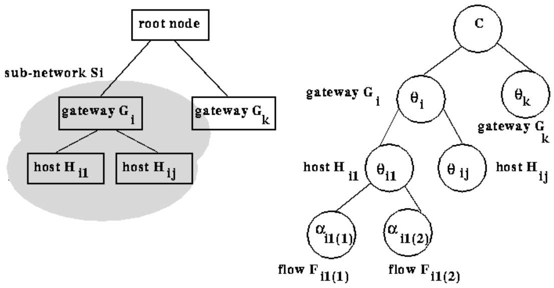


Figure 6: A tree representation of LAN outgoing links

The **root node** consists of the customer edge device (defined above in section 3.1. A **gateway** is the equipment that connects a sub-network to the other subnetworks of the LAN and to the root node. Gateways $G_1$, $G_2$, ... $G_n$ are connected by transmission links with fixed capacity to the root node.

Several gateways may share a virtual link (an ATM VP is a multiplex of ATM VCs). The root node manages bandwidth on a virtual link (e.g. a VP) by sharing it between gateways (e.g. a VC per gateway). It first gives each gateway $G_i$ a static weight $\Phi_i$ that represents the minimum amount of bandwidth it can use. Then, it periodically computes new weights depending on the effective bandwidth utilization of each gateway and informs the gateways about them. Depending on real bandwidth needs and traffic evolution, the root node decides whether to negotiate with the provider edge device the increase or the decrease of bandwidth allocation on a specific virtual link. This enables an optimum balance to be achieved on links connecting the LAN to the other sites of the VPN. A gateway $G_i$ in each subnetwork $S_i$ is responsible for all outgoing traffic generated by hosts enclosed in $S_i$. It shares this amount of bandwidth among all hosts using the same virtual link by assigning a weight to each host. Each host should ensure that bandwidth is fairly shared between flows set up on the same virtual link. Generally, we can distinguish at least two traffic service classes: **the guaranteed service class** and **the best-effort service class**. Each flow belongs to one of these 2 classes and packets should be scheduled by the host according to the traffic class they belong to and to the traffic parameters of active flows.

### 3.5 The hierarchical bandwidth manager

The hierarchical bandwidth manager relies on two protocols performed by the nodes of the hierarchy. Each node in the tree respects either an **Inter-Node Bandwidth Share** protocol or an **Intra-Host Regulation** protocol. In the following, we will focus on the management of a single virtual link. By applying the same mechanism on each virtual link, all VPN virtual links are controlled.

Each virtual link $L$ is characterized by a set of parameters $(C, S, D)$ where $C$ denotes its capacity (in Megabits/s), $S$ the source address and $D$ the destination address of the link. Let us suppose that the root node has to control traffic on a virtual link $L$ with the set of parameters $(C, S, D)$. Note that $S$ in our model represents an interface address of the root node. Traffic generated by all hosts within the LAN and destined to some VPN sites is automatically routed through this node. This latter is able to establish virtual links necessary to reach the requested destinations through the VPN.

The virtual link capacity $C$ is to be shared between gateways $G_1$, $G_2$, ... $G_n$. If $\Gamma$ is the set of gateways, then we have:

$$C = \sum_{i \in \Gamma} r_i, \text{ where } r_i = \Phi_i \times C \tag{1}$$

$r_i$ represents the bandwidth share of gateway $G_i$ and $\Phi_i$ its weight. Each host $H_i^j$ connected to gateway $G_i$ has a weight $\Phi_i^j$ that could be attributed on an administrative hierarchy basis. If $\Gamma_i$ is the set of hosts connected to gateway $G_i$, the traffic generated by a host $H_i^j$ should be limited to:

$$r_i^j = \Phi_i \times \Phi_i^j \times C, \text{ where } \Phi_i = \sum_{j \in \Gamma_i} \Phi_i^j \tag{2}$$

Host $H_i^j$ is intended to share $r_i^j$ between its active applications depending on their networking parameter needs. A flow can either belong to the guaranteed service class or to the best-effort service class. In order to deliver QoS to all guaranteed flows, it is necessary to apply traffic shaping on guaranteed flows and to share unused bandwidth between best-effort ones.

To explain the principle of this bandwidth sharing, we present an example of a hierarchy in Figure 7. In this hierarchy, the maximum capacity $C$ of the link is 5Mb/s. Gateway $G_1$ is allowed to send $4Mb/s$. Host $H_1^1$ is allowed a maximum bandwidth of $3Mb/s$. If the total guaranteed traffic at host $H_1^1$ is less than this maximum, it is useful to make the unused bandwidth available for best-effort flows. Otherwise, if host $H_1^1$ has no sufficient traffic to use these $3Mb/s$, excess bandwidth may be used by host $H_1^2$. Moreover, if the aggregate traffic at gateway $G_1$ is less than $4Mb/s$, it is reasonable to make gateway $G_2$ consume the excess bandwidth. Only best-effort flows are authorized to share the excess bandwidth. This is because we assume that guaranteed flows are constantly shaped. Also, excess bandwidth can not be considered as a guaranteed resource.

