Scalability and QoS Guarantee in IP Networks

Peifang Zhou and Oliver Yang (corresponding author)

School of Information Technology and Engineering

University of Ottawa

Ottawa, Ontario, Canada K1N 6N5

Tel: (613) 562-5800 ext. 6120

Fax: (613) 562-5175

E-mail: yang@site.uottawa.ca

Abstract

This paper presents our insights to the issue of achieving scalability and providing QoS guarantee in IP networks. The key idea is to use marking at the edge and aggregation in the core of a network. We propose a traffic classification scheme to facilitate marking and aggregation, a queueing architecture to implement marking in the access routers and aggregation in the core routers, and a scheduling algorithm to provide throughput guarantee. Performance analysis is conducted to evaluate our proposed schemes.

1 Introduction

Internet is the most widely used IP network in the world. However, it offers only a single class of best-effort service; that is, there is no admission control and the network offers no assurance about when, or even if, packets will be delivered [13]. With the impending proliferation of voice over IP and other real-time services, the best-effort service causes two serious problems.

First, the best-effort service model is not well suited for real-time applications such as inter-active voice and video. Traditional data applications tolerate packet delays and packet losses so that they are well served by the current best-effort service employed by the Internet. Real-time applications have radically different characteristics and requirements from data applications. The quality of real-time applications is much degraded if the stringent delay requirement can't be met or too many packets are lost. A significant challenge for the next-generation Internet is to offer services for real-time applications in addition to data applications.

Secondly, the Internet is not reliable in the sense that the performance of the best-effort service is mostly unpredictable. The Internet imposes no admission control. Giving everyone unlimited access to a network with limited bandwidth is a recipe for congestion, epitomized by our daily painful experience of "World Wide Wait." The commercial availability of gigabit transport makes the problem of congestion control even more difficult to solve, since any possible solutions can't rely on network-wide feedback, which is too sluggish [7]. New mechanisms of traffic control in gigabit networks are needed to ensure a *timely* and *guaranteed* delivery of packets. The second challenge for the next-generation Internet is to have robust and predictable performance, and to provide quality of service (QoS) guarantee for

both real-time applications and data applications.

There have been tremendous activities in the research community to solve the above two problems and much progress has been made. ATM has been touted as the solution which can provide QoS guarantee [8]. In the Internet Engineering Task Force (IETF), RSVP and Integrated Service Architecture [2] have been proposed to map service requests to ATM. However, both ATM and RSVP suffer from scaling and state constraints which prevent them from supporting a widespread use of connection-oriented services like voice over IP (VOIP) and video-conferencing. To put in a simple term, it is just impractical to provision a large IP network such as the Internet to handle a separate ATM virtual circuit, or its state equivalent in RSVP, for each of millions of simultaneous flows. A flow is defined as composing of a stream of packets that travel through the same route from the source to the destination, and that require the same grade of transmission service [16].

In this paper, we present our insights to the scalability and QoS problems. To achieve scalability, we use marking of IP flows at the edge and aggregation of multiple IP flows in the core of large IP networks. It will be shown that marking and aggregation are effective techniques to enable scalability. To provide QoS guarantee, we look into the following three pivotal components in traffic management: traffic classification, queueing architecture, and scheduling algorithm.

Our key idea in traffic classification is to define a small number of traffic classes to facilitate marking at the edge and aggregation in the core of IP networks. To implement marking and aggregation, we apply to router design the MPLS paradigm [3] which integrates layer-3 routing with layer-2 switching. We use per-flow queueing [19] for the purpose of marking in access routers, and VC merging [15] for the purpose of aggregation in core routers. We demonstrate that ATM's per-VC queueing architecture [20] is instrumental to both per-

flow queueing and VC-merging. Studies have shown that TCP traffic consists of 90% of the total traffic on the Internet due to proliferation of World Wide Web. To keep TCP sessions from stalling and avoid "World Wide Wait," it is essential to guarantee throughput for IP flows. In this paper, we extend our previous work in ATM networks [21] and propose per-flow scheduling to satisfy the above requirement. Most scheduling algorithms [1, 4, 5, 10, 12, 17] reported in the literature associate a time-stamp with each arriving packet. This type of per-packet approach is cost-prohibitive in high-speed IP networks, rendering these algorithms impractical for implementation. Our scheduling philosophy is per-flow based, not per-packet based. This leads to a significant reduction in terms of computational effort and makes our algorithm feasible for practical implementation in real IP networks.

To provide QoS guarantee, it is imperative to impose admission control at the edge of an network to ensure that the needed resources do not exceed what is available. However, admission control is outside the scope this paper and it won't be presented here. See [18] for detailed discussion on admission control.

The rest of this paper is organized as follows. Section 2 addresses issues of achieving scalability and providing QoS guarantee in IP networks. Section 3 conducts performance analysis on several important QoS parameters. Section 4 concludes the paper.

2 Scalability and QoS guarantee in IP networks

There are three key components in traffic management to enable scalability and provide QoS guarantee in IP networks. Our insight is to use marking at the edge and aggregation in the core of a network. We propose a traffic classification scheme to facilitate marking and aggregation, a queueing architecture to implement marking in the access routers and

aggregation in the core routers, and a scheduling algorithm to provide throughput guarantee.

2.1 Traffic Classification

As discussed in the introduction, the next-generation IP networks have to introduce the concept of QoS and provide levels of service that correspond to different classes of traffic. Real-time interactive voice and video applications impose a stringent delay requirement. Data applications demands throughput guarantee. Routers, the basic building blocks in any IP networks, have to identify different traffic types and handle each of them adequately.

There are two major factors that would influence our choice of classification scheme. First, the Dense Wavelength Division Multiplexing (DWDM) technology will provide an extremely high transport capacity in the order of 10 Tb/s [1] to address the concern of explosive traffic growth, so that we'll have abundant bandwidth available in the backbone. Secondly, both real-time (e.g. voice) and data traffic would be carried by a common public switched Internet. It is observed that data traffic expands exponentially, while voice traffic grows linearly. It is not difficult to draw a conclusion that data traffic would be dominant in the next-generation Internet.

Our key idea in traffic classification is to define a small number of traffic classes to facilitate marking at the edge and aggregation in the core of IP networks. In our classification scheme, we extend our previous work in ATM networks [18] and choose the delay attribute to distinguish real-time and data traffic. The next-generation Internet will provide delay-sensitive class of service and delay-insensitive class of service with QoS guarantee, while keeping the best-effort class of service without QoS guarantee for backward compatibility.

[1] To gauge the capacity of 10 Tb/s, imagine that the entire content of the Library of Congress can be transferred from coast to coast in 20 seconds over a 10 Tb/s link!

