Telecommunication Systems 17:1,2, 135–160, 2001

A Per-Flow Based Node Architecture for Integrated Services Packet Networks

DAPENG WU
Carnegie Mellon University, Pittsburgh, PA, USA

YIWEI THOMAS HOU [*] thou@fla.fujitsu.com
Fujitsu Laboratories of America, 595 Lawrence Expressway, Sunnyvale, CA 94085, USA

BO LI
Hong Kong University of Science and Technology, Kowloon, Hong Kong

H. JONATHAN CHAO
Polytechnic University, Brooklyn, NY, USA

Received 1 November 1999; Revised 1 June 2000

Abstract. As the Internet transforms from the traditional best-effort service network into QoS-capable multi-service network, it is essential to have new architectural design and appropriate traffic control algorithms in place. This paper presents a network node architecture and several traffic management mechanisms that are capable of achieving QoS provisioning for the guaranteed service (GS), the controlled-load (CL) service, and the best-effort (BE) service for future integrated services networks. A key feature of our architecture is that it resolves the out-of-sequence problem associated with the traditional design. We also propose two novel packet discarding mechanisms called selective pushout ($\mathcal{SP}$) and selective pushout plus ($\mathcal{SP}+$). Simulation results show that, once admitted into the network, our architecture and traffic management algorithms provide, under all conditions, hard performance guarantees to GS flows and consistent (or soft) performance guarantees to CL flows, respectively; minimal negative impact to in-profile GS, CL and BE traffic should there be any out-of-profile behavior from some CL flows.

Keywords: integrated services (IntServ) networks, per-flow queueing, guaranteed service (GS), controlled-load (CL) service, best-effort (BE) service, shaped virtual clock (SVC), weighted round robin (WRR), deficit round robin (DRR), call admission control (CAC), packet discarding, quality of service (QoS)

1. Introduction

One of the most challenging problems for the next generation Internet is to support diverse multimedia applications with quality of service (QoS) guarantees. To address this issue, the IETF integrated services (IntServ) working group has specified the *guaranteed service* (GS) [6], the *controlled-load service* (CL) [9], in addition to the traditional *best-effort* (BE) service. The GS guarantees that every packet will arrive within the guar-

[*] Corresponding author.

anteed delivery time, and will not be discarded due to buffer overflow, provided that the flow's traffic conforms to its specified traffic parameters. The CL service is intended to support a broad class of applications which have been developed for use in today's best-effort Internet, but are sensitive to heavy load conditions. The CL service does not specify any target QoS parameters. Instead, acceptance of a request for CL service implies a commitment by the network to provide the requester with a service *closely approximating* the QoS the same flow would receive under lightly loaded conditions. Finally, the BE service will continue to be a major service under IntServ networks.

To support the diverse QoS requirements from IntServ networks, new network architecture and traffic management algorithms must be in place. Such architecture and algorithms must meet the following performance evaluation criteria.

Criterion.

(C1) For the GS, the IETF requires that the architecture and algorithms must ensure that the end-to-end delay bounds are never violated and packets are not lost if a source's traffic conforms to its traffic profile [6].

(C2) For the CL service, an architecture and algorithms should provide a flow, under all load conditions, with a QoS closely similar to the QoS that the same flow would receive under lightly loaded network conditions [9].

(C3) The network architecture and traffic managements algorithms must be capable of controlling non-conforming GS/CL flows by minimizing their negative impact on other conforming GS/CL flows and BE flows [6,9].

Previous work on IntServ networks (e.g., [3,5]) has been focused on class-based queueing architecture. However, when class-based approach is used to support CL service, the following problems may arise.

Problem.

(P1) First of all, it is not clear, under class-based approach, how to effectively isolate out-of-profile packets and minimize their negative impact on other in-profile GS and CL packets (i.e., C3).

(P2) Second, under class-based approach, in-profile GS/CL packets have higher priority and are serviced before any out-of-profile packet. This may result in out-of-sequence problem for packets arriving at the destination, which is undesirable for real-time applications.

(P3) It is impossible for a class-based approach to enforce fair rate allocation among CL flows.

Recent market demand on high performance switches and routers has put QoS support as the key feature in differentiating networking products among various vendors. Furthermore, due to advances in silicon technology, hardware implementation of sophisticated per-flow based traffic management algorithms no longer poses any major cost constraint [1]. Such market demand and hardware capabilities enable us to design per-flow based traffic management mechanisms to control quality of service with much more improved performance than traditional class-based approach for the next generation switches and routers. This paper presents a novel architecture and several traffic management algorithms based on per-flow queueing that not only satisfy the three criteria to support integrated traffic of the GS, the CL, and the BE services, but also resolve the several problems associated with the traditional class-based approach.

Our network architecture strives to offer a good balance between traffic isolation and buffer sharing. We make three separate buffer partitions for the GS, the CL, and the BE flows. Per flow queueing with shaped virtual clock (SVC) scheduling is employed for GS and CL flows while per-flow queueing with deficit round robin (DRR) is employed for BE flows. We introduce an *Adaptive Rate allocation for Controlled-load* (ARC) algorithm to provide soft bandwidth allocation to CL flows while enforcing a guaranteed rate allocation to each GS flow. We present a hybrid call admission control (CAC) algorithm consisting of model-based CAC for GS flows and measurement-based CAC for CL flows. Finally, we design two packet discarding algorithms, called *selective pushout* ($\mathcal{SP}$) for GS flows and *selective pushout plus* ($\mathcal{SP}+$) for CL flows, respectively, to control out-of-profile packets and achieve fairness among the flows. Our simulation results show that once admitted into the network, our architecture offers hard performance guarantees to GS flows under all conditions (C1), consistent performance to CL flows under both light and heavy load conditions (C2), and minimal negative impact on conforming flows should there be any misbehaving flows (C3). Furthermore, our architecture and traffic management algorithms resolve the three problems associated with the traditional class-based approach.

The remainder of this paper is organized as follows. Section 2 presents our network node architecture for supporting the GS, the CL, and the BE services. In section 3, we present our traffic management algorithms. Section 4 shows the simulation results. Section 5 concludes this paper.

2. Architecture

This section presents our network node architecture using per-flow queueing for supporting QoS provisioning in IntServ networks.

In our model, an IntServ network is constructed by interconnecting switches or routers with a set of links. We assume that each switch employs output port buffering. Figure 1 shows our architecture for the GS, the CL, and the BE traffic at each output port of a network node. Under our architecture (figure 1), we partition each output port buffer pool into three parts: one for GS flows, one for CL flows, and one for BE traffic. Within the same buffer partition for GS/CL/BE flows, we employ per-flow queueing for each

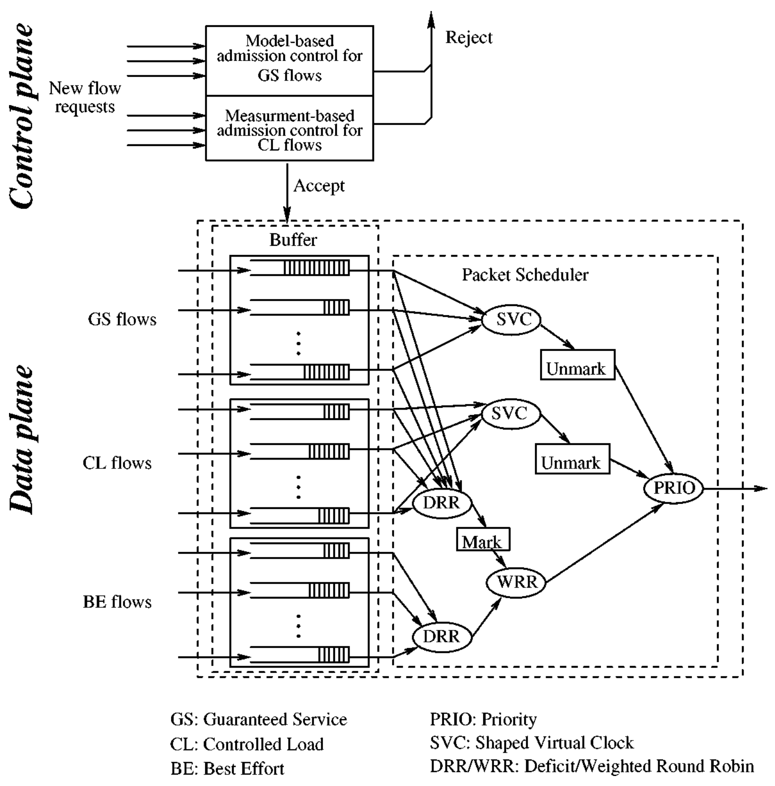


Figure 1. A per-flow based node architecture.

individual flow.[1] Furthermore, flows within the same buffer partition share the buffer pool of that partition while there is no buffer sharing across partitions. We believe this approach offers an excellent balance between traffic isolation and buffer sharing.

