A TRAFFIC MANAGEMENT MODEL FOR VIRTUAL PRIVATE NETWORK LINKS

Leila Lamti
Ecole Nationale Supérieure des Télécommunications
2 Rue de la châtaigneraie, BP 78 35512 Cesson Sévigné - France
Tel: (+33) 2.99.12.70.51
Fax: (+33) 2.99.12.70.30
Leila.Lamti@enst-bretagne.fr

Abstract

A Virtual Private Network (VPN) is a solution that provides corporate networking between geographically dispersed sites. VPN sites consist of local area networks (LANs). 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. 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 link can be found. To respond to such needs, we propose to introduce in each LAN of the VPN a control architecture that we call The Hierarchical Bandwidth Manager. 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 node. Results of the implementation of the bandwidth manager in the ns network simulator are described in this paper.

1. INTRODUCTION

Efficient bandwidth sharing among multiple sessions has been analyzed in several research studies. Heterogeneity is one of the main problems in Quality of Service (QoS) provision and multiple traffic classes with different QoS requirements may be 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.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35580-1_16

J. D. Touch et al. (eds.), *Protocols for High-Speed Networks VI*

With the advent of constrained traffic (real-time applications, jitter or delay constrained sources, ...), the interest for the specification and development of new mechanisms able to cope with such traffic characterizations has increased.

Major networking communities have contributed in this field. The Traffic Management specifications for ATM networks [1] allow flexible bandwidth allocations at high speeds. They also allow a scalable architecture especially in the context of local area networks (LANs). The IETF community has also contributed in this field by designing and developing a new signaling mechanism called ReSerVation Protocol (RSVP) [4]. This protocol is used for resource reservation on the Internet. Both communities has defined several service classes. Each of these service classes is designed to provide certain QoS guarantees to traffic flows. Applications are expected to use one of these classes according to their needs.

In addition to QoS provisioning, a new concept for virtual sub-networking has emerged. This concept known as *Virtual Private Network (VPN)* [2] enables the interconnection of geographically dispersed sites. It provides corporate networking between geographically dispersed sites using the public switched network infrastructure. It can be accessed from any inter-working unit (a router, a LAN multiplexer) and it allows LANs interconnection. VPNs were more marketable products than well-defined telecommunication services because of a lack of standardization. The VPN concept is receiving more attention now. Several research projects as well as international standardization activities are involved in this field. Major networking communities have contributed in this field, especially within the IETF community where a significant amount of work is being done for the specification of the VPN service deployment over the IP backbone [8]. ATM backbones may also support a VPN service deployment and in the literature, several papers propose the use of ATM Virtual Paths for the provision of VPN services [7], [15].

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 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 links bandwidth allocation. Since traffic characterization is a difficult process, customers tend to over-allocate bandwidth on virtual links in order to prevent their traffic from any congestion, leading to an inefficient network resources 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) for flows multiplexed on the same virtual link, an adequate and sufficient bandwidth amount 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 receives the amount of bandwidth it has been agreed to use.

Our proposal will rely on two important features: flow isolation and traffic

shaping. Flow isolation is necessary since sensitive applications with stringent QoS requirements could malfunction due to bandwidth lack 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 control model that we call *The Hierarchical Bandwidth Manager*. 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. Control algorithms are performed by each node in the hierarchy in order to reach the 2 above-cited goals. By reaching these goals, bandwidth utilization could be maximized.

The paper is organized as follows: In section 2, we explain how the proposed manager may be combined to a VPN implementation and present its principle. In section 3, 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 4, we explain how the inter-node bandwidth share protocol can be implemented in existing equipments. The achieved results of both protocols in the *ns* network simulator [14] show that bandwidth regulation goals are reached.

2. TRAFFIC CONTROL REQUIREMENTS

Each company 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. We can represent VPN sites and their interconnection by means of a unique model. Some definitions are given in the following:

Customer Edge Device: It is the edge device at the customer side that connects the LAN to the provider network. This consists of the IP router that connects the LAN to the Internet backbone. It may also consist of the ATM switch that connects the LAN to the public ATM network.