In summary, there will three classes of service provided by the future Internet, delay-sensitive, delay-insensitive, and best-effort. This paper is mainly concerned with achieving scalability and providing QoS guarantee for IP flows. To maintain clarity of our presentation, we choose to ignore the best-effort class of traffic in the rest of the paper.

There exist cases, namely traffic associated with routing protocols and traffic related to transactions, in which only a few packets are exchanged (we call them short transfers). One way to handle these short transfers is to treat them as a special flow and reserve resources accordingly. This strategy will work well as long as short transfers compose of a small portion of the total traffic, and the aggregation remains stable [16].

2.2 Queueing Architecture

The Internet uses a distributed scheme in which each IP packet travels hop by hop between source and destination. Streamlining this routing process could significantly boost the Internet performance. Several vendors have proposed to overhaul the legacy router architecture [6, 9, 11, 14]. There is one fundamental concept shared by all these proposals: map the route information to short fixed-length labels, so that next-hop routers can be determined quickly through indexing rather than some type of searching (or matching longest prefixes) [15].

The Internet Engineering Task Force (IETF) recently established a working group on Multi Protocol Label Switching (MPLS), which aims to standardize the label swapping paradigm that integrates layer-3 routing with layer-2 switching [3]. ATM as a layer-2 switching technology is considered in this paper, because of its wide availability and its capability to support QoS. We choose AAL5 as the IP packet encapsulation method, because AAL5 is simple, efficient, powerful in error detection, and it has been widely used in data communications. If MPLS is used, a signaling protocol is required for mapping route information to

ATM VPI/VCI labels, for label distributions, etc. Details of the signaling mechanism are outside the scope of this paper and they won't be presented.

In any large IP networks, routers positioned at different locations have different functionalities. In this paper, we divide routers into two categories: access routers at the edge of a network and core routers in the core of a network. We use per-flow queueing [19] for the purpose of marking in the access routers and VC merging [15] for the purpose of aggregation in the core routers. We demonstrate that ATM's per-VC queueing architecture [20] is instrumental to both per-flow queueing and VC merging.

2.2.1 Per-flow Queueing in Access Routers

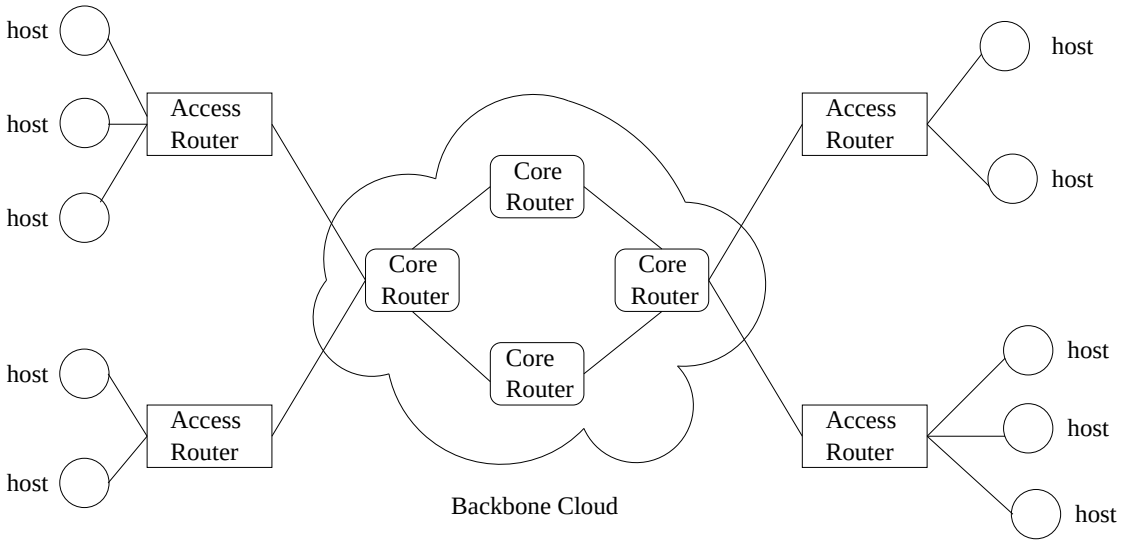

Figure 1: A general IP network with hosts, access routers, and core routers.

Figure 1 illustrates a general topology for an IP network: hosts are attached to access routers, and access routers are connected to a backbone cloud formed by core routers.

Access routers play the role of multiplexing IP traffic from multiple hosts onto the backbone cloud of the network. Under the paradigm of MPLS, access routers handle IP traffic flows utilizing a layer-2 switching mechanism. Since we choose ATM as the layer-2 switching

technology, IP route information will be mapped into ATM VPI/VCI labels. In other words, we mark each IP flow with a separate VC. We advocate the use per-flow queueing [19] as the switching architecture in access routers.

Figure 2 depicts a general per-flow queueing network. The access routers are used to route IP packets from the source endpoint, through the backbone cloud, and to the destination endpoint. We assume that routing tables have been set up in routers. For the clarity of presentation, we temporarily assume that there is no VC merging done by core routers. In the next subsection, we will remove this assumption and explain why VC merging is necessary.

Per-flow queueing is a natural evolution of its ATM counterpart of per-VC queueing [20]. Instead of providing a separate queue for each VC in per-VC queueing, per-flow queueing supplies a separate queue for each IP flow and map an IP flow to an ATM VC if MPLS is used. After a packet from an IP flow is received by the first-hop access router, the packet is stored directly in a queue allocated for the IP flow *without* segmentation. The packet is then read out from the queue, 48 bytes each time, as payloads for multiple ATM cells (padding bytes may be needed in the last cell). As these cells traverse in the core of the network, cell headers steer cells into a dedicated queue associated with the IP flow. At the last-hop access router, each queue only collects cells which belong to the same flow, therefore packets are *automatically* reassembled in the queue. Per-flow queueing creates a virtual pipe for each IP flow and facilitates routers to provide high-speed delivery and QoS guarantee. Details of per-flow queueing architecture in access routers can be found in [19].