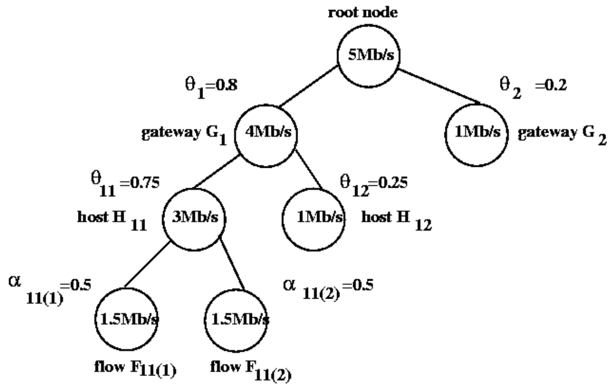


Figure 7: An example of a hierarchy

## 4 The control components

Each host applies traffic shaping on guaranteed flows. When guaranteed flows are silent, the host sends the best-effort flows taking into consideration the maximum allowed rate (this will be explained later how this rate can be dynamically evaluated). This ensures that each host sends at its maximum capacity. This concept is referred to as the intra-host regulation protocol and is implemented through a mechanism called the fair shaper (it will be explained in section 4.2).

A dynamic weight computation ensures that when some nodes are idle or sending less than their allocated bandwidth, the excess bandwidth will be distributed between active sibling nodes. Each parent node periodically estimates current load, re-evaluates its children weights and informs its children about them (always greater or equal to the initial ones). Only the root node and gateways are concerned with such dynamic weight computations. This concept is referred to as the inter-node bandwidth share protocol and will be described in the next section.

### 4.1 The inter-node bandwidth share protocol

Consider a parent node $N_i$ (the root node or a gateway) with weight $\Phi_i$ and capacity $C_i$ ($\Phi_i = 1$ for the root node and $C_i = C \times \Phi_i$ ). Let $\Gamma_i$ be the set of its children. Let $N_i^j$ be a child node, $\Phi_i^j$ its weight, $\Delta_i^j$ the excess bandwidth that it can borrow from sibling nodes in addition to its share and $\lambda_i^j$ its actual transmission rate.

If $U_i$ denotes the set of underloaded nodes (i.e. $\{ N_i^j \mid \lambda_i^j < \Phi_i^j \times C_i \}$), then:

**Definition 1** *A bandwidth allocation is said to provide inter-node bandwidth share if for each node*

$N_i^j$, *there exists an excess bandwidth* $\Delta_i^j$ *defined as follows:*

$$\Delta_i^j = \begin{cases} \frac{(C_i - \sum_{j \in \Gamma_i} \lambda_i^j) \times \Phi_i^j}{\sum_{k \notin U_i} \Phi_k} & \text{if } N_i^j \notin U_i \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

*whenever* $Card(U_i) \neq 0$ *and the new weight becomes:*

$$\Phi_i^j := \Phi_i^j + \frac{\Delta_i^j}{C_i} \tag{4}$$

where Card(X) denotes the cardinality of a set X. According to this definition, an inter-node bandwidth share is optimal whenever:

$$\sum_{j \in \Gamma_i} (\Delta_i^j + \lambda_i^j) = C_i$$

The root node $N_i$ needs to keep information on the actual transmission rate of its children nodes and their respective weights. New weights are computed according to equation (4) and sent to corresponding nodes periodically.

## 4.2 The intra-host regulation protocol: The fair shaper

Before presenting the intra-host regulation protocol, we explain the context in which the protocol was designed.

Our purpose is to share on a host machine, bandwidth between flows using the same outgoing link. We assume that a multi-task operating system (e.g. Solaris 2.5) is running on the host. Each application belongs to a certain service class. Depending on its networking parameter needs, an application requests a set of traffic parameters. It is then expected to respect this set of parameters. The operating system must manage end-system resources so that processing needs implied by the bandwidth and delay requirements of each connection could be satisfied.

In current operating system implementations, flow isolation is not respected and traffic shaping is not performed. Guaranteed flows may send more traffic than what has been negotiated. The overall multiplex may consequently exceed the link capacity. Moreover, a guaranteed flow may also be "disturbed" by a best-effort one when they both compete for accessing common resources (CPU, memory access, network driver access, ... ).