Under the above buffering architecture, we design our per-flow based traffic management algorithms with the aim of achieving the three criteria for the GS, the CL, and the BE services and solve the several problems associated with the class-based approach. For clarity, we briefly summarize our traffic management algorithms here, all of which will be elaborated in section 3.

The first part of our architecture includes rate and buffer allocation, as well as packet scheduling, which we will present in detail in section 3.1. For GS traffic, we employ a simple calculation to allocate rate and buffer requirements, which provides a deterministic QoS guarantee (i.e., hard delay bound for each packet and zero packet loss rate) for each flow. On the other hand, for CL flows, we can choose a much less conservative approach, since it only requires soft QoS guarantees. We show how to estimate the effective bandwidth of a CL flow by measuring the entropy of such flow. To support

[1] We employ per-flow queueing for BE traffic since it has been shown that TCP applications can achieve better performance under per-flow queueing than those under a common FIFO shared queue [8].

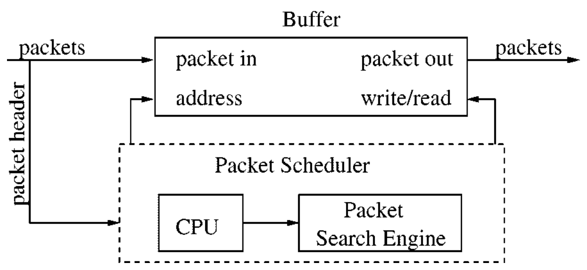


Figure 2. Packet processing at a node.

the link sharing between the GS and the CL flows, we present a rate assignment strategy called ARC (short for Adaptive Rate allocation for Controlled-load) to provide hard bandwidth guarantee to GS flows under all conditions and consistent (or soft) bandwidth allocation to CL flows.

Also shown in figure 1 is a hierarchical packet scheduling architecture where a priority link scheduler is shared among a shaped virtual clock (SVC) for GS flows, a second SVC for CL flows, and a weighted round robin (WRR) for aggregate traffic from BE flows and out-of-profile GS/CL packets,[2] where BE flows and out-of-profile GS/CL packets are each serviced by a deficit round robin (DRR) scheduler, respectively. Service priority is first given to the SVC scheduler for GS flows, and then to the SVC scheduler for CL flows. The WRR scheduler has the lowest priority in receiving service.

Note that the architecture shown in figure 1 is conceptual. Actually, physical packets do not pass the schedulers. The implementation architecture is shown in figure 2. In the implementation architecture, packets are kept in the buffer. The packet selected by the packet scheduler is directly transmitted to the output link without physically passing through the scheduler.

An incoming GS/CL packet may already be marked as out-of-profile by an upstream node. At a node that implements our architecture, each GS/CL flow is reshaped. That is, the out-of-profile status of a GS/CL packet is re-determined by the shaper (embedded in SVC scheduler) [7], regardless of whether the packet is marked or not by an upstream node. The GS/CL packets scheduled by an SVC are all considered in-profile at this node. Thus, these GS/CL packets are unmarked (see figure 1). The GS/CL packets, which are scheduled by DRR, are considered out-of-profile at this node. Therefore, these GS/CL packets are marked (see figure 1). As a result, an in-profile GS/CL packet at the input link may become out-of-profile packet at the output link due to reshaping. Similarly, an out-of-profile GS/CL packet at the input link may become in-profile packet at the output link after reshaping. The reason why we use per-flow queueing and SVC scheduler for in-profile GS/CL packets is based on the results in [7], where it has been shown that SVC scheduling is able to provide bounds on the burstiness of session traf-

[2] How to assign weight for WRR depends on the policy of network operators. If the network operators favor GS/CL users, they may intend to assign a larger weight for out-of-profile GS/CL packets and assign a smaller weight for BE traffic.

fic (delay bounds) and enforce bandwidth allocation for each individual flow. In other words, per-flow queueing with a SVC scheduler in our architecture solves problem P3 associated with the class-based approach.

Note that the packet scheduler SVC/DRR does not distinguish in-profile and out-of-profile GS/CL packets. The SVC or DRR scheduler always serves the Head-of-Line GS/CL packet in the buffer. In other words, the sequence of a flow's packets sent out to the output link is exactly the same as the sequence of the flow's packets arriving at the input link. Thus, there is no out-of-sequence problem at the node. On the other hand, the buffer management scheme ($\mathcal{SP}+$) does distinguish in-profile and out-of-profile GS/CL packets to make discarding/pushout decision. That is, out-of-profile status of a packet is only useful for buffer management but not useful for the scheduler.

The second part of our traffic management algorithms is on call admission control (CAC), which is presented in section 3.2. The objective of CAC is to maximize network utilization (i.e., admit as many flows as possible) and, at the same time, guarantee QoS requirements of all admitted flows (i.e., hard QoS guarantees to GS flows and consistent (or soft) performance to CL flows). We design a simple hybrid CAC algorithm which consists of a model-based CAC for GS flows and a measurement-based CAC for CL flows.[3]

The last part of our architecture is on buffer management, or more specifically, packet discarding strategies when some buffer partition is full.

For GS buffer partition, since the CAC algorithm for an incoming GS flow includes buffer allocation, an admitted flow will have buffer space reservation throughout its path. Therefore, if all the GS packets are in profile, there should not be any buffer overflow. But some upstream nodes may transmit out-of-profile GS packets if there is left-over bandwidth at those nodes. To deal with out-of-profile GS packets, we propose a packet discarding mechanism called *selective pushout* ($\mathcal{SP}$), which will be presented in section 3.3. $\mathcal{SP}$ employs an embedded linked list structure, which cleverly resolves the out-of-sequence problem (P2) associated with the traditional class-based approach.

For CL flows, the buffer partition could also overflow since the traffic behavior of such flows is unpredictable and therefore it is impossible to reserve any buffer space for each CL flow. We propose another mechanism, called *selective pushout plus* ($\mathcal{SP}+$) for CL flows. Under $\mathcal{SP}+$, when the free buffer cannot accommodate the incoming in-profile CL packet, out-of-profile packets will be pushed out first; if the free buffer is still not enough after all out-of-profile packets are discarded, then in-profile packets from the quasi-longest queue will be pushed out. $\mathcal{SP}+$ can control non-conforming behavior from CL flows and, at the same time, achieve fair buffer sharing among competing flows. The details of $\mathcal{SP}+$ algorithm is given in section 3.3.

For BE buffer partition, we use the dynamic threshold algorithm proposed in [2] to achieve fairness among BE flows.

[3] There is no admission control for BE traffic and such type of flows are always admitted.

3. Traffic control algorithms

In this section, we give details on scheduling, call admission control, and buffer management algorithms.

3.1. *Resource allocation and scheduling for GS and CL flows*

3.1.1. Model-based rate calculation for GS flows

According to [6], the end-to-end queueing delay bound for a GS flow j is given as follows.

$$D_j \leqslant \begin{cases} \dfrac{\sigma_j - L_j^{\max}}{p_j - \rho_j} \cdot \left(\dfrac{p_j}{R_j} - 1\right) + \dfrac{L_j^{\max} + C_{\mathrm{tot}_j}}{R_j} + D_{\mathrm{tot}_j} & \text{if } \rho_j \leqslant R_j < p_j, \\ \dfrac{L_j^{\max} + C_{\mathrm{tot}_j}}{R_j} + D_{\mathrm{tot}_j} & \text{if } \rho_j \leqslant p_j \leqslant R_j, \end{cases} \tag{1}$$

where σ_j is the leaky bucket size for flow j, ρ_j is the token generating rate for flow j, p_j is the peak rate of flow j, R_j is the allocated rate for flow j, $L_j^{\max}$ is the maximum packet size of flow j, C_{tot_j} is the rate-dependent error term for flow j, D_{tot_j} is the rate-independent error term for flow j.

Therefore, for a given delay requirement for a GS flow $j \in \mathcal{F}_{\mathrm{GS}}$, its required rate R_j can be easily derived from the above formula, which is a linear function of $1/R_j$. Note that the rate R_j for the flow $j \in \mathcal{F}_{\mathrm{GS}}$ is derived from a leaky bucket model, which is a conservative approach for bandwidth allocation. It is appropriate for GS service since such flows have hard delay requirements [6].