Provider Edge Device: It is the edge device at the provider side to which the customer edge device is connected. As 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 in 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.

Figure 1 presents the generic VPN connectivity model that we will use for the rest of the paper and resumes all the above-mentioned definitions.

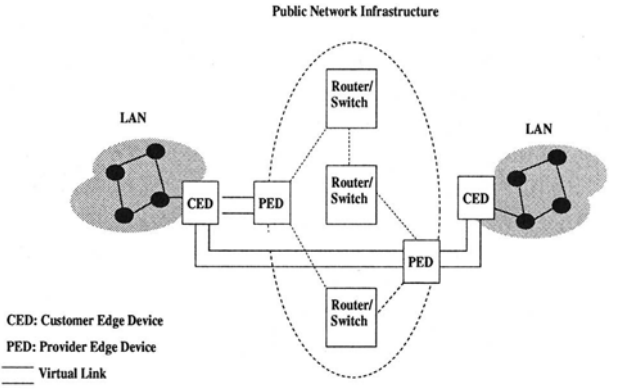

Figure 1 The generic VPN connectivity model

Requirements

In the generic VPN connectivity model, the LAN is connected to the VPN through a virtual link. For a VPN customer, an important task is to allocate the virtual links with sufficient capacities to deal with the external traffic needs. The high peak-to-mean ratios in LAN outgoing traffic volume 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 efficiently so that each flow receives the amount of bandwidth it has been agreed to use.

In this paper, we suppose that each virtual link is allocated an amount of bandwidth that we propose to share fairly. The sharing model we propose consists of a *hierarchical bandwidth manager* able to control the access to a VPN link from within a VPN site. The hierarchical bandwidth manager has two components: 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 top level of the hierarchy is the VPN link access device, what we called the customer edge device; the next level consists of gateway routers of subnets. And the last level consists of individual hosts. Starting from an initial set of bandwidth allocations, each parent machine monitors the bandwidth usage of each child machine, and periodically reallocates the excess bandwidth among the children machines that are not under-using their bandwidth allocations. The intra-host regulation protocol uses a rate regulator for guaranteed traffic, best-effort traffic is kept in a separate queue. Whenever no guaranteed traffic is eligible to send, best-effort traffic, if any, is served.

The Traffic Management Model

In our model, the *root node* consists of the customer edge device (defined above in section 2). A *gateway* is the equipment that connects a sub-network to the other subnetworks of the LAN and to the root node. Throughout this study, we consider a LAN as composed of subnetworks S_1, S_2, ... S_n (see Figure 2). Gateways G_1, G_2, ... G_n are connected by transmission links with

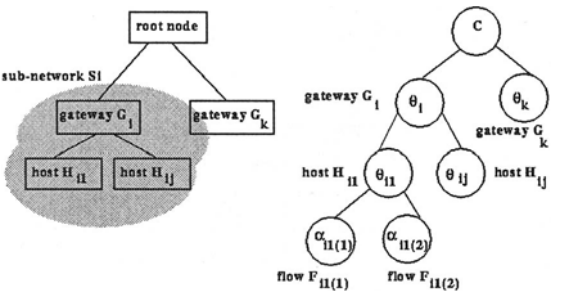

Figure 2 A tree representation of LAN outgoing links

fixed capacity to the root node.

During the VPN configuration, the root node negotiates with the provider edge device the establishment of a virtual link and the allocation of its peak bandwidth. 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). We attribute to each gateway G_i a static weight Φ_i that represents the minimum amount of bandwidth it can use. Gateways should then cooperate with the root node to dynamically re-adjust bandwidth allocations depending on the effective bandwidth utilization. The root node periodically computes new weights 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.

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 (a weight is given to each host). After receiving its weight from the root node, a gateway is then responsible for sharing this amount of bandwidth among hosts enclosed in S_i.

Finally, each host should ensure that bandwidth is fairly shared between applications (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.