In this context, we propose to perform traffic shaping at the source so that any bursty source is prevented from sending non conforming packets. Each flow is confined to respect its negotiated traffic parameters. Also, by using an adequate packet scheduling algorithm, flow isolation can be ensured.

### 4.2.1 The algorithm design

In order to perform traffic shaping coupled with packet scheduling, we have designed a new algorithm that we call the Fair Shaper to be implemented in the network driver of the Solaris 2.5 operating system. Such an algorithm applies traffic shaping per connection with a very small probability to block any application process when a non conforming packet arrives to the network driver (process blocking is very costly [13]). It also uses the Self Clocked Fair Queuing algorithm proposed by Golestani [11] as a packet scheduler. We first explain the main idea behind the fair shaper algorithm before justifying the choice of combining traffic shaping with packet scheduling.

In a classical network driver (the IP driver in our case), all flows are multiplexed in the driver queue. This driver is managed by a single process, i.e. all flows share the same environment. All packets (from different flows) are stored in this common queue and are served in a FIFO (First In First Out) manner. This FIFO queueing discipline does not ensure flow isolation and does not take into account the networking parameters of guaranteed flows. In order to ensure flow isolation, we propose to modify this queueing discipline by integrating a simple and efficient mechanism.

The proposed mechanism computes a departure time and an insertion tag for each guaranteed packet before it inserts it in the driver queue. Insertion tags can be computed according to a Generalized Processor Sharing GPS-like policy [17]. Since we are concerned with a packet system (the network driver), we propose to use a packet approximation of the GPS policy. One of the Packet Fair Queueing algorithms PFQ [3] could be used. As explained above, the SCFQ algorithm is used for its simplicity and because it presents good fairness properties, especially in the context of our work (more explanations on this choice can be found in [15]). The SCFQ algorithm computes insertion tags so that flows are served proportionally to their traffic parameters. Packets are then inserted in the driver queue in the increasing order of these tags.

For traffic shaping, we compute a theoretical departure time for each arriving guaranteed packet. This theoretical departure time is computed according to the flow traffic parameters. If the packet comes in advance (e.g. if the source is bursty), it is blocked in the driver queue waiting for a timer to expire.

Since we will use the same queue for traffic shaping and for packet scheduling, we propose to combine the two mechanisms into a single algorithm that we call the fair shaper.

### 4.2.2 Why combine fair queueing and traffic shaping?

It can be said that it is sufficient to insert guaranteed packets in increasing order of their departure times in the queue to ensure fairness. This is not optimal in the context of our work. In fact, we take into account the operating system properties in the design of the fair shaper. Here, the driver queue offers an indexed access since it has a limited size (the number of simultaneous active applications is also limited by the processing capacity of the machine). The fair queuing complexity is hence reduced to $O(1)$ (since we do not have to perform an ordered insertion). But if we insert packets in increasing order of departure times, the complexity is increased to $O(log(N))$ and we will be obliged to re-arrange the queue for certain packet arrivals. A packet $P_i^k$ from flow $i$ coming early (has to be delayed) may be inserted in a wrong position. When another packet $P_j^m$ arrives later but has to be sent before $P_i^k$, we will have to re-arrange the queue to insert $P_j^m$ before $P_i^k$. To avoid such sorting (a costly operation in the context of high speed connections), it is necessary to compute exact packet positions by a packet fair queuing algorithm. We now illustrate why it is not possible to do traffic shaping by only using a weighted fair queuing algorithm. When all sources are active and sending at their respective maximum rates, the output multiplex is naturally shaped. In the absence of some connections, the server fairly divides this bandwidth excess among other back-logged connections (work-conserving property) and hence individual shaping is violated.

### 4.2.3 The fair shaper algorithm

The fair shaper consists of a packet scheduling server. It is composed of an arrival module and a service module (see Figure 8). When a guaranteed packet (from a guaranteed flow) arrives, the arrival module computes its position in the queue based on a Weighted Fair Queueing algorithm [3]. This position is called virtual finishing service tag. The arrival module also computes its theoretical departure time based on a minimal packet inter-arrival time and then inserts the packet in the server queue. When a best-effort packet arrives, the arrival module has to insert it in the first empty entry in the server queue. This ensures that the server will be work-conserving and hence resource utilization maximized. The arrival module links empty entries together to form a second queue. This queue is used to store best-effort packets.

