ELSEVIER

Computer Networks 34 (2000) 481–501

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

On network bandwidth allocation policies and feedback control algorithms for packet networks

Y. Thomas Hou [a,*], Bo Li [b], Shivendra S. Panwar [c], Henry Tzeng [d]

[a] *Fujitsu Laboratories of America, 595 Lawrence Expressway, Sunnyvale, CA 94085, USA*
[b] *Hong Kong University of Science and Technology, Kowloon, Hong Kong*
[c] *Polytechnic University, Brooklyn, NY, USA*
[d] *Amber Networks, Inc., Santa Clara, CA, USA*

Received 29 February 2000; received in revised form 18 April 2000; accepted 29 May 2000

Abstract

This paper summarizes our experience on the design of network bandwidth allocation policies and distributed rate calculation algorithms for packet-switched networks. In particular, we discuss two rate allocation policies: the generalized max–min (GMM) and the weight-proportional max–min (WPMM) policies, both of which generalize the classical max–min rate allocation policy. For the design of distributed algorithms to achieve these two rate allocation policies, we focus on rate-based distributed flow control where special control packets are employed to achieve the information exchange between a source and the network. We categorize two broad classes of distributed rate calculation algorithms in the literature using live algorithms as illustrations. To give insight, we compare the design tradeoffs between these two classes of algorithms in terms of performance objectives and implementation complexities. Furthermore, we discuss important extensions within each class of algorithms.

Keywords: Max–min; Minimum rate; Peak rate; Rate allocation policy; Congestion and flow control; Explicit rate; Feedback; ATM; Available bit rate; Packet network

1. Introduction

The available bit rate (ABR) service class defined by the ATM Forum supports applications that allow the ATM source end system to adjust the information transfer rate based on the bandwidth availability in the network [4]. Our objective in this paper is not to review the ABR details and standards, which we refer to overview papers such as [2,9,12,15,29], but rather, to share with the readers our experience on some important principles of such rate-based feedback control. This paper is targeted to readers who already have some familiarity with rate-based flow control and therefore, are well-positioned to have a broader and deeper understanding of such traffic control algorithms. In particular, we offer a systematic study on several network bandwidth allocation policies and discuss the design methodologies of

* Corresponding author. Tel.: +1-408-530-4529; fax: +1-408-530-4515.
E-mail address: thou@fla.fujitsu.com (Y. Thomas Hou).

1389-1286/00/$ - see front matter
PII: S1389-1286(00)00132-8

distributed rate calculation algorithms in the broader context of packet-switched networks.

We would like to point out that the material covered in this paper is very general and can be applied to any flow oriented packet-switched networks, although for convenience, we use ABR terminology to illustrate a particular distributed rate calculation algorithm.

We start with a brief review of the classical max–min rate allocation policy [5], which has been widely accepted as an optimal network bandwidth sharing criterion among user traffic flows, including for ATM ABR service. [1] The classical max–min rate allocation policy cannot support a minimum rate requirement and a peak rate constraint for each flow. To address this issue, we present two network bandwidth sharing policies, each of which is a generalization of the classical max–min. The first policy, called the generalized max–min (GMM) [22], makes a direct generalization of the max–min by first satisfying each flow's minimum rate requirement and then maximizes the rate of the session that is the smallest among all sessions while satisfying this session's peak rate constraint, given the best smallest rate allocation, we continue to maximize the rate of the connection with the second smallest rate, and so forth. The second policy, called the weight-proportional max–min (WPMM) [23], associates a weight with each session. It allocates each session its minimum rate requirement and shares the remaining network bandwidth among user flows using a weight proportional version of the max–min policy based on each flow's weight. We show that the classical max–min rate allocation is a special case of both the GMM and the WPMM policies.

Since a centralized algorithm for either the GMM or the WPMM rate allocation requires global network information, which is difficult to obtain, we are interested in the design of distributed algorithms to achieve the same rate allocation objective in the absence of global knowledge about the network and without synchronization of different network components. We consider a network in which the switches maintain their own controls and communicate these controls to the source by feedback. In particular, we focus on the end-to-end rate-based feedback control scheme, where special control packets are used in both forward and backward paths. The source uses control packets in the forward path to inform the switches about the source's rate information. The switches perform rate calculations for each flow and use the control packets in the backward path to advise the sources to adjust their rates. The goal is to properly design this flow control protocol so that eventually each source's rate conforms to the rate allocation objective (i.e., GMM or WPMM).