3.1.2. Buffer allocation for GS flows

To guarantee zero packet loss for GS flows, appropriate buffer must be allocated for each GS flow. In [4], Georgiadis et al. derived an upper bound on the buffer requirement for a GS flow. Based on this result, we allocate to flow $j \in \mathcal{F}_{\mathrm{GS}}$ at the lth switch with buffer space

$$b_j^{(l)} = L_j^{\max} + \frac{(p_j - X)(\sigma_j - L_j^{\max})}{(p_j - \rho_j)} + \sum_{k=1}^{l}\left[\frac{C_j^{(k)}}{R_j} + D_j^{(k)}\right]X, \tag{2}$$

where

$$X = \begin{cases} \rho_j & \text{if } \dfrac{\sigma_j - L_j^{\max}}{p_j - \rho_j} \leqslant \displaystyle\sum_{k=1}^{l}\left[\frac{C_j^{(k)}}{R_j} + D_j^{(k)}\right], \\ R_j & \text{if } \dfrac{\sigma_j - L_j^{\max}}{p_j - \rho_j} > \displaystyle\sum_{k=1}^{l}\left[\frac{C_j^{(k)}}{R_j} + D_j^{(k)}\right] \text{ and } p_j > R_j, \\ p_j & \text{otherwise.} \end{cases}$$

In the above equations, $C_j^{(k)}$ and $D_j^{(k)}$ are the rate-dependent error term and the rate-independent error term at the kth switch for flow $j \in \mathcal{F}_{\mathrm{GS}}$, respectively. C_{tot_j} and D_{tot_j} are the respective sum of $C_j^{(k)}$ and $D_j^{(k)}$ along the path of flow $j \in \mathcal{F}_{\mathrm{GS}}$.

3.1.3. Measurement-based rate estimation for CL flows

Unlike GS flows, CL flows cannot be modeled with a set of simple parameters. Furthermore, they do not have hard delay requirements. We can adapt more efficient bandwidth allocation based on the measurement of a CL flow's actual traffic behavior (instead of using a set of rigid parameters).

To measure the effective bandwidth for CL flows, we divide time axis into small fixed interval d and denote t_Y the time required to accumulant a total of Y bits for a particular CL flow. Clearly, t_Y is a variable depending on the particular incoming CL flow traffic behavior. We also introduce a threshold $T_{\max}$ to set up an upper bound on the measurement interval and take the minimum of t_Y and $T_{\max}$ as our measurement window T. That is, $T = \min\{t_Y, T_{\max}\}$.

Denote M the total number of d's within a measurement window T, i.e., $M = \lceil T/d \rceil$. Let $A_i^T(k)$, $1 \leqslant k \leqslant M$, be the number of bits arrived in the kth measurement interval. We first estimate the scaled cumulant generating function (SCGF) $\Lambda(\delta)$.

$$\Lambda^T(\delta_i) = \frac{1}{T} \log \frac{1}{M} \sum_{k=1}^{M} \mathrm{e}^{\delta_i A_i^T(k)},$$

where $\delta_i = -(\log \varepsilon_i - \log \gamma_i)/B_{\mathrm{CL}}$, B_{CL} is the size of the CL buffer partition, ε_i is the packet loss rate requested by source $i \in \mathcal{F}_{\mathrm{CL}}$, and γ_i is the probability that flow $i \in \mathcal{F}_{\mathrm{CL}}$ is non-empty. Let λ_i be the peak rate of flow i. Then, we can obtain the effective bandwidth of CL flow i by

$$\alpha(\delta_i) = \min\left\{\lambda_i, \frac{\Lambda^T(\delta_i)}{\delta_i}\right\}. \tag{3}$$

In our measurement, we only measure the number of packets in bits that have successfully entered the buffer partition, *excluding* discarded packets. This is because that discarded packets will not be served by the scheduler, and thus it is only necessary to consider the packets that have successfully entered the buffer and allocate appropriate rate for their service. Furthermore, we find that such measurement technique has the additional advantage of preventing out-of-profile flows from unfairly increasing their rate share in the scheduler by sending more packets (albeit their rates are limited by their peak rates in equation (3)).

We assume the requirement for packet loss rate is available in order to calculate the required bandwidth. For CL flows, users are not required to explicitly request such QoS parameter. But for engineering purpose (i.e., to estimate required bandwidth), we may assign a value for packet loss rate suitable for the particular CL service.

3.1.4. Rate assignment for GS and CL flows

To provide hard rate guarantee to each GS flow and soft rate guarantee to each CL flow, we employ the following rate assignment strategy in the SVC scheduler. When the sum of guaranteed rates from GS flows (calculated from equation (1)) and the estimated rates from CL flows is less than the link capacity, we use these rates directly in the SVC for the corresponding GS or CL flows and the delay requirement for each GS flow is always guaranteed. On the other hand, if the sum of calculated GS rates and measured CL rates is greater than the link capacity, we shall still use the calculated rate for $j \in \mathcal{F}_{\mathrm{GS}}$ flow as the rate for such flow in the SVC scheduler but use a down-scaled version of the estimated rate for $i \in \mathcal{F}_{\mathrm{CL}}$ flow (by a factor of remaining capacity divided by the sum of estimated CL rates) as the rate for the corresponding CL flow in the SVC scheduler. We name this rate assignment ARC, for Adaptive Rate assignment for Controlled-load.

Algorithm 1 (Adaptive Rate allocation for Controlled-load flows – ARC). For an admitted CL flow $i \in \mathcal{F}_{\mathrm{CL}}$, its rate R_i is given by

$$R_i = \begin{cases} \alpha(\delta_i) & \text{if } \displaystyle\sum_{i\in\mathcal{F}_{\mathrm{CL}}} \alpha(\delta_i) + \sum_{j\in\mathcal{F}_{\mathrm{GS}}} R_j \leqslant C, \\ \alpha(\delta_i) \cdot \dfrac{C - \sum_{j\in\mathcal{F}_{\mathrm{GS}}} R_j}{\sum_{i\in\mathcal{F}_{\mathrm{CL}}} \alpha(\delta_i)} & \text{if } \displaystyle\sum_{i\in\mathcal{F}_{\mathrm{CL}}} \alpha(\delta_i) + \sum_{j\in\mathcal{F}_{\mathrm{GS}}} R_j > C, \end{cases}$$

where C is the link rate.

Once we use the R_j, $j \in \mathcal{F}_{\mathrm{GS}}$, and R_i, $i \in \mathcal{F}_{\mathrm{CL}}$ as the rate for the respective GS or CL flow in our SVC scheduler, we have the following property. The rate allocation for each GS flow is no less than its calculated guaranteed rate while the rate allocation for each CL flow may have occasional fluctuations (due to on-line measurement of each CL flow traffic behavior), which is understood to be a soft bandwidth guarantee.

3.1.5. Shaped virtual clock scheduling for in-profile GS/CL packets

Figure 3 shows the conceptual structure of the buffer and the SVC scheduler. In the buffer, each flow has a logical queue. When a packet of flow i arrives, the packet is put

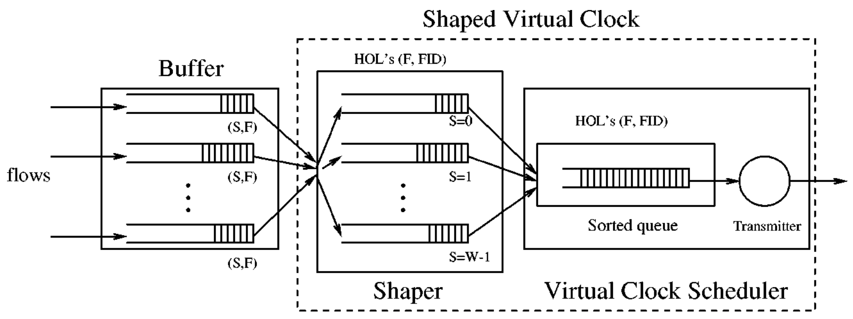


Figure 3. Structure of a shaped virtual clock (SVC) scheduler.

into queue i in the buffer. Only the HOL packet of each queue will be stamped with a virtual start time S and a virtual finish time F. An HOL packet occurs in either one of the following two cases: (1) An incoming packet of flow i joins empty queue i and immediately becomes the HOL; or (2) A packet in queue i departs and its next packet in non-empty queue i immediately becomes the HOL. The information, i.e., (S, F) pairs, will be used by the SVC for scheduling HOL packets. Note that only HOL packets are stamped with (S, F) while other packets (non-HOL packets) are not stamped with (S, F) upon arrival at the buffer [1].

A SVC scheduler maintains a virtual system time $V(t)$, virtual start times $S_i(t)$ $(i \in \{1, \ldots, N\})$, and virtual finish times $F_i(t)$ $(i \in \{1, \ldots, N\})$, where N is the maximum number of flows that the system can support. $S_i(t)$ and $F_i(t)$ are updated upon the arrival of the HOL packet of flow i as follows:

$$S_i(t) = \max\left\{V(t), F_i\left(t^-\right)\right\}, \tag{4}$$

and

$$F_i(t) = S_i(t) + \frac{L_i^{\mathrm{HOL}}}{r_i}, \tag{5}$$

where $F_i(t^-)$ is the finish time of queue i before the update, and L_i^{HOL} is the length of the HOL packet for queue i. The information $V(t)$, $S_i(t)$, and $F_i(t)$ are all represented with finite bits in implementation. Thus, without loss of generality, we suppose W is the maximum of $V(t)$, $S_i(t)$, and $F_i(t)$ that the system can represent. The basic operations of the SVC are described as follows.