Our model is inspired from the *Hierarchical Link Sharing* concept proposed in the literature [6]. According to this concept, each node in a hierarchy is assigned a weight and the main idea is that each node, by means of a set of control functions, should receive its allocated bandwidth (which is proportional to its weight). According to the link sharing concept, 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 and Zhang [13]. Another proposal is *The Class Based Queueing CBQ* by Floyd and Jacobson [6]. In [3], Bennet and Zhang presented an overview of *The Hierarchical Packet Fair Queueing Algorithms H-PFQ*. 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.

The above-mentioned models differ from each other in the definition of the class scheduling technique and in the applied policy in case of excess bandwidth distribution. In case of congestion, some packets may be lost since neither shaping nor policing is performed. They also assume that the root node of the tree performs packet scheduling on the aggregate traffic in a centralized manner. All intermediate nodes in the tree (other than the root node) are virtual. They are used by the root node to decide which class to serve with a given priority. These models are proposed for public network routers.

The same need for link sharing exists in a LAN 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) regarding 4 points:

- 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 can not be used by guaranteed flows.
- The implementation of our model is distributed. Not only the root node is serving packets (like for CBQ since the tree representation is logical) but also intermediate nodes perform control functions and schedule packets before sending them to the root node.

The Hierarchical Bandwidth Manager

The hierarchical bandwidth manager relies on two protocols performed by the nodes of the hierarchy [10]. 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.

The link capacity C is to be shared between gateways G_1, G_2, ... G_n. If Γ 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 Φ_i its weight. Each host H_i^j connected to gateway G_i has a weight Φ_i^j that could be attributed on an administrative hierarchy basis. If Γ_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, the host has to perform an admission control function. This function evaluates available bandwidth in order to decide whether to accept a new guaranteed flow or not. This is to ensure that already established guaranteed flows continue to receive the committed QoS parameters for the whole lifetime of the connection. Beyond connection admission control, it is necessary to apply a traffic shaping on guaranteed flows and to share unused bandwidth between best-effort ones.

To explain the principle of this bandwidth sharing, we present a hierarchy example in Figure 3. In this hierarchy, the maximum capacity C of the link is 5Mb/s. Gateway G_1 is allowed to send $4Mb/s$ and host H_{11} $3Mb/s$. If the total guaranteed traffic at host H_{11} is less than $3Mb/s$, it is useful to make the unused bandwidth available for best-effort flows that may be set up on host H_{11} (best-effort flows are not represented in this figure since they don't have any minimum bandwidth allocation). If host H_{11} has no sufficient traffic (guaranteed and best-effort flows included) to consume these $3Mb/s$, excess bandwidth may be used by host H_{12}. 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.

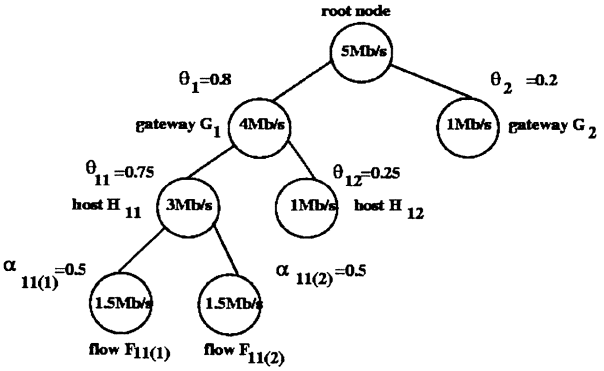

Figure 3 A hierarchy example

3. CONTROL COMPONENTS OF THE HIERARCHICAL BANDWIDTH MANAGER

Each host applies traffic shaping on guaranteed flows. When guaranteed flows are idle (or should be idle due to traffic shaping conditions), the host sends best-effort packets taking into account the maximum allowed rate (it will be explained later how this rate can be dynamically evaluated). Each host is entitled to send at its maximum capacity. This concept is referred to as intra-host regulation protocol and is implemented through a mechanism called the *fair shaper* (it will be explained in section 3).

A dynamic weight computation scheme 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 them about their new weights (always greater or equal to initial ones). Similarly, children nodes re-compute weights of their descendant nodes. This process is done recursively to share the excess bandwidth along the tree nodes. In our model, 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.