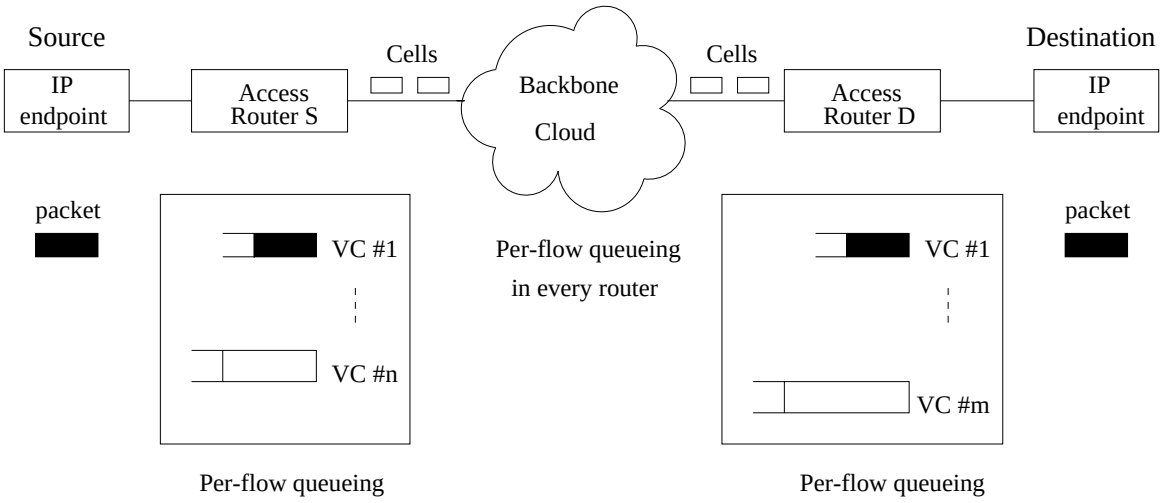

Figure 2: A per-flow queueing network.

2.2.2 VC Merging for Core Routers

In the previous subsection, we temporarily assumed that per-flow queueing was also implemented in core routers and there was no VC merging. In this section, we remove the above assumption and explain why VC merging is necessary.

As mentioned before, ATM is chosen as the layer-2 switching mechanism in MPLS and route information is mapped into ATM VPI/VCI labels. If core routers implement per-flow queueing only, each source-destination pair is mapped to a unique VC value at each core router. The drawback is that it is not scalable because n sources and n destinations would require $O(n^2)$ labels. This is why per-flow queueing without label merging is only suitable for access routers. To support scalability which is a fundamental requirement, core routers have to perform some type of label merging at layer-2. In the context of ATM, VPI/VCI are MPLS labels. There are two basic methods for label merging. The first one is called VP-merging, which maps each destination to a unique VP value. The VC value is used to identify the sender so that packets can be re-assembled correctly at the receiver. However, VP-merging is not practical since the VP space is limited to to only 4096 entries at the

network-to-network interface (NNI). The second method is called VC-merging, which maps incoming VC labels for the same destination to the same outgoing VC label. This method is scalable and does not have the space constraint problem as in VP-merging. However, VC-merging suffers another type of constraint. With VC-merging, cells for the same destination are indistinguishable at the output of a switch. Therefore cells belonging to different packets for the same destination can't be interleaved with each other, or else the receiver won't be able to reassemble packets. Figure 3 illustrates the key difference between VC-merging and non-VC merging [15]. The boundary between two adjacent packets can be identified by the "End-of-Packet" (EOP) indication used by AAL5.

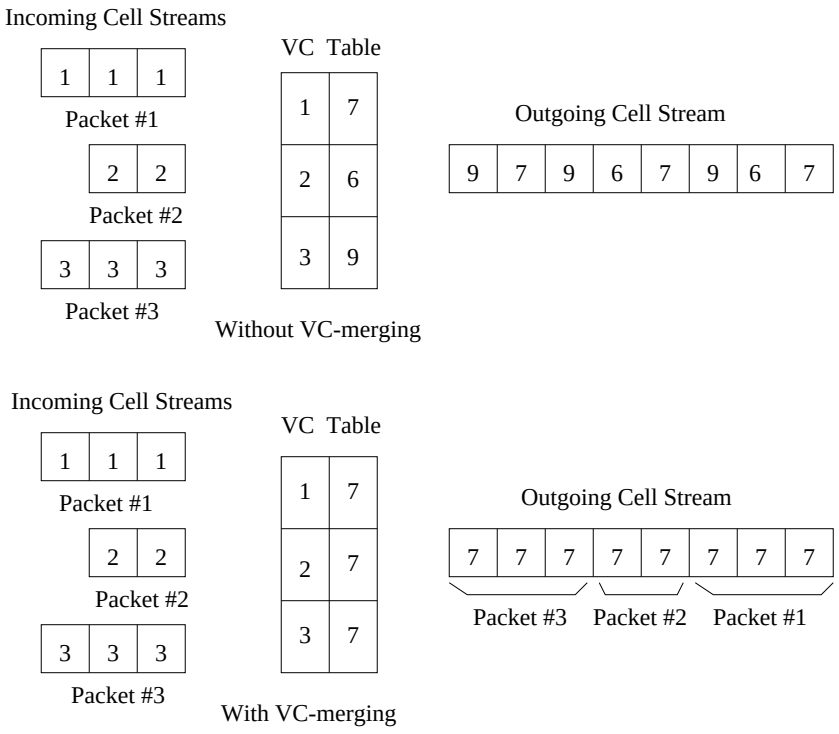

Figure 3: Comparison of non-VC merging and VC merging at an output port.

To establish the feasibility that core routers can be implemented by ATM switches, we have to demonstrate that an ATM switch is capable of merging a group of VCs at the output. A prominent requirement of a VC-merge capable ATM switch is that cells belonging

to different packets should not interleave if they are going to be merged to the same outgoing VC. Therefore, each incoming cell for a given packet on an incoming VC needs to be stored in a separate buffer until the last cell of the packet arrives. From the previous experience in our per-VC queueing architecture [20], we immediately recognize that the requirement of not interleaving cells from different incoming VCs can be *perfectly* met by the per-VC queueing which provides a separate queue for each incoming VC!

The only difference between a per-VC queueing cell switch and a VC-merge capable switch is when scheduling starts. In a conventional per-VC queueing cell switch, there is no such concept as packet and scheduling starts whenever there is a cell available in one of VC queues. In a VC-merge capable switch however, scheduling is deferred until the last cell for a given packet arrives. Afterwards, all cells pertaining to the packet are scheduled in an atomic manner for transmission to the next hop. Recognizing the only difference between the two types of switches, we can make a simple modification to a per-VC queueing cell switch [20] and convert it into a VC-merge capable switch. The only modification required for this conversion is to add a "End-of-Packet" (EOP) detection circuitry to the address list and outputs a "Packet-Ready" signal to the Scheduling module as shown in Figure 4. In other words, an arrival of the last cell of a packet triggers the EOP detection circuitry, which in turn asserts the "Packet-Ready" signal to inform the Scheduling module that a packet is ready for transmission. After all the cells pertaining to the packet have been sent out, EOP detection circuitry de-asserts the "Packet-Ready" signal and waits for the completion of the next packet. The architecture of per-VC queueing with EOP detection is shown in Figure 4.