(i) When a new HOL packet (its session index, say i) comes, the SVC computes the virtual start time $S_i(t)$ and virtual finish time $F_i(t)$ for this packet according to equations (4) and (5). Then its (F, FID) pair is placed in the shaper queue, where FID denotes the flow identification of the HOL packet. In this way, all the HOL packets are represented in the shaper queue.

(ii) The shaper maintains a logical queue for each discrete start time S. At every time slot, the shaper performs the eligibility test, and sends the (F, FID) pair(s) of those eligible packet(s), if any, to the scheduler queue. Specifically, if the current system time (modulo W) is equal to S, all the (F, FID) pairs in queue S, if any, will be moved to the virtual clock scheduler queue. In other words, only those that are eligible can be stored in the virtual clock scheduler queue.

(iii) The virtual clock scheduler prioritizes all eligible (F, FID) pairs based on their finish times F and chooses the packet with the smallest finish time to transmit first.

(iv) When an HOL packet (its session index, say i) is chosen to be transmitted and leaves the system, the scheduler queue removes its (F, FID) pair and selects another HOL packet, if any, to serve. In the meantime, if queue i in the buffer is not empty, this session can have another packet (regarded as a new HOL packet arrival) to join the shaper queue.

(v) When an out-of-profile GS/CL packet is sent out by the DRR scheduler, its associated pair (F, FID) will be deleted in the shaper queue.[4]

Note that packets do not pass the SVC but are kept in the buffer before being sent out to the output link. The packet selected by the SVC is directly transmitted to the output link without physically passing the SVC.

3.2. *Call admission control*

The objective of call admission control (CAC) is to maximize network utilization (i.e., admit as many flows as possible) and, at the same time, guarantee QoS requirements of all admitted flows. We design a simple hybrid CAC algorithm which consists of a model-based CAC for GS flows and a measurement-based CAC for CL flows. There is no admission control for BE traffic and such type of flows are always admitted.

The following is our CAC algorithm for the GS and CL flows, where μ is target utilization and B_{GS} is the size of GS buffer partition.

Algorithm 2 (CAC algorithm for GS and CL traffic).

```
Upon receiving a new GS flow ($\nu$) request with rate $R_\nu$ and buffer size $b_\nu$
  if $(\sum_{j\in\mathcal{F}_{\mathrm{GS}}} R_j + \sum_{i\in\mathcal{F}_{\mathrm{CL}}} \alpha(\delta_i) + R_\nu \leqslant \mu\cdot C)$ and $(\sum_{j\in\mathcal{F}_{\mathrm{GS}}} b_j + b_\nu \leqslant B_{\mathrm{GS}})$
    admit the new GS flow $\nu$ and stop;
  else
    reject the new GS flow $\nu$ and stop.
Upon receiving a new CL flow ($\upsilon$) request with peak rate $p_\upsilon$
  if $(\sum_{j\in\mathcal{F}_{\mathrm{GS}}} R_j + \sum_{i\in\mathcal{F}_{\mathrm{CL}}} \alpha(\delta_i) + p_\upsilon \leqslant \mu\cdot C)$
    admit the new CL flow $\upsilon$ and stop;
  else
    reject the new CL flow $\upsilon$ and stop.
```

In algorithm 2, we use the peak rate of a CL for admission control rather than the token generating rate ρ. This is because that our previous experience in [10] has shown that the ρ parameter can be less than the required rate and, therefore, the targeted QoS could be violated if we only reserve a bandwidth of ρ.

3.3. *Packet discarding mechanisms*

An arriving packet is allowed to enter the particular buffer partition if there is enough remaining space for the buffer partition. Otherwise, we have to either discard the incoming packet or discard some other packet(s) in the buffer in order to make room for the incoming packet. In this section, we present two packet discarding mechanisms: one

[4] The (F, FID) pair of an out-of-profile GS/CL packet could not be placed in the scheduler queue since all packets in the scheduler queue are in-profile and will be scheduled with a higher priority than that for the DRR scheduler.

is called *selective pushout* ($\mathcal{SP}$), which is employed for the GS partition; the other is called *selective pushout plus* ($\mathcal{SP}+$), which is employed for the CL partition.

3.3.1. Selective pushout ($\mathcal{SP}$) for GS flows

Figure 4 shows the flow chart of the $\mathcal{SP}$ mechanism. According to figure 4, when an unmarked packet arrives at the node, $\mathcal{SP}$ makes every effort to let it enter the GS buffer. Specifically, if the remaining buffer space is not enough and total buffer occupancy by marked packets is larger than the size of the incoming packet, we randomly choose a queue with marked packets and push out enough marked packets from this queue; if all the marked packets in this queue have been discarded while the free space is still not enough, we randomly choose another queue which has marked packets; we continue discarding marked packets in this manner until there is enough free buffer space to accommodate the incoming unmarked packet. On the other hand, when a marked packet arrives at the GS buffer, $\mathcal{SP}$ will let it join the buffer only if there is enough buffer space. Therefore, $\mathcal{SP}$ achieves the highest possible loss protection for in-profile (unmarked) GS packets.

In our implementation, we maintain the following data structure for *each* GS flow to achieve $\mathcal{SP}$ mechanism (see figure 5). Each data unit in the GS buffer consists of a

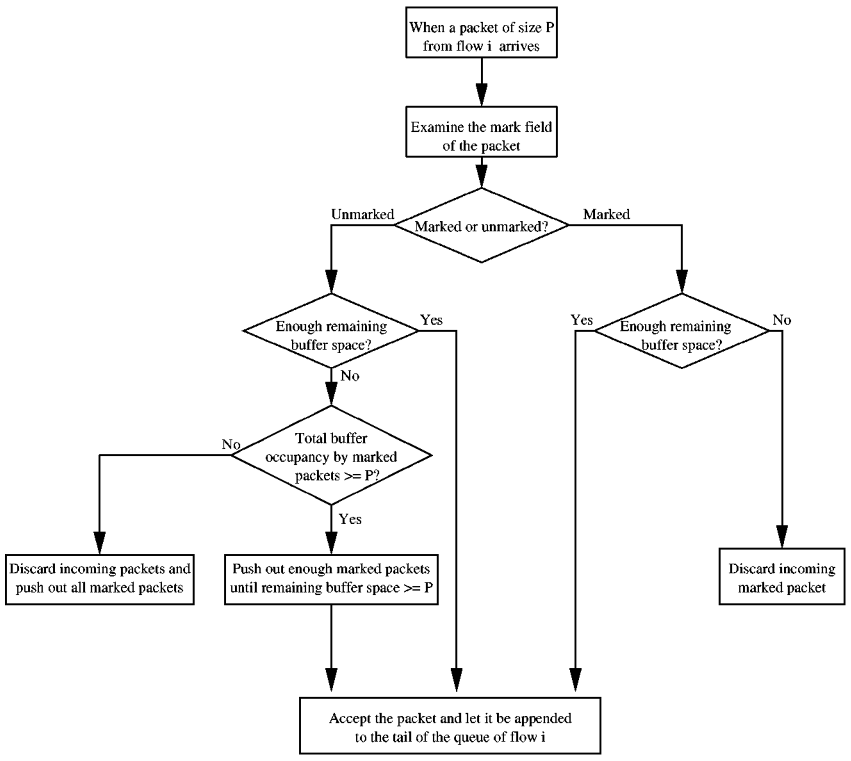

When a packet of size P from flow i arrives
Examine the mark field of the packet
Marked or unmarked?
Unmarked
Marked
Enough remaining buffer space?
Yes
No
Total buffer occupancy by marked packets >= P?
No
Yes
Enough remaining buffer space?
Yes
No
Discard incoming packets and push out all marked packets
Push out enough marked packets until remaining buffer space >= P
Discard incoming marked packet
Accept the packet and let it be appended to the tail of the queue of flow i

Figure 4. Flow chart of $\mathcal{SP}$ discarding mechanism for the GS buffer partition.

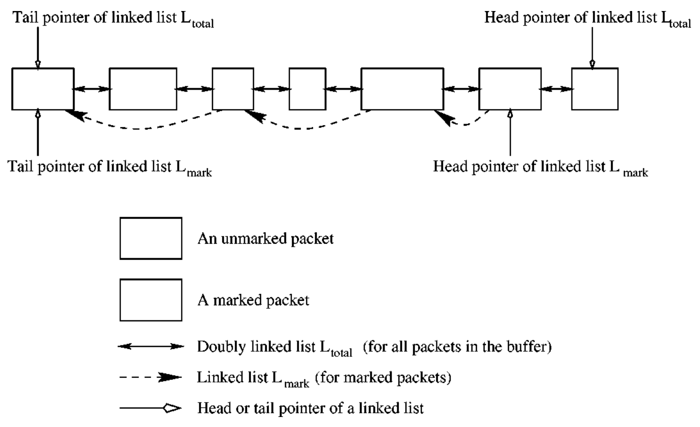

Tail pointer of linked list L_{total}
Head pointer of linked list L_{total}
Tail pointer of linked list L_{mark}
Head pointer of linked list L_{mark}
An unmarked packet
A marked packet
Doubly linked list L_{total} (for all packets in the buffer)
Linked list L_{mark} (for marked packets)
Head or tail pointer of a linked list

Figure 5. Linked list structure for packets of each GS or CL flow in the buffer.

physical IP packet and three pointers, of which two pointers are used for doubly linked list L_{total} and the third is used for linked list L_{mark} as follows.