The Inter-node Bandwidth Share Protocol

Consider a parent node N_i (the root node or a gateway) with weight Φ_i and capacity C_i [1]. Let Γ_i be the set of its children. Let N_i^j be a child node, Φ_i^j its weight, Δ_i^j the excess bandwidth it can borrow from sibling nodes in addition to its share. λ_i^j represents its actual transmission rate.

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

[1] $\Phi_i = 1$ for the root node and $C_i = C \times \Phi_i$

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 Δ_i^j defined as follows:*

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

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

$$\Phi_i^j := \Phi_i^j + \frac{\Delta_i^j}{C_i} \quad (4)$$

According to this definition, an inter-node bandwidth share is optimal whenever:

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

A parent node N_i needs to keep information on the actual transmission rate of its children and their initial and current weights. New weights are computed according to equation (4) and periodically sent. A theoretical evaluation of the update interval optimal value depends on the LAN architecture and available buffers in gateways.

Intra-host Regulation Protocol: The Fair Shaper

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 (Solaris 2.5) is running on the host machine. We suppose that 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.

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). It also uses the *Self Clocked Fair Queuing* algorithm proposed by Golestani [9] 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 for each guaranteed packet a departure time and an insertion tag before it inserts it in the driver queue. Insertion tags can be computed according to a Generalized Processor Sharing GPS-like policy [12]. 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 [5] could be used. As explained above, the SCFQ algorithm will be used since it is the simplest one and presents good fairness properties, especially in the context of our work (more explanations on this choice can be found in [11]). 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*. More details on this algorithm and its motivations are described in [11].

The Fair Shaper Algorithm

The fair shaper consists of a packet scheduling server implemented in the network driver. It is composed of an arrival module and a service module (see Figure 4). When a guaranteed packet arrives, the arrival module computes its position in the queue based on the SCFQ algorithm. 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. The packet is then inserted in the driver 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 maximize bandwidth utilization when sufficient traffic is available. The arrival module links empty entries (in the driver queue) to form a second queue. This queue is used to store best-effort packets. The service

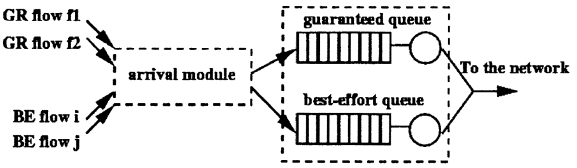

Figure 4 The fair shaper architecture

module is back-logged whenever packets are still in the driver 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 α_i is computed proportionally to its traffic parameters (for example one could consider CBR flows with α_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 SCFQ algorithm 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-based VPN, 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 [11]). 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 [11] and is based on the GPS and SCFQ algorithms properties.

4. IMPLEMENTATION

The hierarchical bandwidth manager controls bandwidth utilization on all virtual links of a VPN. It consists of a proposal for site-level traffic management. In this section, we present the guidelines for the implementation of the inter-node bandwidth share protocol within the root node and the gateways. A detailed description of the fair shaper implementation can be found in [11].

The 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 nodes. 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 3).
2. The gateway tests if this set is empty.
   (a) If it is empty, no excess bandwidth is to be distributed.
   (b) Else, 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 (an 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) Else, nothing is to be done.

4. The gateway sends to all its children (hosts), within control messages, its new bandwidth share (new current weight read from links table) and their new weights computed in step 2.a (read from hosts table).

Notes

- When a node temporarily sends at a lower rate, the unused bandwidth will be viewed as an 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 the traffic variability etc, ... We should hence choose a period small enough so that queue overflow and guarantees violations are avoided. If this situation occurs, at the end of the period, the parent node re-attributes to its children their initial weights.

- It is to be noted that in the hosts 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.

- IP packets are exchanged between the tree nodes to update state information.

Simulation Results

We use the ns [14] network simulator from Lawrence Berkeley National Laboratory (LNBL) for our simulations. 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. We modified the classifier object to support the algorithm described in 4. We use the simple topology presented in 2.