It is worthy to note that backward compatibility can be maintained for an EOP-equipped VC-merge capable switch to support cell switching. If we consider the special case in which a packet consists of only a single cell, we immediately recognize that in order to support cell

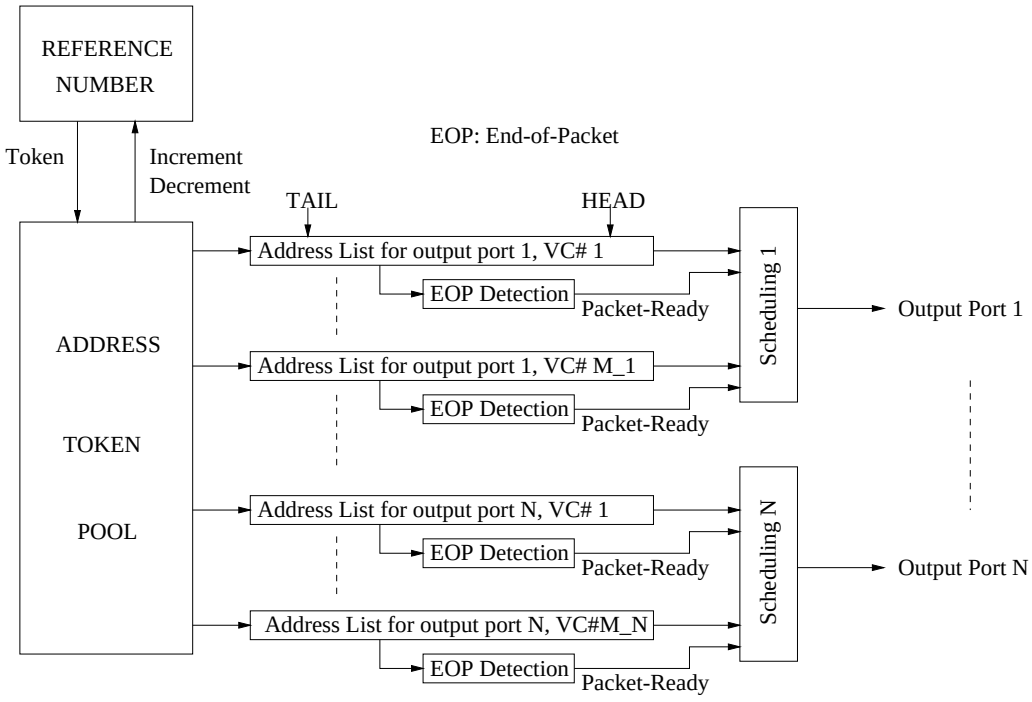

Figure 4: Per-VC queueing with EOP detection to support VC-merging.

switching, we can configure EOP detection circuitry so that it triggers on the arrival of a single cell. In this fashion, we can configure a VC-merge capable switch to support *both* VC merging and pure cell switching.

2.3 Scheduling

The literature is voluminous in the research area of fair scheduling in packet and ATM networks [1, 5, 10, 12, 16]. Mostly all algorithms are based on virtual time-stamp. Although they are efficient in guaranteeing the fairness among traffic flows, the resultant per-packet approach is cost-prohibitive in high-speed Gb/s or Tb/s networks. Moreover, the use of a reference clock implies that it can't be reset to zero until the system is idle.

In the following, we extend our previous work in per-VC queueing scheduler [21] and propose a per-flow scheduler (PFS). PFS eliminates virtual time-stamp and scheduling com-

putation is carried out on a per flow basis, not on a per packet basis. This leads to a significant reduction in the computational effort. PFS has two important attributes. First, it divides outgoing link capacity among competing IP flows on a per-flow basis and provides a *guaranteed* rate to every flow. Secondly, it proportionally allocates any unused bandwidth.

As described before, we mark each packet within an IP flow using an ATM VPI/VCI label in the access routers, and merge labeled IP flows in the core routers. There is a one-to-one mapping between an IP flow and an ATM VC in the access routers. Multiple flows or ATM VCs from access routers are aggregated into a single flow or VC in the core routers. Therefore one flow from an outgoing port of a core router can be considered as superposition of multiple flows from access routers. PFS deals with flow scheduling in every outgoing port of *both* access and core routers.

PFS selects packets for transmission on a frame basis (logically) with N packet slots in a frame. It will be shown later that an introduction of N-slot frame structure guarantees not only a bandwidth, but also a low maximum delay for delay-sensitive flows.

If the link capacity is C_L and flow i requires a bandwidth of bw_i, the slot ration r_i for flow i in a frame is

$$r_i = \frac{bw_i}{C_L} \times N, \quad i \in \{DS, DI\}, \tag{1}$$

where DS designates the set of delay-sensitive flows and DI represents the set of delay-insensitive flows. If the flow i represents an aggregation of multiple IP flows in the core routers, the parameter bw_i will be the sum of bandwidths of these IP flows.

At the beginning of a frame, the scheduler for an outgoing link inspects all VC queues associated with the link, and schedules transmission of packets in each VC queue up to the flow's ration. In the case of VC merging in the core routers, a group of VC queues are

inspected. The delay-sensitive VCs have a higher priority than delay-insensitive VCs. Recall that we classify traffic into three types: delay-sensitive, delay-insensitive, and best effort. For the clarity of presentation, we assume peak-rate allocation for delay-sensitive flows, and average bandwidth allocation for delay-insensitive flows. Unused bandwidth left by delay-sensitive flows may be allocated to best-effort traffic. As mentioned before, best-effort traffic is ignored in the rest of this paper in order to maintain clarity of our presentation.

Since we use peak-rate allocation for delay-sensitive flows, packets in delay-sensitive VC queues never exceed their rations. Let n_j be the number of packets in the VC queue for flow j at the beginning of a frame, r_j be the slot ration for flow j, DS represent the set of delay-sensitive flows, we have

$$n_j \leq r_j, \quad j \in DS. \tag{2}$$

For delay-insensitive flows, the numbers of packets at the beginning of a frame can be larger than their rations. We divide the set of delay-insensitive flows DI into two subsets: DIU and DIO. DIU designates the set of delay-insensitive flows for which the numbers of packets at the beginning of a frame are under their rations, i.e.,

$$DIU = \{\text{delay-insensitive flow } k,\ n_k \leq r_k\}. \tag{3}$$

DIO represents the set of delay-insensitive flows for which the numbers of packets at the beginning of a frame are over their rations, i.e.,

$$DIO = \{\text{delay-insensitive flow } l,\ n_l > r_l\}. \tag{4}$$

From the scheduling point of view, some slots may not be assigned to any flows, and not

all slots allocated to flows in DS and DIU are utilized. We can calculate extra slots es as follows:

$$es = (N - \sum_{i} r_i) + \sum_{j_i} (r_{j_i} - n_{j_i}) + \sum_{k_i} (r_{k_i} - n_{k_i}), \quad i \in \{DS, DI\}, \;\; j_i \in DS, \;\; k_i \in DIU. \tag{5}$$