### 4.2.4 The service discipline.

The service module is backlogged whenever packets are still in its queue. It extracts the first packet from the guaranteed queue and sends it only if its theoretical departure time is reached. Otherwise, it sends the head of the best-effort queue. For each guaranteed flow $i$, a weight $\alpha_i$ is computed proportionally to its traffic parameters (for example one could consider CBR flows with $\alpha_i$ proportional to $PCR$). When the $k^{th}$ packet $P_i^k$ of length $l_i^k$ arrives at time $a_i^k$ from guaranteed flow $i$, the arrival module computes its virtual finishing service tag $F_i^k$ according to the Self Clocked Fair Queueing algorithm SCFQ [11] and its theoretical departure time $D_i^k$ as follows:

$$F_i^k = \max(F_i^{k-1}, v(a_i^k)) + \frac{l_i^k}{\alpha_i} \tag{5}$$

$$D_i^k = \max(D_i^{k-1} + T_i, a_i^k) \tag{6}$$

The virtual time function $v(t)$ is equal to the service tag of the packet in service at time $t$. $T_i$ corresponds to the minimum inter-arrival time between packets considered as bursts. In the case of an ATM virtual link (a VP or a VC), this parameter is computed so that the burstiness caused by the fragmentation of an IP packet into ATM cells is absorbed (more details on this computation can be found in [15]). The following theorem proves the correctness of this algorithm by demonstrating that theoretical departure times and service tags are coherent:

**Theorem 1** *For any pair of packets $P_i^k$, $P_j^l$ which have virtual finishing service tags $F_i^k$, $F_j^l$ respectively, and theoretical departure times $D_i^k$, $D_j^l$ respectively,*

$$F_i^k \leq F_j^l \Leftrightarrow D_i^k \leq D_j^l \tag{7}$$

The proof of this theorem can be found in [15] and is based on the Generalized Processor Sharing GPS [17] and the Packet Fair Queueing PFQ [3] properties.

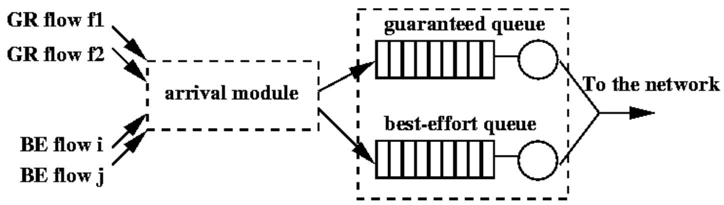


Figure 8: The fair shaper architecture

## 5 Protocol design

In this section, we present guidelines for the implementation of the inter-node bandwidth share within the root node and the gateways. We also describe the implementation of the fair shaping mechanism in hosts. A detailed description of the fair shaper implementation can be found in [15].

### 5.1 Inter-node bandwidth share protocol.

The protocol we propose is completely distributed, i.e., gateways do not need to keep state information on each other. A gateway just communicates with the root node and its children hosts. As explained in section 4.1, only the root node and gateways are concerned with dynamic weight computation. For the sake of simplicity, we explain how the protocol works in a gateway. At the root node, we use the same protocol with some name translation (e.g. we just replace host by gateway and gateway by root node to determine the algorithm at the root node).

The gateway needs to keep some information on its own weight, the bandwidth capacity of a link, the weights of all its children and their actual transmission rates. Two tables are defined by the gateway: the first one, called **links table**, stores information about the available links, their capacities, the gateway initial and current weights. An entry in this table has the following structure:

| Link identifier | Link capacity | Initial weight | Current weight |
|---|---|---|---|

The second table, called **hosts table**, stores information about hosts. An entry in this table is composed as follows:

| Host identifier | Initial weight | Current weight | Effective rate |
|---|---|---|---|

The initial weight is a float number that is attributed statically by a network manager to a specific host. The current weight is used to keep track of the current share. The effective transmission rate is a counter of received bytes from a host during an update interval.

At the end of an update interval, the gateway follows these steps to compute new weights:

1. It reads the hosts table to determine the set of under-loaded hosts (as defined in section 4.1).
2. The gateway tests if this set is empty.
   (a) If it is empty, no excess bandwidth is to be distributed.
   (b) If not, new weights are computed according to equations 3 and 4.
3. The gateway verifies if its parent node has sent a control message with a new weight.
   (a) If the weight has changed (excess bandwidth is to be shared at the parent node or a sibling node becomes active and wants to recuperate its bandwidth share), the gateway updates the links table with its new current weight.
   (b) , nothing is to be done.
4. The gateway sends its new bandwidth share (new current weight read from links table) to all its children (hosts) and their new weights, within control messages. These new weights are computed in step 2.a (read from hosts table).