Linked list L_{total}: is a doubly linked list of all packets from a GS flow (both marked and unmarked). L_{total} is updated whenever an incoming packet is appended to the tail of the queue of the GS flow or a packet is served at the front of the queue of the GS flow by the output link.

Linked list L_{mark}: is the linked list of the marked packets *embedded* in the linked list L_{total}. L_{mark} is updated whenever an incoming marked packet is appended to the tail of the queue of the GS flow or a marked packet is either served by the output link *or* discarded by pushout mechanism.

By using the embedded linked list structure, out-of-profile GS packets can be easily found and discarded while the doubly linked list for L_{total} is able to preserve the connectivity of L_{total} when the packet at the head of L_{mark} is discarded. The introduction of the embedded linked list structure also solves out-of-sequence problem (P2) associated with the class-based approach, which puts out-of-profile GS packets into a separate queue and service these packets with a lower priority.

3.3.2. Selective pushout plus ($\mathcal{SP}+$) for CL flows

Figure 6 shows the flow chart of the $\mathcal{SP}+$ mechanism, where the queue length of flow i, $QL[i]$, is in the unit of bits. In our $\mathcal{SP}+$ mechanism, a register is used to estimate the longest queue (LQ) in the CL buffer partition and is only updated upon the arrival and/or departure of a packet. How to update LQ upon the arrival of a packet is included in the flow chart of figure 6 (arrival updating). Similarly, when a packet from flow $i \in \mathcal{F}_{\text{CL}}$ departs from the output port, if $QL[LQ]$ is less than $QL[i]$, LQ will be updated to i (departure update).

Like $\mathcal{SP}$ mechanism, $\mathcal{SP}+$ makes every effort to let the unmarked CL packet enter the CL buffer. The difference between $\mathcal{SP}+$ and $\mathcal{SP}$ occurs when the remaining buffer

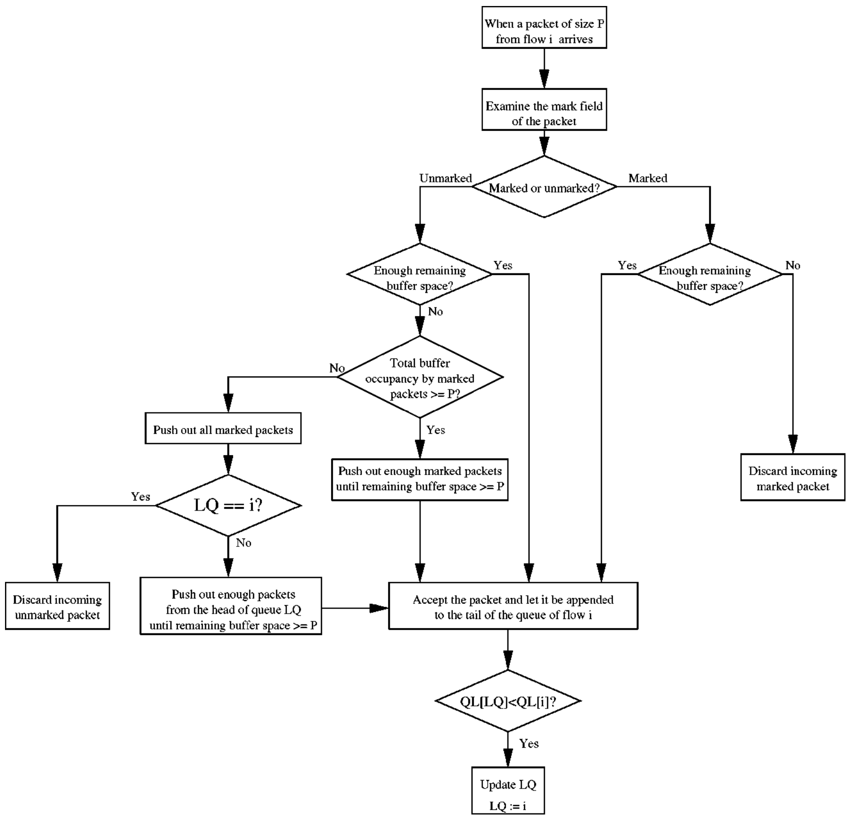

When a packet of size P from flow i arrives
Examine the mark field of the packet
Marked or unmarked?
Unmarked
Marked
Enough remaining buffer space?
Yes
No
Yes
Enough remaining buffer space?
No
Total buffer occupancy by marked packets >= P?
No
Yes
Push out all marked packets
Push out enough marked packets until remaining buffer space >= P
Discard incoming marked packet
LQ == i?
Yes
No
Discard incoming unmarked packet
Push out enough packets from the head of queue LQ until remaining buffer space >= P
Accept the packet and let it be appended to the tail of the queue of flow i
QL[LQ]<QL[i]?
Yes
Update LQ LQ := i

Figure 6. Flow chart of $\mathcal{SP}+$ discarding mechanism for the CL buffer partition.

space is not enough and total buffer occupancy by marked packets is less than the size of the incoming unmarked packet. In this case, if the incoming unmarked packet does not belong to the quasi-longest queue, we push out enough packets from the head of queue *LQ* till the free buffer space could accommodate the incoming packet. This mechanism is called *Quasi-PushOut Plus* (QPO+) [11]. The merit of QPO+ is that QPO+ tends to punish the user with quasi-longest queue when the network is congested. This helps to achieve fairness and to control any non-conforming behavior.

In our implementation, we maintain the same data structure for each CL flow as that for each GS flow (see figure 5). Note that we perform pushout on all marked packets before we check if flow *i* is the quasi-longest queue in figure 6. This is because previous quasi-longest queue may no longer be the quasi-longest queue after pushout.

4. Simulation results

In this section, we implement our IntServ architecture and traffic management algorithms on our network simulator and perform simulations on various benchmark network configurations and traffic conditions.

4.1. Simulation settings

The network configurations that we use are the *peer-to-peer* (figure 7), the *parking-lot* (figure 10), and the *chain* (figure 13) network configurations. All switches in the simulations are assumed to have output port buffering with internal switching capacity equal to the aggregate rates of its input ports. At each output port of a switch, we implement our architecture and traffic management algorithms.

On the connection level, we assume that a GS or CL flow's inter-arrival times is exponentially distributed with an average of 50 s, with the holding time exponentially distributed with an average of 100 s.

The simulation parameters for the GS, the CL, and the BE services are shown in table 1. For GS flows, we use the simple constant bit rate as their traffic pattern. This helps to simplify our simulations without any loss of generality in demonstrating the performance of our architecture and traffic management algorithms. For each BE flow, we use persistent TCP data traffic. For CL flows, we use an exponentially distributed on/off model with average $E(T_{\text{on}})$ and $E(T_{\text{off}})$ for on and off periods, respectively. During each on period, the packets are generated at its peak rate p_i, $i \in \mathcal{F}_{\text{CL}}$. The average bit rate for a CL flow $i \in \mathcal{F}_{\text{CL}}$ is, therefore, $p_i \cdot E(T_{\text{on}})/(E(T_{\text{on}}) + E(T_{\text{off}}))$. Delay bound is obtained by the ratio of σ over ρ. In our simulations, the requested packet loss ratio ϵ for all CL flows is set to 10^{-3}.

In table 2, we list the simulation parameters at each end system (i.e., sender and receiver) and network components (i.e., link and switch). Buffer size in table 2 is the size of the entrance buffer before the leaky bucket. In our simulations, for GS flow $j \in \mathcal{F}_{\text{GS}}$, C_{tot_j} is assumed to be zero and D_{tot_j} only consists of the packet processing

Table 1
Simulation parameters for three types of services.

GS	Peak rate	1.5 Mbps
	Packet size	1 Kbits
	Delay bound	10 ms
CL	Peak rate	1.5 Mbps
	$E(T_{\text{on}})$	2 ms
	$E(T_{\text{off}})$	2 ms
	Packet size	1 Kbits
	Packet loss ratio requirement	10^{-3}
	Delay bound	20 ms
BE (TCP)	Peak rate (light load)	1 Mbps
	Peak rate (heavy load)	10 Mbps
	Mean packet processing delay	300 μs
	Packet processing delay variation	10 μs
	Packet size	1 Kbits
	Maximum receiver window size	64 Kbytes
	Default timeout	500 ms
	Timer granularity	500 ms
	TCP version	Reno

Table 2
Simulation parameters at an end system and network components.