To achieve multiplexing gain among flows, we need to re-allocate these extra slots left by flows in the sets DS and DIU, and use them to dispatch packets which belong to flows in the set DIO. In the mean time, flows in DIU should be credited for "freeing up" slots which allows the scheduler to transmit packets from DIO flows. DIU flows should be compensated by being allowed to transmit more packets than rations later on. Data traffic are bursty. When a burst arrives, a flow may experience the transition from the set DIU to DIO, and it is fair for the flow to claim previously unused rations to transmit more packets and get the burst quickly out of the VC queue. In other words, the scheduler should schedule more packet transmissions for flows which haven't used up their rations. This should be viewed as part of scheduler's commitment of providing average bandwidth to every delay-insensitive flow. For this purpose, we introduce a *grant* variable, g_i for flow i, to keep track of unused ration on a per-flow basis. The value of g_i is non-negative. It is initialized as 0 (zero). It is incremented when fewer packets than flow i's ration are scheduled at the beginning of a frame, i.e., the updated value $\hat{g}_k$ is

$$\hat{g}_k = g_k + (r_k - n_k), \quad n_k < r_k, \tag{6}$$

and it is decremented when more packets (up to g_k) than flow i's ration are scheduled during a frame,

$$\hat{g}_k = g_k - m_k, \quad m_k < g_k, \tag{7}$$

where m_k is the number of packets transmitted over flow k's ration.

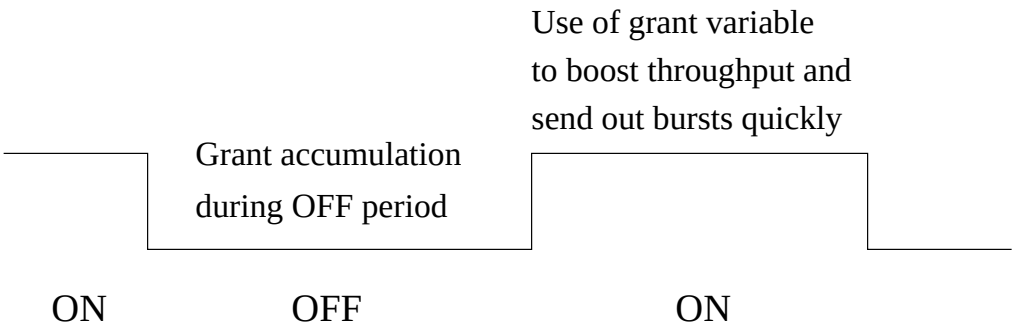

Figure 5: Use of *grant* variable to boost instantaneous throughput.

The use of *grant* variable is a very desirable feature for bursty traffic, because it boosts instantaneous throughput when a burst of packets arrives. Delay-insensitive traffic tend to be bursty and the so-called ON/OFF model is often used to represent bursty traffic. During OFF periods, there is no packet arrival and *grant* is accumulated to record unused rations. During ON periods when bursts of packets arrive, the *grant* variable authorizes the scheduler to send out more packets than what the ration allows, effectively boosting instantaneous throughput to drain bursts out of the queue much more quickly. This scenario is illustrated in Figure 5.

Per-flow scheduler (PFS) calculates extra slots available in each frame, and selects DIO flows which have *grants* available for transmitting more packets than their rations. The selection is based on the *grant* variable. The larger value of *grant*, the higher priority a flow has. If *grants* are used up for all flows but extra slots are still not exhausted, the scheduler will select flows in the set DIO for packet transmission. Based on the "pay per bandwidth" billing policy that end users pay more for higher bandwidth, the scheduler will distribute extra slots based on the bandwidth requirement. The higher bandwidth a flow requires, the higher priority the flow has. Here we need to introduce another *absorption*

variable to keep track of extra bandwidth consumption on a per-flow basis. The *absorption* variable, a_i for flow i, is non-negative. It starts at 0 (zero), and it is incremented when a flow takes advantage of extra slots available in a frame to transmit more packets. The smaller value of *absorption*, the higher priority a flow has in consuming any extra slots. To maintain fairness among flows which have required different bandwidths and to reflect tariff imposed on different rates, any extra bandwidth left by flows in the sets DS and DIU should be distributed in proportion to bandwidth requirements for flows in the set DIO. In other words, extra slot usage needs to be normalized by the bandwidth requirement. If m_l extra slots are used by flow l, its absorption variable a_l should be updated as follows:

$$\hat{a}_l = a_l + K \times \frac{m_l}{bw_l}, \quad l \in DIO, \tag{8}$$

where K is a positive constant, bw_l is the bandwidth requirement of flow l, and $\hat{a}_l$ is the updated value of a_l.

In summary, per-flow scheduler (PFS) schedules packet transmission of both delay-sensitive and delay-insensitive flows on a frame basis. For delay-sensitive flows, enough slots are reserved to accommodate peak-rate allocation. Any unused slots will be *immediately* exploited to serve other flows. For delay-insensitive flows, PFS first selects packets under flows' rations. It then allocates any extra slots available to schedule packet transmission for flows that have unused rations. Finally, PFS checks the *absorption* variables to dispatch more packets than flows' rations. PFS operates on the packet level to enforce bandwidth requirement on the flow level. In essence, bandwidth is guaranteed for *all* flows, i.e., both delay-sensitive and delay-insensitive flows. For a delay-insensitive flow, it could achieve a higher throughput than the average bandwidth it requested during the flow's setup time.

This feature is well suited to bursty data traffic, since it provides a fast draining mechanism when bursts arrive.

Similar to arguments presented in [21], PFS is inherently stable and the minimum rate supported by PFS is not limited to C_L/N, the ratio between the link capacity and the number of packet slots in a frame.

3 Performance Analysis

In this section, we conduct performance analysis focusing on the most important QoS parameters: delay, delay jitter, throughput, and buffer requirement. We'll show that QoS guarantee applies not only to a single IP flow, but also to an aggregated IP flow which is merged from multiple IP flows in the core routers.

3.1 Delay and Delay Jitter Bounds

In a previous section, we proposed a logical frame structure of N packet slots over a link with capacity C_L. The maximum frame duration is denoted as T_{max}. Now we derive the end-to-end delay bound for delay-sensitive flows, assuming the same N-slot frame structure and link capacity C_L within the network. The end-to-end delay is defined as the time elapsed between the arrival of a packet at the first-hop access router (Access Router S in Fig. 2) and the receipt of the packet at the last-hop access router (Access Router D in Fig. 2).