### 5.2 Notes

- When a node temporarily sends at a lower rate, the unused bandwidth will be viewed as excess at the parent node and is reallocated to the other nodes. When this same node speeds up to its regular rate in the next period, the total traffic at the parent node will exceed its maximum capacity. We use for this purpose at the parent node a specific queue that stores packets in excess and schedules them in the next period since no space is left for the current period.

  Clearly, such a solution may violate a guaranteed flow. Nevertheless, this violation may be reduced if the period duration is chosen according to the LAN dimensions, to traffic variability etc, ... We should hence choose a period small enough to avoid queue overflow and to guarantee no violations. If this situation occurs, the parent node re-attributes their initial weights to its children at the end of the period.

- It is to be noted that in the host table, we always keep the initial weight in order to be able to decrease the transmission rate of some hosts when under-loaded ones become able to use their full share. We keep information on the current weight to determine if the host is under-loaded or not.

- In the links table, we keep information on the gateway initial weight in order to be able to decrease its share when other gateways become able to use their initial share.

Due to their scalability and resilience to partial failures, IP based networks are ideal for the transport of short control messages. To implement the inter-node protocol, the root node needs to exchange short messages with gateways in order to inform them on their weights periodically. Also, gateways need to communicate with hosts via simple messages. For the current implementation, we used simple IP packets. We plan to use the SNMP protocol for the control message exchange.

## 6 Implementation of the fair shaper:

We implemented the fair shaper in end-stations. A multi-task operating system was upgraded by the server implementation (Solaris 2.5). In order to guarantee a minimal number of modifications in existing applications, we implemented our solution in the kernel space. A brief description of this environment is hence necessary to later tackle the implementation details. A stream is a full-duplex bidirectional processing and data-transfer path. It is established between a network driver in the kernel space and an application in the user space. In a classic TCP/IP (UDP/IP) stack, we identify:

- The stream head which is an interface between the user and the protocol stack. It maps user requests into corresponding kernel routines. A standardized application programming interface API is offered to the user to access networking services.
- A cooperating module TIMOD which is responsible for API processing.
- The TCP, UDP and IP drivers as well as the ARP module.
- The DLPI driver which provides an interface between the network device and the IP driver.

There is one stream per logical network interface. Each connection request initiated by the user application (either a TCP or a UDP application) creates a logical stream. From an operating system point of view, a TCP or UDP process is created for each connection but a unique IP process is used for all connections. For each connection, we add a specific data structure which is interpreted as the traffic contract of the source. These information elements are transferred by the stream head to the IP driver. Packets from all connections are multiplexed at the IP driver. This latter creates an indexed table entry for each connection to store specific information elements (source port, destination port, traffic contract, etc.). So, we are able to know to which connection a packet belongs.

We have used a video application (constant bit rate MPEG I coder) to represent the guaranteed application and a classic UDP-based client-server application to represent the best effort one. The experiments undertaken in the context of the JAMES project [12] allowed us to test the connectivity between 2 sites through a public ATM network (on the PVC connecting IFNF-Milan and CNET-Lannion). We also tested the performances of the fair shaper between the 2 sites.

### 6.1 Tests related to the JAMES project

During this work, the ONIP protocol stack [2] was tested. ONIP implements a protocol stack based on the direct communication between TCP and the ATM layer using a dummy IP layer. This protocol stack includes the implementation of the fair shaper. In particular, it is able to deliver a per virtual-channel QoS to user applications. Some applications were modified in order to adapt them to experimental needs. Applications in which the QoS plays a fundamental role were tested in greater detail (video applications). Tests were done in different network conditions in terms of load, concurrence, fairness of bandwidth share and congestion. The availability of the JAMES network [12] was useful to perform tests in real environments on a wide area ATM network (on the PVC connecting IFNF-Milan and CNET-Lannion). An access to the JAMES infrastructure was used. On the customer side, a native ATM access was available. For this reason, an ATM switch was connected to the JAMES infrastructure. Workstations with ATM adapters providing ATM API's for a direct access to the ATM/AAL layers were connected to the switch. Tests were performed only on PVCs and not on SVCs due to major problems of interoperability between the signaling modules of the testbed (signaling was successfully tested in the local environment). The purpose was to analyze the performance of the fair shaper applied at the source. Flows belonging to different service classes (UBR, CBR and VBR) were multiplexed on the same VC. A best-effort flow (UBR) was represented by a Mosaic client modified to run on top of the ONIP stack. A guaranteed flow (CBR) consists of an application called MPX that was listening on an UDP port. The client can request a particular film to the source which starts sending the requested film on the specified port. The MPX application waits on such a port and displays the film as soon as it receives packets. We measured performances by analyzing the output of the atmsnoop command, which is included in the ATM distribution of SUN ATM SBus adapters. It is to be noted that a contract of 2Mb/s on the PVC was negotiated with the ATM network operator.