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 uses 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. As it can be seen in Figure 5, guaranteed flows GR_{11} and

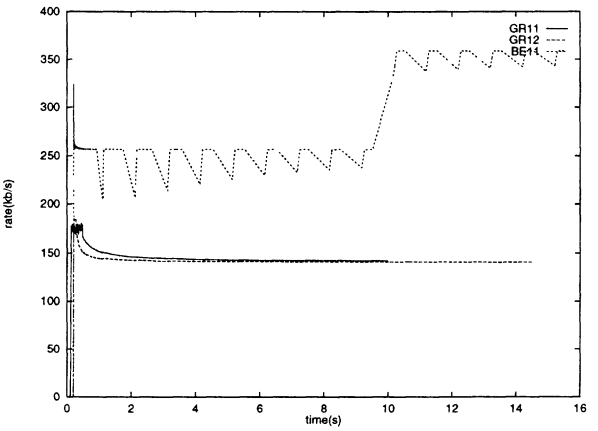

Figure 5 The fair shaper results

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 grow up to $3.5Mb/s$.

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 attributed again 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 be easily 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}. As it can be seen in Figure 6,

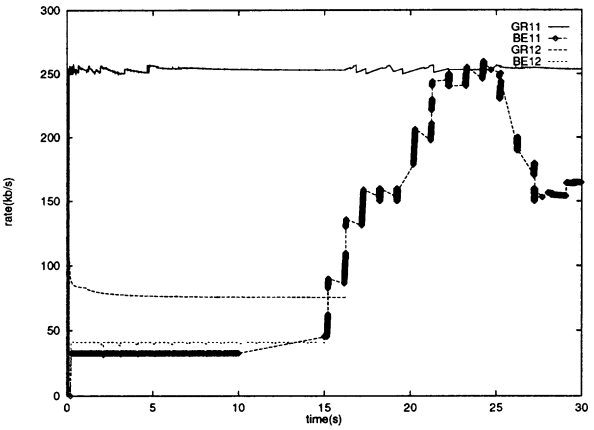

Figure 6 The inter-node bandwidth share protocol results

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 grow 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 ($1Mb/s$).

5. 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. We introduced two new concepts called inter-node bandwidth share and intra-host regulation. 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 use and traffic regulation.

REFERENCES

[1] ATM-Forum. Traffic Management Specification, version 4.0. ATM Forum/af-tm-0056.000, April 1996.

[2] R. Bell. Virtual private Networks: The Major Issues, problems and Opportunities. Presented at the IEE half-day Colloquium on Virtual Networking, October 1993.

[3] J. Bennet and H. Zhang. Hierarchical Packet Fair Queueing Algorithms. Presented at ACM SIGCOMM'96, August 1996.

[4] B. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. IETF RFC 2205, September 1997.

[5] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. *Journal of Internetworking Research and experience*, pages 3–26, October 1990.

[6] S. Floyd and V. Jacobson. Link-Sharing and Resource Management Models for Packet Networks. *ACM/IEEE Transactions on networking, Vol. 3, No. 4*, pages 365–386, 1995.

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

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

[9] S.J. Golestani. A Self-Clocked Fair Queueing Scheme for Broadband Applications. In *Proceedings of IEEE INFOCOM'94*, pages 636–646, April 1994.

[10] L. Lamti and H. Afifi. A Hierarchical Bandwidth Manager for Local Area Network Outlets. Presented at IEEE Globecom'98, November 1998.

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

[12] A. Parekh and R. Gallager. A Generalized Processor Sharing Approach to Flow Control - The Single Node Case. *ACM/IEEE Transactions on networking Vol. 1, No. 3*, pages 344–357, 1993.

[13] I. Stoica and H. Zhang. A Hierarchical Fair Service Curve Algorithm for Link-Sharing Real-Time and Priority Services. Presented at ACM SIGCOMM'97, September 1997.

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

[15] S. Walters and M. Ahmed. Broadband Virtual Private Networks and their Evolution. Presented at XIV ISS, October 1992.