First we obtain the delay bound introduced by the first core router within the backbone cloud shown in Fig. 1. Since we use VC merging in the core routers, multiple IP flows from access routers are aggregated into a single flow coming out of the first core router. We assume that k flows are merged in the first core router and the peak-rate summation of these

k flows is less than C_L/N, i.e.,

$$\sum_{i=1}^{k} pr_i \leq \frac{C_L}{N}, \tag{9}$$

where pr_i is the peak rate for flow i. Due to VC merging, we arrange packets from these k flows to share the same slot position in consecutive frames leaving out of the first core router.

There are two components for the delay encountered by a packet entering the first core router of a backbone cloud: the waiting time to transmit the packet, and the service time of the packet. We first calculate the waiting time. In the worst case, one packet from each of the k flows arrives at the same time and misses the start of a frame. The packets from k flows have to wait for the start of the next frame. We identify the packet from one flow as "tagged". This "tagged" packet has to wait for the finish of transmission of the packets from other $(k-1)$ flows, and therefore the "tagged" packet experiences the longest waiting time, which is

$$d_w \leq T_{max} + (k-1)T_{max} + T_{max} = (k+1)T_{max}. \tag{10}$$

The first term in the above equation represents the upper bound of waiting time for the start of a new frame, the second term is the time elapsed to transmit $(k-1)$ packets in the same slot position, and the last term is the upper bound between the start of frame and the start of transmission of the "tagged" packet.

Now we calculate the second component of the delay which is the packet transmission time. Since we have an N-slot frame and we arrange k flows to share the same slot in consecutive frames, the bandwidth perceived by each of the k flows is $\frac{1}{k} \times \frac{C_L}{N}$. Let l_{max} and l_{min} denote the maximum length and minimum length of packets. The packet transmission

time m is

$$m \leq l_{max}/(\frac{1}{k} \times \frac{C_L}{N}) = l_{max} \times \frac{kN}{C_L}. \tag{11}$$

Combining the above two terms, we can express the delay d_{1c} at the first core router as

$$d_{1c} = d_w + m \leq (k+1)T_{max} + l_{max} \times \frac{kN}{C_L}. \tag{12}$$

Since the above k flows are merged into a single flow leaving out of the first core router and going into other core routers. Reusing Eqn. (10), we have delays introduced at the other core routers with a maximum of $2T_{max}$ each. If there are h hops between the first core router and the last core router, the maximum delay introduced by the switching routers other than the first one will be $h \times 2T_{max}$. The delay introduced by all core routers is given by

$$d_c = d_{1c} + h \times 2T_{max} \leq (k+1)T_{max} + l_{max} \times \frac{kN}{C_L} + h \times 2T_{max}. \tag{13}$$

In a similar fashion, we can obtain the delay introduced by the first-hop and the last-hop access routers:

$$d_a \leq 2T_{max} + 2T_{max} = 4T_{max}. \tag{14}$$

Adding the the propagation delay $prop$, we will attain the end-to-end delay d,

$$d = prop + d_a + d_c \leq prop + (k+1)T_{max} + l_{max} \times \frac{kN}{C_L} + (h+4) \times 2T_{max}. \tag{15}$$

Based on the above analysis and derivation, we can easily obtain the upper bound and lower

bound of the end-to-end delay d:

$$d_{max} \leq prop + (k+1)T_{max} + l_{max} \times \frac{kN}{C_L} + (h+4) \times 2T_{max} \tag{16}$$

and

$$d_{min} \geq prop + l_{min} \times \frac{kN}{C_L}. \tag{17}$$

The delay jitter dj is defined as the difference between d_{max} and d_{min}. The upper bound for delay jitter can be derived from Eqn. (16) and Eqn. (17) as follows:

$$\begin{aligned} dj &= d_{max} - d_{min} \\ &\leq [prop + (k+1)T_{max} + l_{max} \times \frac{kN}{C_L} + (h+4) \times 2T_{max}] - [prop + l_{min} \times \frac{kN}{C_L}] \\ &\leq (k+1)T_{max} + l_{max} \times \frac{kN}{C_L} + (h+4) \times 2T_{max}. \end{aligned} \tag{18}$$

It is straightforward to show that the propagation delay across a wide area (e.g. 50 ms across continental USA) is the dominant factor in the end-to-end delay over a Gb/s or Tb/s network. A significant advantage of introducing an N-slot frame structure is to ensure a low maximum delay for high-speed transport.

If the combined rate of aggregated flows is so large that at least two packets have to be forwarded in a frame, we can decompose the flow into sub-flows, such that we can use the above derivation to obtain similar results.

3.2 Throughput

Per-flow scheduler (PFS) has attractive properties and they can be expressed by the following lemmas and theorem.

Lemma 1: PFS provides bandwidth guarantee for every flow.

Proof: For an outgoing link with capacity C_L and N (logical) slots in a frame, we allocate slot ration r_i for flow i according to Eqn. (1):

$$r_i = \frac{bw_i}{C_L} \times N, \quad i \in \{DS, DI\}. \tag{19}$$

The bandwidth allocated by PFS to flow i is

$$\begin{aligned} bw &= \frac{r_i}{N} \times C_L \\ &= \frac{bw_i}{C_L} \times N \times \frac{1}{N} \times C_L \\ &= bw_i \qquad\qquad \textbf{Q.E.D.} \end{aligned}$$

Lemma 2: PFS is fair in the sense that it allocates any extra bandwidth in proportion to requested average rates of delay-insensitive flows.

Proof: We use the *absorption* variable to keep track of extra slot usage. If there are any extra slots left by flows in the sets DS and DIU, PFS sorts out flows in the set DIO according to the *absorption* variable. Flows with less *absorption* values have a higher priority to use extra slots to transmit more packets. After packet transmissions, a flow's *absorption* variable is increased and the flow has less chance to use any more slots. In the long run, the differences among *absorption* variables are much smaller than variables themselves. Let I and J represent extra slots consumed by delay-insensitive flows i and j during the interval $(0, T_l)$. Ignoring the small difference between *absorption* variables a_i and a_j, we have

$$a_i = a_j,$$

$$K \times \frac{I}{bw_i} = K \times \frac{J}{bw_j},$$
$$\frac{I/T_l}{J/T_l} = \frac{bw_i}{bw_j}. \qquad \textbf{Q.E.D.}$$

Theorem: For delay-insensitive flow i, the extra bandwidth received beyond the requested average bandwidth bw_i is

$$\frac{bw_i}{\sum_i bw_i} \times (C_L - \sum_i bw_i - \sum_j bw_j), \quad i \in DI, \;\; j \in DS.$$