End system	GS	σ	15 packets
		ρ	1500 packets/s
		Buffer size	10 packets
	CL	σ	20 packets
		ρ	1000 packets/s
		Buffer size	10 packets
	TCP	Packet processing delay	500 µs
		Buffer size	500 packets
Switch	Buffer size	GS	500 packets
		CL	500 packets
		BE	1000 packets
	Packet processing delay		4 µs
	Bits required for CL measurement window		100 Kbits
Link	Link speed		10 Mbps
	Distance	End system to switch	1 km
		Switch to switch	1 km

delays at all the switches along its path,[5] i.e., $D_{\mathrm{tot}_j} = h \cdot D_j^{(k)} = h \cdot 4$ µs, where h is the number of switches along the path for flow j. We assume the propagation delay is 5 µs per kilometer. Therefore, the end-to-end delay bound is determined by the end-to-end queueing delay (see (1)) and the total propagation delay. The leaky bucket parameters (σ, ρ) in table 2 are chosen based on the following requirements: the dropped ratio is zero for GS flows and less than 10^{-3} for CL flows;[6] the ratio σ/ρ is equal to the target delay bound.

In our simulations, we set the link capacity to be 10 Mbps and set the target link utilization μ to be 0.90 in order to cushion any traffic fluctuation and measurement error.

We ran our simulator for 300 s simulation time and found that 50 simulated seconds are sufficient for our simulator to warm up. In order to obtain 95% confidence interval, we repeat each simulation eight times, each of which with a different seed.

4.2. Simulation results

We organize our presentation as follows. Section 4.2.1 presents the performance of the GS, the CL, and the BE traffic under light and heavy load conditions and show that criteria C1 and C2 are satisfied. In section 4.2.2, we show that our architecture and algorithms can effectively control non-conforming flows by minimizing their negative impact on other conforming flows (criterion C3). Section 4.2.3 compares our $\mathcal{SP}+$ packet discarding with the tail-dropping mechanism.

[5] The reason why D_{tot_j} does not include the propagation delay is that the propagation delay does not contribute to the buffer requirement in (2).

[6] Dropped ratio is the ratio of the dropped packets at the entrance buffer over the total generated packets.

4.2.1. Performance under light and heavy load conditions

We investigate the QoS experienced by the GS, the CL, and the BE traffic under light and heavy load conditions using various benchmark network configurations. The purpose of the simulations is to demonstrate that our network architecture and traffic management algorithm can achieve criteria C1 and C2.

(i) The peer-to-peer network. For this network (figure 7), the output port link of SW1 is the only bottleneck link for all flows. Figure 8 shows the link utilization on Link12 under light and heavy load conditions. Table 3 shows the number of flows under light and heavy load conditions in our simulation. Note that only the GS and the CL flows requires admission control while there is no admission control for BE traffic.

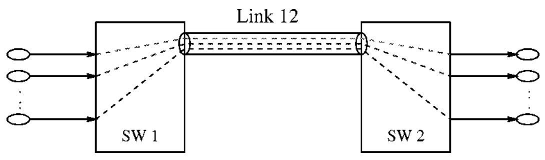


Figure 7. A peer-to-peer network.

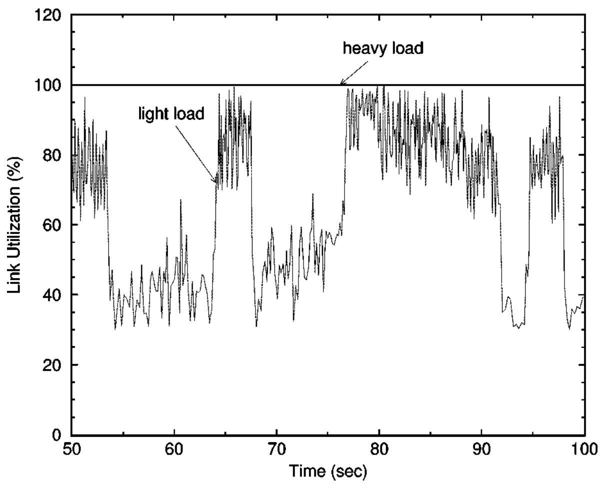


Figure 8. Link12 utilization under light and heavy load in the peer-to-peer network.

Table 3
Number of flows under light and heavy load conditions
in the peer-to-peer network.

Load conditions	Number of flows		
	GS	CL	BE (TCP)
Light	3	3	3
Heavy	4	8	5

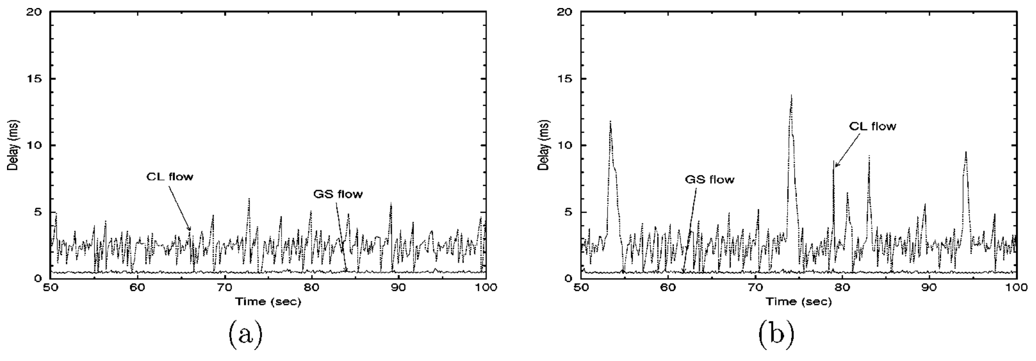


Figure 9. End-to-end delays for packets of a GS flow and a CL flow under (a) light load and (b) heavy load in the peer-to-peer network.

Table 4
Performance of BE (TCP) traffic under light and heavy load conditions in the peer-to-peer network.

Performance of BE (TCP) traffic		Load conditions	
		Light	Heavy
Throughput (Kbps)	TCP1	325	62.4
	TCP2	291	64.6
	TCP3	312	61.2
	TCP4	–	63.1
	TCP5	–	62.5
Packet loss rate (%)	Link 12	0	2.5

Under light load, the 95% confidence intervals for the maximum end-to-end delay (in ms) for GS and CL flows are (0.792, 0.846) and (7.35, 9.21), respectively; under heavy load, they are (0.801, 0.852) and (12.66, 14.34), respectively. We find that the delays experienced by each GS and CL flows are bounded and are less than the delay requirements for GS and CL flows, respectively. For illustration, we randomly pick up a GS flow and a CL flow among admitted GS and CL flows and plot their delay behavior under light and heavy load conditions in figure 9. As shown in figure 9, the delays experienced by this GS flow are bounded under both light and heavy load conditions and is much less than the delay bound requirement (10 ms). For the CL flow, its delays are also bounded under both conditions and are less than its delay requirements (20 ms). As expected, there is a delay increase for this CL flow under heavy load than under light load. But such increase is normal and the specific application supported by this CL flow should operate properly without any significant performance degradation.

Under both light and heavy load conditions, there is no packet loss from any GS or CL flow. In addition, we observe no out-of-sequence phenomena for each flow.

Table 4 shows the performance of BE flows under light and heavy load. We observe that the throughput of TCP1, TCP2, and TCP3 decrease under heavy load as expected. Unlike GS and CL traffic, there is packet loss for BE traffic under heavy load conditions.

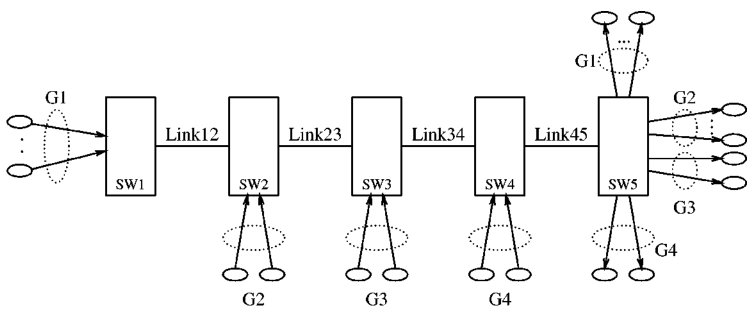


Figure 10. A parking-lot network.

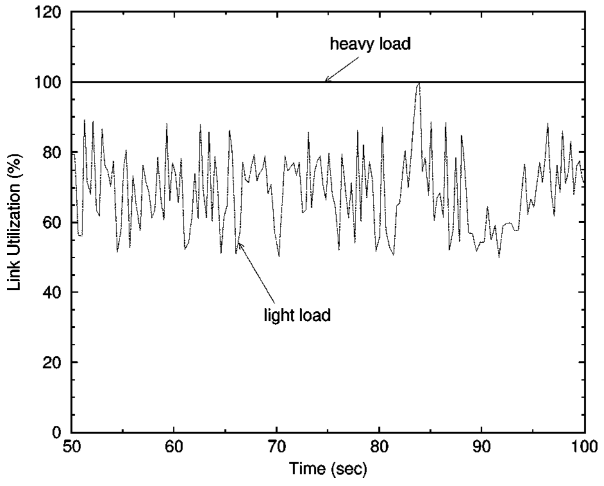


Figure 11. Link utilization of Link45 under light and heavy load in the parking-lot network.