Figure 9 shows the packet inter-arrival times for video traffic.

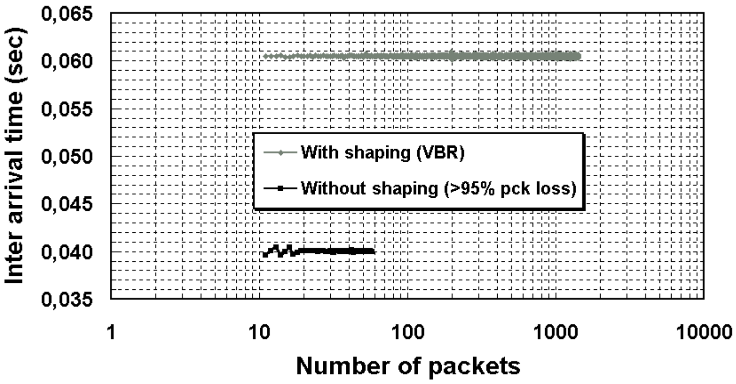


Figure 9: Video traffic with and without fair shaping

The upper curve represents the traffic relative to the transmission of an Mpeg I movie with the fair shaper applied at the transmitting source (Inter-arrival time was computed according to 4). The lower curve refers to the transmission of the same Mpeg movie but with no shaping applied at the source. In this latter case, without shaping at the source, more than 95% of packets were lost and only fewer frames of the Mpeg movie could be seen. This was of course due to the fact that the source was sending too fast and therefore violating the network contract of 2 Mbits/s. On the other hand, with the fair shaping mechanism applied at the source, the inter arrival time between packets was constant and no packets were lost. The Mpeg movie could be seen with the same quality as when seen locally. The PCR was set to 3000 cells/s. Considering that the length of each packet was 9040 bytes, we expected to have a packet inter-arrival time of 0.063 seconds. The value that was obtained was 0.0605 seconds.

Figure 10 shows the inter-arrival times between packets relative to an HTTP connection (UBR) and a video connection (CBR).

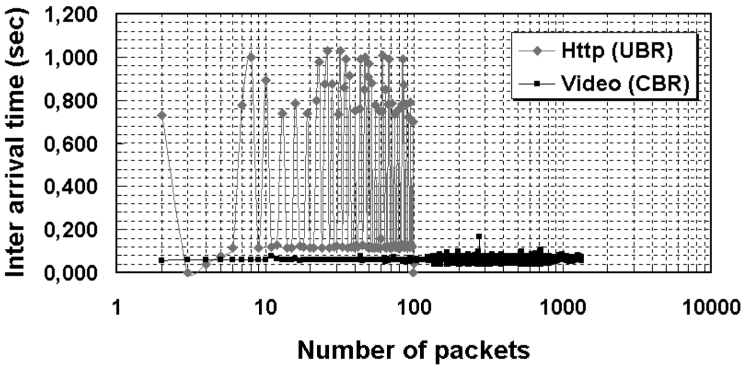


Figure 10: Video traffic with and without the fair shaper

As it can be seen, since the video traffic is CBR, the inter-arrival time between packets remains constant (0.06s) while the inter-arrival time between UBR packets (http connection) fluctuates in a considerable way, reaching maximum values of $1 second$. Therefore, the movie was still of good quality even if an http connection was running simultaneously.

## 7 Simulation of the inter-node bandwidth share

We used the ns [22] network simulator from Lawrence Berkeley National Laboratory (LNBL) for our simulations. In current implementations of the ns network simulator, support is included for Drop-Tail (FIFO) queueing, RED buffer management, CBQ mechanism and variants of Fair Queueing algorithms [5]. We implemented the fair shaper algorithm as a new scheduling discipline. This consists of a new class implemented as a module added to the available package. In each node created by the simulator, a classifier object receives packets, examines their fields and then maps the values to an outgoing interface. As explained above, only gateways and the root node are concerned with the implementation of the inter-node bandwidth share protocol. We modified the classifier object to support the algorithm described in 5.1. We used the simple topology presented in 3.5.

#### 7.0.1 Experiment 1

We first test the implementation of the fair shaper. We suppose that all nodes of the hierarchy are idle except for host $H_{11}$. In host $H_{11}$, we establish a best-effort flow $BE_{11}$ and 2 guaranteed flows, $GR_{11}$ and $GR_{12}$. Both guaranteed flows have the same weight (0.5) and are intended to share the amount of bandwidth of their parent node ($3Mb/s$). Since all the other nodes of the hierarchy are idle, $H_{11}$ can use up to $5Mb/s$. Guaranteed flows should be shaped so that each one can use up to $1.5Mb/s$ and the best-effort flow will consume the remaining bandwidth $2Mb/s$. We measured the instantaneous rate of each flow.