Proof: From **Lemma 2**, any bandwidth capacity left, which is $C_L - \sum_i bw_i - \sum_j bw_j, \;\; i \in DI, \;\; j \in DS$, will be distributed in proportion to delay-insensitive flows' required average rates. For delay-insensitive flow i which has requested an average bandwidth bw_i, its portion in total required average bandwidth is $bw_i / \sum_i bw_i$, therefore flow i can benefit from the following extra bandwidth beyond bw_i:

$$extra \;\; bandwidth = \frac{bw_i}{\sum_i bw_i} \times (C_L - \sum_i bw_i - \sum_j bw_j), \quad i \in DI, \;\; j \in DS. \qquad \textbf{Q.E.D.}$$

3.3 Buffer occupancy

In this subsection, we analyze the buffer requirement for VC merging in a VC-merge capable switch (core router). Recall that IP packets are segmented into ATM cells in the access routers and VC merging is performed in the core routers.

We use a discrete-time model and assume that the operation of transmission links are time-slotted with each time slot being the transmission time of an ATM cell. The switch size is $N \times N$. The arrival processes on the input ports are assumed to be bursty. To represent burstiness of the incoming traffic, we use the so-called ON/OFF model. The traffic on each

input port is assumed to have two states: an ON state in which a cell arrives in every slot, and an OFF state in which no cell arrives. Each input port alternates between these ON and OFF states. Let r denote the state transition probability from ON to OFF state, and s from OFF to ON state. The average cell arrival rate, or the offered load ρ, of each input port is $(1/r)/(1/r + 1/s) = s/(r + s)$. It is assumed that cells within an ON period form a single packet.

To analyze the buffer requirement for VC merging, we need to characterize the packet arrival process from all input ports. As reasoned before, packet transmission to an output port can start only after the last cell of the packet is received because of the non-interleaving requirement of VC merging. In other words, a batch of cells received during the ON period have to be stored in the buffer before being sent out to an output port. It is easy to see that buffer occupancy is affected by these batches of cells. Here comes our key observation: the ON-OFF bursty arrival process can be treated as a bulk arrival process with the bulk size being equal to the corresponding ON period and the bulk interarrival time being equal to the sum of the OFF and ON periods. Embedded Markov chain analysis can be applied to the above bulk arrival process. However, an exact construction of the Markov chain requires an infinite number of states. To simplify analysis, we approximate the bulk arrival process by a bulk Bernoulli arrival process, i.e., in an any given time slot, the probability that a bulk arrives on a particular input port is p. The probability p can be estimated by the occurrences of a new start of ON period, which can be obtained as:

$$p = \frac{1}{\frac{1}{r} + \frac{1}{s}} = \frac{rs}{r + s}. \tag{20}$$

The bulk size g is equal to the ON period and therefore it follows a geometric distribution

with the following probability generating function (PGF):

$$G(z) = \frac{rz}{1-(1-r)z}. \tag{21}$$

We assume that the buffer size is infinite. Since the central buffer is shared among all incoming traffic, the destination distribution has an influence on queue length distribution of output queues, but it does not affect the statistics of overall buffer occupancy. Without the loss of generality, we assume that each incoming bulk has equal probability $1/N$ of being addressed to any given output port.

Now we use a similar analysis approach as in [20] to derive the queue length distribution for each output port. Fixing our attention on a particular output queue (the "tagged" queue), we define the random variable v as the number of bulk arrivals at the tagged queue during a given time slot. It follows that v has a binomial distribution with PGF:

$$V(z) = \sum_{i=0}^{N} z^i Pr[v=i] = (1-p/N+zp/N)^N. \tag{22}$$

Letting a be the number of cell arrivals during a given time slot, we can obtain its PGF $A(z)$ from the above $V(z)$:

$$A(z) = V(G(z)) = (1-p/N+G(z)p/N)^N, \tag{23}$$

where $G(z)$ is the PGF of bulk size distribution which is expressed in Eqn. (21).

Letting q_m be the number of cells in an output queue at the end of mth time slot, and

a_m be the the number of cell arrivals during the mth time slot, we have

$$q_m = \max(0, \quad q_{m-1} + a_m - 1). \tag{24}$$

Using the same technique as in [20], we obtain the PGF for the steady-state queue size:

$$Q(z) = \frac{(1-\rho)(1-z)}{(1-p/N+G(z)p/N)^N - z}. \tag{25}$$

Differentiating Eqn. (25) with respect to z and taking the limit as $z \to 1$, we can obtain the mean $\overline{q}$ and variance σ_q^2 of steady-state queue size.

As derived in [20], the minimum buffer size B_{min} to achieve a desired cell loss probability 10^{-k} is governed by

$$B_{min} = u_k\sqrt{N}\sigma_q + N\overline{q}, \tag{26}$$

where the values of $u_k, k = 1, 2, \ldots$, can be found in [20].

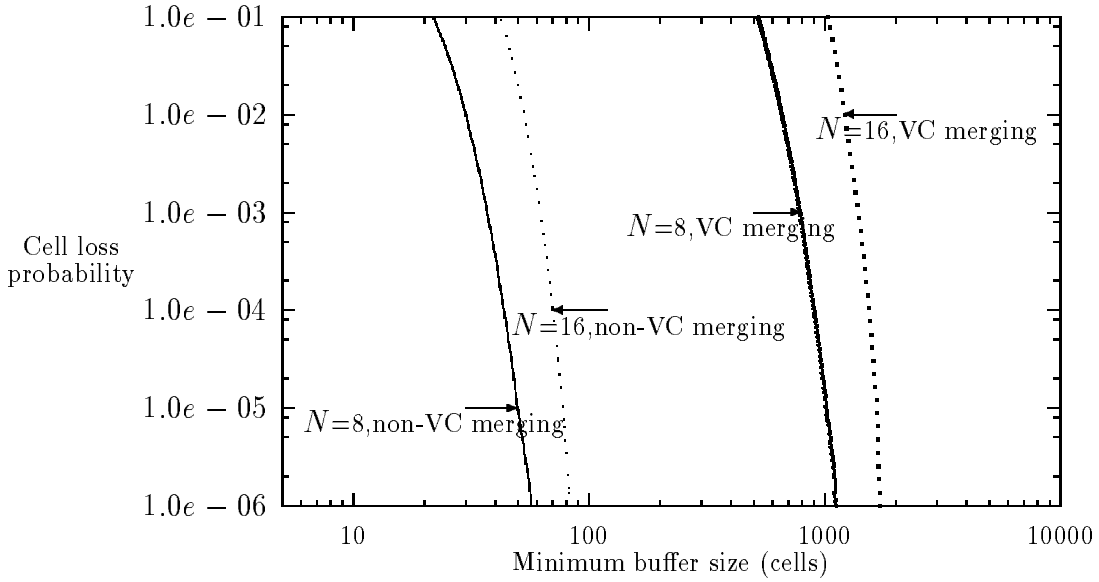

Figure 6: Comparison of cell loss probabilities between VC merging and non-VC merging.