(ii) The parking-lot network. The parking-lot network is shown in figure 10, where path G1 consists of multiple flows and traverse from the first switch (SW1) to the last switch (SW5), path G2 starts from SW2 and terminates at the last switch (SW5), and so forth. Clearly, Link45 is the potential bottleneck link for all flows.

Figure 11 shows the link utilization at Link45 under light and heavy load conditions. Table 5 shows the number of flows on each path under light and heavy load conditions in our simulation. Under light load, the 95% confidence intervals for the maximum end-to-end delay (in ms) for GS and CL flows are (1.586, 1.691) and (8.32, 9.28), respectively; under heavy load, they are (1.718, 1.784) and (15.11, 15.83) for GS and CL flows, respectively. We also find that the delays experienced by each GS and CL flows are bounded and are less than the delay requirements for GS and CL flows, respectively. In figure 12, we plot the delays experienced by the GS flow and the CL flow traversing SW1 to SW5 (path G1) under light and heavy load. As shown in both figures, the delays experienced by this GS flow are bounded and are much less than its delay bound requirement (10 ms). For the CL flow, its delays are also bounded under

Table 5
Number of GS, CL, and BE flows on each path under light and heavy load conditions in the parking-lot network.

Path	Traffic type	Number of flows	
		Light load	Heavy load
G1	GS	1	1
	CL	1	2
	BE (TCP)	1	2
G2	GS	1	1
	CL	1	2
	BE (TCP)	1	2
G3	GS	1	1
	CL	1	2
	BE (TCP)	1	2
G4	GS	1	1
	CL	1	2
	BE (TCP)	1	2

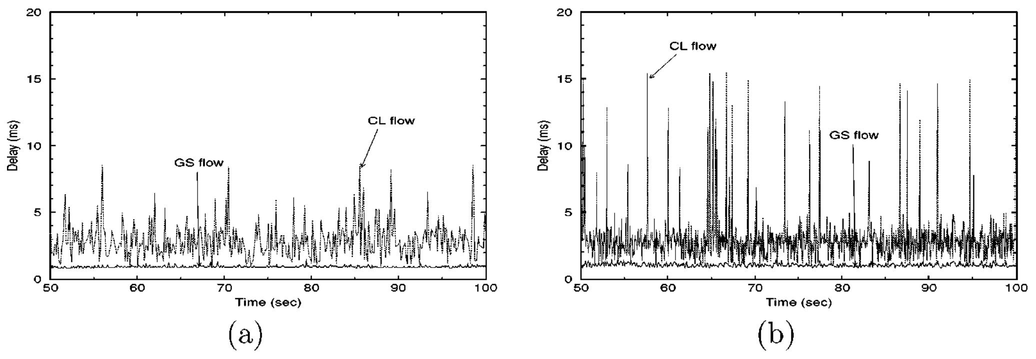


Figure 12. End-to-end delays for packets of a GS flow and a CL flow under (a) light load and (b) heavy load in the parking-lot network.

both conditions and are less than the delay requirements (20 ms). As expected, there is some occasional delay increase for this CL flow under heavy load than under light load. Again, such increase is normal and is considered satisfying our performance objective for CL flows. Under both light and heavy load conditions, there is no packet loss from any GS or CL flow. In addition, we observe no out-of-sequence phenomena for each flow.

Table 6 shows the performance of BE flows under light and heavy load. We observe that the throughput of TCP1, TCP2, TCP3, and TCP4 all decrease under heavy load as expected. In contrary to GS and CL traffic, there is packet loss for BE traffic under heavy load conditions. We find such loss occurs at Link34 (output port of SW3) and Link45 (output port of SW4), respectively.

Table 6
Performance of BE (TCP) traffic under light and heavy load conditions in the parking-lot network.

Performance of BE (TCP) traffic		Load conditions	
		Light	Heavy
Throughput (Kbps)	TCP1	33.2	11.2
	TCP2	32.8	10.9
	TCP3	32.5	10.5
	TCP4	31.7	10.1
	TCP5	–	11.2
	TCP6	–	10.9
	TCP7	–	10.5
	TCP8	–	10.1
Packet loss ratio (%)	Link12	0	0
	Link23	0	0
	Link34	0	2.5
	Link45	0	15.2

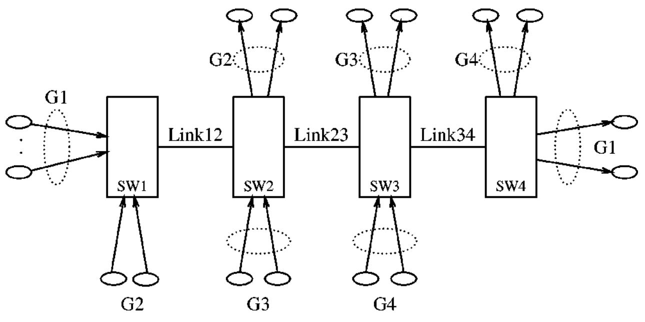


Figure 13. A chain network.

(iii) The chain network. This is one of the benchmark network configurations used to examine traffic behavior under the impact of traversing interfering traffic. The specific chain configuration that we use is shown in figure 13 where path G1 consisting of multiple flows and traverses from the first switch (SW1) to the last switch (SW4), while all the other paths traverse only one hop and "interfere" the flows in G1. Figure 14 shows the link utilization of each link during (a) light load and (b) heavy load conditions, respectively. Table 7 shows the number of flows under light and heavy load conditions.

Under light load, the 95% confidence intervals for the maximum end-to-end delay (in ms) for GS and CL flows are (1.524, 1.556) and (8.01, 8.83), respectively; under heavy load, they are (1.566, 1.603) and (13.07, 14.11) for GS and CL flows, respectively. We find that the end-to-end delays experienced by each GS and CL flow satisfy their respective delay requirements. Furthermore, we find that the packet loss is zero for both GS and CL flows. As an illustration, figure 15 shows the delay experienced by the GS

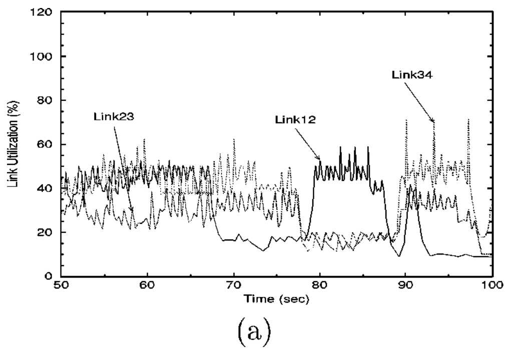


Figure 14. Link utilization under (a) light load and (b) heavy load in the chain network.

Table 7
Number of flows on each path under light and heavy load conditions in the chain network.

Path	Traffic type	Number of flows	
		Light load	Heavy load
G1	GS	1	1
	CL	1	2
	BE (TCP)	1	3
G2	GS	0	1
	CL	1	2
	BE (TCP)	1	3
G3	GS	1	1
	CL	1	2
	BE (TCP)	1	3
G4	GS	1	1
	CL	1	2
	BE (TCP)	1	3

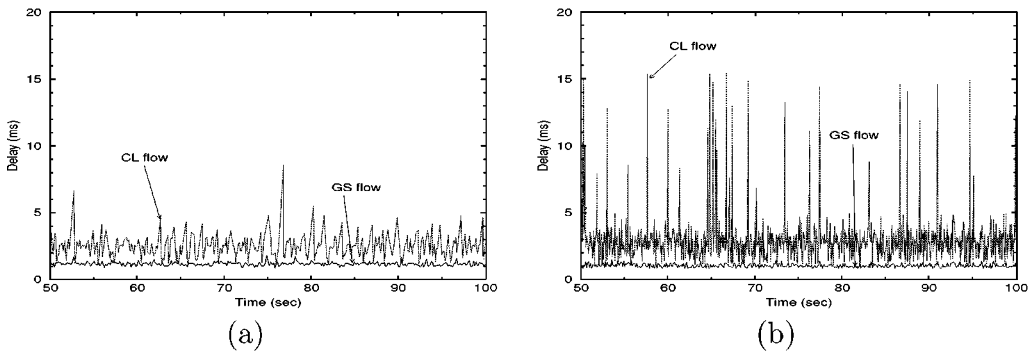


Figure 15. End-to-end delay of a GS flow and a CL flow under (a) light load and (b) heavy load in the chain network.