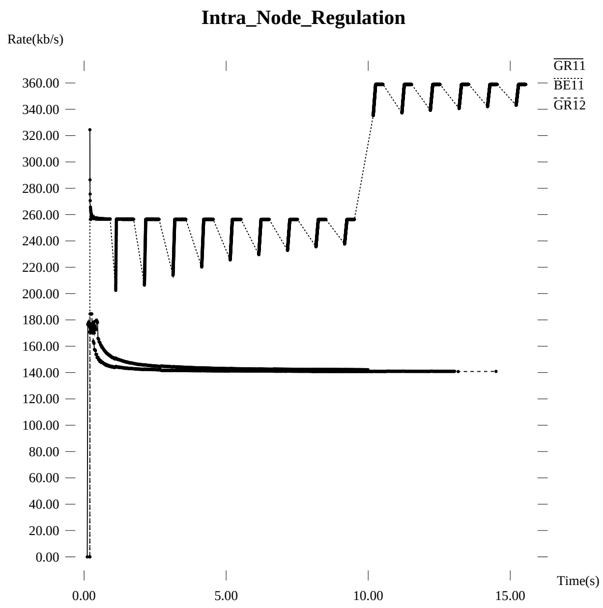


Figure 11: The Fair Shaper Results

As it can be seen in Figure 11, guaranteed flows $GR_{11}$ and $GR_{12}$ are well shaped and the best-effort $BE_{11}$ has a variable rate with a mean rate of approximately $2.2Mb/s$. When flow $GR_{11}$ is stopped (at time 10), flow $BE_{11}$ sees its rate increase up to $3.5Mb/s$.

### 7.0.2 Experiment 2

In this experiment, we suppose that gateway $G1$ is active during [0..30]. Gateway $G2$ is active during [0..20], idle during ]20..25] and again active at ]25..30]. At time 20, when $G2$ becomes idle, gateway $G1$ should be enabled to consume the total capacity of the link ($5Mb/s$). Also, at time 25, when gateway $G2$ is enabled again, this latter should be able to speed up to its allocated rate ($1Mb/s$) and at the same time, $G1$ is again attributed its initial weight (sending rate less or equal to $4Mb/s$).

In the following experiment, we only represent the behavior of flows belonging to gateway $G1$. The behavior of the other flows (from gateway $G2$) can easily be guessed. During the whole duration of the experiment, we establish a guaranteed flow $GR_{11}$ (shaped to $2.5Mb/s$) and a best-effort flow $BE_{11}$ in host $H_{11}$. During [0..15], a guaranteed flow $GR_{12}$ (shaped to $0.8Mb/s$) and a best-effort flow $BE_{12}$ are active in host $H_{12}$.

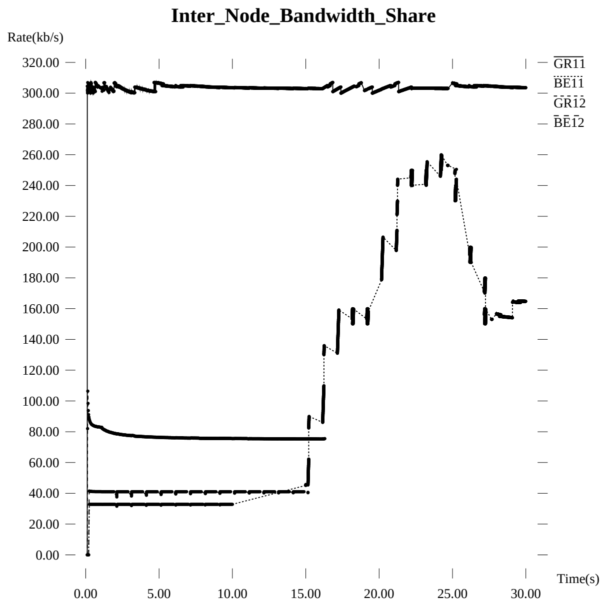


Figure 12: The Inter-Node Bandwidth Share Protocol Results

As can be seen in Figure 12, at time 15, when host $H_{12}$ becomes idle, flow $BE_{11}$ is enabled to consume the excess bandwidth (its rate approximates $1.5Mb/s$). At time 20, when gateway $G2$ becomes idle, flow $BE_{11}$ sees its rate increase up to $2.5Mb/s$. At time 25, when gateway $G2$ becomes active again, best-effort flow $BE_{11}$ sees its rate reduced to $1.5Mb/s$ since flows from gateway $G2$ are now able to consume their allocated bandwidth ($1Mb/s$).