Figure 6 compares the performance of cell loss probability between per-flow queueing (non-VC merging) [20] and VC merging for switch sizes $N = 8$ and 16 under bursty traffic. We select parameters s and r of the ON/OFF model such that they represent both bursty traffic source and heavily offered load. The state transition probability r from ON to OFF state is chosen to be 0.1, and the state transition probability s from OFF to ON state is chosen to be 0.5. The average burst length $1/r$ is 10 cells, and the average idle period $1/s$ is 2 cells. The offered load $\rho = (1/r)/(1/r + 1/s)$ is 0.83, which is high. We can see that the above traffic is quite bursty and heavily loaded. Figure 6 clearly indicates that VC merging requires a significant increase in buffer size in order to achieve the same cell loss probability as non-VC merging. The main reason of this excessiveness is due to the fact that it is not work-conserving to do VC merging in a switch (core router). As reasoned before, packet transmission can't start until the last cell of the packet is received. Therefore there are periods in which no packet is transmitted while cells keep coming into the buffer. This will eventually lead to a huge backlog if incoming traffic is heavy and bursty, which is exactly the case in Figure 6. In summary, buffer requirement can be excessive for VC merging when compared to non-VC merging. The findings presented here are in contrast with a previous investigation [15].

4 Concluding Remarks

This paper presented our insights to the issue of achieving scalability and providing QoS guarantee in IP networks. The key idea was to use marking at the edge and aggregation in the core of a network. We proposed a traffic classification scheme to facilitate marking

and aggregation, a queueing architecture to implement marking in the access routers and aggregation in the core routers, and a scheduling algorithm to provide throughput guarantee. Performance analysis was conducted to evaluate our proposed schemes.

References

[1] J. Bennett and H. Zhang, "WF2Q: worst-case fair weight fair queueing,", in *Proc. IEEE INFOCOM'96*, San Francisco, CA, Mar. 1996, pp. 120-128.

[2] R. Braden, D. Clark, and S. Shenker, "Integrated services in the Internet architecture: an overview," *RFC 1633*, Jun. 1994.

[3] R. Callon, P. Doolan, N. Feldman, A. Fredette, G. Swallow, and A. Viswanathan, "A framework for Multiprotocol Label Switching," Internet draft, *draft-ietf-mpls-framework-00.txt,* May 1997.

[4] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *Proc. ACM SIGCOMM'89*, Austin, TX, Sept. 1989, pp. 1-12.

[5] S. Golestani, "A self-clocked fair queueing scheme for broadband applications," in *Proc. IEEE INFOCOM'94*, Toronto, Ontario, Canada, Jun. 1994, pp. 636-646.

[6] Y. Katsube, K. Nagami, and H. Esaki, "Toshiba's router architecture extensions for ATM: overview," *RFC 2098*, Toshiba R & D Center, Feb. 1997.

[7] L. Kleinrock, "The latency/bandwidth tradeoff in gigabit networks," *IEEE Commun. Magazine*, vol. 30, no. 4, pp. 36-40, Apr. 1992.

[8] D. McDysan and D. Spohn, *ATM theory and application*, 2nd edition. New York, NY: McGraw-Hill, 1998.

[9] P. Newman, T. Lyon, and G. Minshell, "Flow labelled IP: a connectionless approach to ATM," in *Proc. IEEE INFOCOM'96*, San Francisco, CA, Mar. 1996, pp. 1251-1260.

[10] A. Parekh and R. Gallager, "A generalized processor sharing approach to flow control in integrated services networks-the single node case," in *Proc. IEEE INFOCOM'92*, Florence, Italy, May 1992, pp. 915-924.

[11] Y. Rekhter, B. Davie, D. Katz, E. Rosen, and G. Swallow, "Cisco Systems' tag switching architecture overview," *RFC 2105*, Cisco Systems Inc., Feb. 1997.

[12] J. Roberts, "Virtual spacing for flexible traffic control," *Int. J. of Commun. Systems*, vol. 7, pp. 307-318, 1994.

[13] S. Shenker, "Fundamental design issues for the future Internet," *IEEE J. Select. Areas Commun.*, vol. SAC-13, no. 7, pp. 1176-1188, Sept. 1995.

[14] A. Viswanathan, N. Feldman, R. Boivie, and R. Woundy, "ARIS: aggregate route-based IP switching," Internet Draft, *draft-viswanathan-aris-overview-00.txt*, IBM, Mar. 1997.

[15] I. Widjaja and A. Elwalid, "Performance issues in VC-merge capable switches for IP over ATM networks," in *Proc. IEEE INFOCOM'98*, San Francisco, CA, Mar. 1998, pp. 372-380.

[16] L. Zhang, "Designing a new architecture for packet switching communication networks," *IEEE Commun. Magazine*, vol. 25, no. 9, pp. 5-12, Sept. 1987.

[17] L. Zhang, "VirtualClock: a new traffic control algorithm for packet switching networks," in *Proc. ACM SIGCOMM'90*, Philadelphia, PA, Sept. 1990, pp. 19-29.

[18] P. Zhou and O. Yang, "Traffic control in high-speed ATM networks," CCNR technical report, Dept. ECE, U. of Ottawa, Ottawa, Ontario, Canada, Oct. 1996. An extended version was presented in *Proc. IEEE IC3N'98*, Lafayette, LA, Oct. 1998, pp. 183-190.

[19] P. Zhou and O. Yang, "Integrated cell and frame switching in ATM networks," CCNR technical report, Dept. ECE, U. of Ottawa, Ottawa, Ontario, Canada, Oct. 1997. Also in *IEEE Commun. Lets.*, vol. 3, no. 6, pp. 183-184, Jun. 1999.

[20] P. Zhou and O. Yang, "Design of per-VC queueing ATM switches," in *Proc. IEEE ICC'98*, Atlanta, GA, Jun. 1998, pp. 304-308.

[21] P. Zhou and O. Yang, "A feasible scheduling algorithm for per-VC queueing ATM switches," in *Proc. IEEE ICATM'99*, Colmar, France, Jun. 1999, pp. 295-304.

[22] P. Zhou and O. Yang, "Scalability and QoS guarantee in IP networks," in *Proc. IEEE IC3N'99*, Boston, MA, Oct. 1999, pp. 427-433.