Table 8
Performance of BE (TCP) traffic under light and heavy load conditions in the chain network.

Performance of BE (TCP) traffic		Load conditions	
		Light	Heavy
Throughput (Kbps)	TCP1	33.1	30.3
	TCP2	32.6	28.9
	TCP3	32.9	29.6
	TCP4	32.2	24.9
	TCP5	–	30.3
	TCP6	–	28.9
	TCP7	–	29.6
	TCP8	–	25.8
	TCP9	–	30.3
	TCP10	–	28.9
	TCP11	–	29.6
	TCP12	–	24.9
Packet loss ratio (%)	Link 12	0	0
	Link 23	0	0.69
	Link34	0	5.37

and the CL flows traversing from SW1 to SW4 (path G1) under (a) light and (b) heavy load conditions. Table 8 lists the throughput and packet loss ratios for BE flows under light and heavy load conditions.

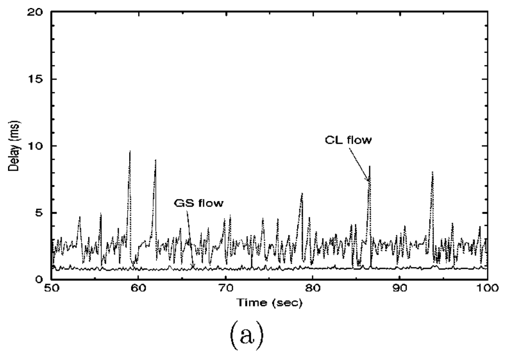
4.2.2. Control of non-conforming CL flows

For those nodes that have policing mechanism, non-conforming GS/CL flows can be effectively controlled by marking out-of-profile GS/CL packets and discarding them when the corresponding buffer partition is full.

On the other hand, according to [9], network elements must not assume that each CL source or upstream elements have policing mechanism in place. Under such circumstances, the CL packets that are actually out-of-profile may not be marked at upstream elements/sources. We show that our architecture and algorithms can effectively control such non-conforming CL flows and thus achieve criterion C3.

We use the parking-lot configuration under heavy traffic load for demonstration. The non-conforming flow is on path G4, which shares the bottleneck link (Link45) with all other flows on paths G1, G2 and G3. The non-conforming flow submits a peak rate of 1.5 Mbps as its traffic parameter for admission control but actually transmits at a peak rate of 10 Mbps. Since there is no policing mechanism for this flow at the entrance to the network, all out-of-profile packets from this flow are not marked.

Our simulations show that in the presence of such non-conforming CL flow, the contracted QoS to those conforming GS/CL flows can still be guaranteed while the non-conforming flow can be effectively isolated (due to per-flow queueing) and suffers from



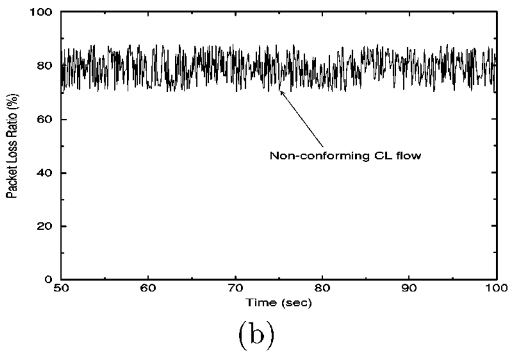


Figure 16. (a) End-to-end delay for conforming GS and CL flows; and (b) packet loss ratio for the non-conforming CL flow in the parking-lot network.

Table 9
Throughput of TCP connections under the presence of a non-conforming CL flow in the parking-lot configuration.

Flow	TCP1	TCP2	TCP3	TCP4	TCP5	TCP6	TCP7	TCP8
Throughput (Kbps)	8.2	7.8	7.6	7.1	8.2	7.8	7.6	7.1

large packet loss rate (due to $\mathcal{SP}+$ packet discarding). In particular, we plot the delays for a conforming GS and CL flows on path G1 in figure 16(a), which shows that the delays experienced by these conforming GS and CL flows are bounded and meet their respective delay requirements. Furthermore, we find that the packet loss rate for these conforming flows remains zero during the simulation run. On the other hand, figure 16(b) shows that the packet loss ratio experienced by the non-conforming CL flow suffers from heavy packet loss during the simulation.

The throughput of TCP connections (see table 9) are comparable to those under heavy load in table 6. So the non-conforming CL flow does not starve BE traffic.

We have just demonstrated that our node architecture and traffic management algorithms are capable of controlling non-conforming CL flows. Such effective control is credited mostly to per-flow queueing based $\mathcal{SP}+$ algorithm. In the following subsection, we further examine $\mathcal{SP}+$ algorithm.

4.2.3. $\mathcal{SP}+$ vs. tail-dropping

We compare the performance of $\mathcal{SP}+$ with tail-dropping packet discarding scheme. Again, we use the same simulation settings in section 4.2.2, except we discard the incoming packet when the buffer partition is full (tail-dropping) instead of $\mathcal{SP}+$. Poisson call arrival is not used, and instead, we just run the simulation for 300 s.

Figure 17 shows that under tail-dropping, even conforming CL flow experiences large packet loss. On the other hand, the same conforming CL flow experienced zero packet loss under $\mathcal{SP}+$ packet discarding mechanism in section 4.2.2.

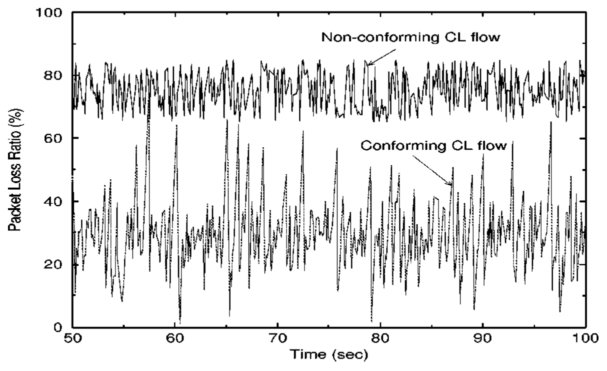


Figure 17. Packet loss ratio for conforming and non-conforming CL flows in the parking-lot network under tail-dropping packet discarding mechanism.

5. Conclusions

This paper presents a per-flow based node architecture and traffic management algorithms, which have been demonstrated, to offer QoS provisioning for integrated traffic of the guaranteed service, the controlled-load, and the best-effort services for the IntServ networks. Our main contribution is that our architecture and traffic management algorithms not only meet the three criteria for IntServ networks, but also resolve several problems associated with the traditional class-based approach.

References

[1] H.J. Chao, Y.R. Jenq, X. Guo and C.H. Lam, Design of packet-fair queueing schedulers using a RAM-based searching engine, IEEE Journal on Selected Areas in Communications 17(6) (June 1999) 1105–1126.

[2] A.K. Choudhury and E.L. Hahne, Dynamic queue length thresholds in a shared memory ATM switch, in: *Proc. of IEEE INFOCOM'96,* San Francisco, CA, USA, March 1996, pp. 679–687.

[3] D. Clark, S. Shenker and L. Zhang, Supporting real-time applications in an integrated services packet network: architecture and mechanism, in: *Proc. of ACM SIGCOMM'92,* Baltimore, MD, USA, August 1992.

[4] L. Georgiadis, R. Guerin, V. Peris and R. Rajan, Efficient support of delay and rate guarantee, in: *Proc. of ACM SIGCOMM'96,* Stanford, CA, USA, August 1996.

[5] S. Jamin, P.B. Danzig, S. Shenker and L. Zhang, A measurement-based admission control algorithm for integrated services packet networks, IEEE/ACM Transactions on Networking 5(1) (February 1997) 56–70.

[6] S. Shenker, C. Partridge and R. Guerin, Specification of guaranteed quality of service, RFC 2212, Internet Engineering Task Force (September 1997).

[7] D. Stiliadis and A. Varma, A general methodology for designing efficient traffic scheduling and shaping algorithms, in: *Proc. of IEEE INFOCOM'97,* Kobe, Japan, April 1997.

[8] B. Suter, T.V. Lakshman, D. Stiliadis and A.K. Choudhury, Design considerations for supporting TCP with per-flow queueing, in: *Proc. of IEEE INFOCOM'98,* San Francisco, CA, USA, March 1998, pp. 299–306.

[9] J. Wroclawski, Specification of the controlled-load network element service, RFC 2211, Internet Engineering Task Force (September 1997).

[10] D. Wu and H.J. Chao, Efficient bandwidth allocation and call admission control for VBR service using UPC parameters, in: *Proc. of IEEE INFOCOM'99,* New York, NY, USA, March 1999.

[11] D. Wu, Y.T. Hou, Z.-L. Zhang, H.J. Chao, T. Hamada and T. Taniguchi, On implementation architecture for achieving QoS provisioning in integrated services networks, in: *Proc. of IEEE ICC'99*, Vancouver, BC, Canada, June 1999, pp. 461–468.