These results show that bandwidth regulation goals are reached with such an implementation. Details of implementation of the fair shaper on a Solaris 2.5 machine can be found in [15].

## 8 Conclusion

In this paper, we have presented a bandwidth regulation mechanism able to control link bandwidth in a LAN connected to a VPN. The mechanism copes with best-effort and guaranteed flows so that bandwidth left unused by guaranteed flows is divided among best-effort ones throughout the LAN. We used a tree representation of the LAN nodes. The Hierarchical Bandwidth Manager is performed in a hierarchical and distributed manner. We introduced two new concepts called inter-node bandwidth share and intra-host regulation protocol. We explained how both of them dynamically enforce bandwidth regulation over the whole network. We also presented a new algorithm called the fair shaper used at the leaf nodes and introduced some implementation details in a Solaris 2.5 operating system. The bandwidth manager is shown to achieve near to optimal bandwidth utilization and traffic regulation.

## References

[1] J. Bennet and H. Zhang. Hierarchical Packet Fair Queueing Algorithms. In *Proc. SIGCOMM'96*, 1996.

[2] D. Bonjour, O. Elloumi, and H. Afifi. Internet Applications over Native ATM. *Computer Networks and ISDN systems*, (1998).

[3] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. *Journal of Internetworking Research and Experience*, pages pp. 3–26, (1990).

[4] N. Duffield, A. Greenberg, and P. Mishra. A Performance Oriented Service Interface for Virtual Private Networks. Work in progress ¡draft-duffield-vpn-qos-framework0, 1998.

[5] K. Fall and K. Varadhan. *ns* Notes , Documentation. A collaboration between reserachers at UC Berkley, LBL, USC/ISI and Xerox PARC, 1998.

[6] S. Floyd and V. Jacobson. Link-Sharing and Resource Management Models for Packet Networks. *ACM/IEEE Transactions on networking Vol.3 No.4*, (1995).

[7] ATM forum. LAN emulation over ATM 1.0. af-lane-0021.000, 1995.

[8] ATM forum. Multi-Protocol Over ATM Specifications v1.0. af-mpoa-0087.000, 1997.

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

[10] B. Gleeson and J. Heinanen. A Framework for IP based Virtual Private Networks. Work in progress ¡draft-gleeson-vpn-framework-00.txt, 1998.

[11] S. Golestani. A Self-Clocked Fair Queueing Scheme for Broadband Applications. In *Proc. IEEE INFOCOM'94*, pages pp. 636–646, 1994.

[12] TF-TEN group. The JAMES project. http://www.labs.bt.com/profsoc/james/, 1997.

[13] S. Khanna, M. Sebrée, and J. Zolnowsky. Real-time Scheduling in SunOS 5.0. In *Proc. Usenix 92-Winter 92*, 1992.

[14] L. Lamti and H. Afifi. A Hierarchical Bandwidth Manager for Local Area Network Outlets. In *Proc. IEEE Globecom'98*, 1998.

[15] L. Lamti and H. Afifi. The Fair Shaper: An Efficient Mechanism for Internet B,width Share over ATM in a Multi-Tasks OS. In *Proc. IEEE ATM workshop ATM'98*, pages pp. 56–64, 1998.

[16] L. Lamti, S. Ben-Yacoub, and H. Afifi. A Statistical Approach for VPN Links Traffic Characterization. In *Technical Report, ENST Bretagne*, 1999.

[17] A. Parekh and R. Gallager. A Generalized Processor Sharing Approach to Flow Control - The Single Node Case. *ACM/IEEE Transactions on networking*, pages pp. 344–357, (1993).

[18] S. Sathaye. The ATM forum Traffic Management Specification, version 4.0. ATM Forum/95-0013R6, 1996.

[19] W. Simpson. IP in IP Tunneling. IETF RFC 1853, 1995.

[20] I. Stoica and H. Zhang. A Hierarchical Fair Service Curve Algorithm for Link-Sharing Real-Time and Priority Services. In *Proc. SIGCOMM'97*, 1997.

[21] D.P. Tranchier, P.E. Boyer, Y.M. Rouaud, and J.Y Mazeas. Fast Bandwidth Allocation in ATM Networks. In *Proc. XIV ISS*, 1992.

[22] Berkley University. *ns [online]*. available http://www-nrg.ee.lbl.gov/ns, 1998.

[23] S. Walters and M. Ahmed. Broadband Virtual Private Networks and their Evolution. In *Proc. XIV ISS*, 1992.