Since the objective of this paper is to share our experience on the design methodologies of distributed rate calculation algorithms that can converge to the GMM or WPMM rate allocation policy, we will focus only the so-called *explicit rate* feedback control algorithms and will not discuss any binary feedback algorithm (e.g., [36,44]), which has rate oscillations and strictly speaking, cannot converge to a particular rate allocation policy.

We classify the explicit rate calculation algorithms into two broad classes based on how much state information for each traffic flow is required at the switch. Class 1 algorithms employ only a few switch variables and use simple heuristics based on congestion information (e.g., queue length, load) to achieve the rate allocation objective. They do not require the switch to maintain the state information of each traversing flow (also called per flow accounting for ABR). We show that such algorithms provide satisfactory performance in a local area network environment. Class 2 algorithms use per flow accounting at a switch's output port for rate calculation. With this additional complexity, such algorithms can provide guaranteed convergence to the particular rate allocation objective under *any* network configuration and *any* set of link distances [10,22,24]. We compare these two classes of algorithms in terms of convergence property, sensitivity to system parameters, state requirement, computational complexity, etc. and show the design tradeoffs between these two classes of algorithms in terms of performance objectives and implementation complexity. Both

[1] We use the terms "flow", "session", "connection", and "virtual connection" interchangeably throughout the paper.

Class 1 and Class 2 algorithms are rate calculation algorithms and do not impose any special requirements on buffering and scheduling schemes. In fact, most of Class 1 and Class 2 algorithms assume a simple shared buffer and a FIFO scheduling policy. We will discuss how sophisticated buffering and scheduling policies can be employed to further enhance the performance of each class of algorithms.

The remainder of this paper is organized as follows. Section 2 briefly reviews the classical max–min rate allocation and shows how the GMM and the WPMM policies can extend the classical max–min policy with minimum rate and peak rate support. In Section 3, we classify some well-known distributed rate calculation algorithms into two broad classes and Section 4 discusses the design tradeoffs between these two classes of algorithms in terms of performance objectives and implementation complexities. Some important extensions within each class of algorithms are also discussed. Section 5 concludes this paper.

2. Max–min rate allocation policy and extensions

We are interested in optimal allocation of network bandwidth for each user flow in packet-switched networks. Specifically, we want to have a rate allocation to be feasible in the sense that the total throughput of all sessions crossing any link does not exceed that link's capacity; we also want the feasible rate allocation to be fair (in some sense) to all sessions and the network to be utilized as much as possible.

In this section, we first briefly review the classical max–min rate allocation policy, which has been widely accepted as a fair and efficient criterion to allocate network bandwidth [5]. Then we move on to generalize the classical max–min policy.

2.1. The classical max–min

We use the following simple example to illustrate the basic concept of max–min rate allocation.

Example 1 (*Max–min rate allocation*). As shown in Fig. 1, one session (s_1) traverses the tandem connection of all links, and other sessions go through only one link. It is plausible to limit sessions 1, 2, and 3 to a rate of $1/3$ each, since this gives each of these sessions as much rate as the others. It would be rather pointless, however, to restrict session 4 to a rate of $1/3$. Session 4 might better be limited to $2/3$, since any lower limit would waste some of the capacity of Link23 without benefiting sessions 1, 2, or 3, and any higher limit would be unfair because it would further reduce session 1.

This example leads to the idea of maximizing the network use allocated to the sessions with the minimum allocation, thus giving rise to the term *max–min* flow control. After these poorly treated sessions are given the greatest possible allocation, there might be considerable latitude left for choosing allocations for the other sessions. It is then reasonable to maximize the allocation for the most poorly treated of these other sessions, and so forth, until all allocations are specified. An alternative way to express this intuition, which turns out to be equivalent to the above, is to maximize the allocation of each session subject to the constraint that an incremental increase in i's allocation does not cause a decrease in some other session's allocation that is already as small as i's or smaller.

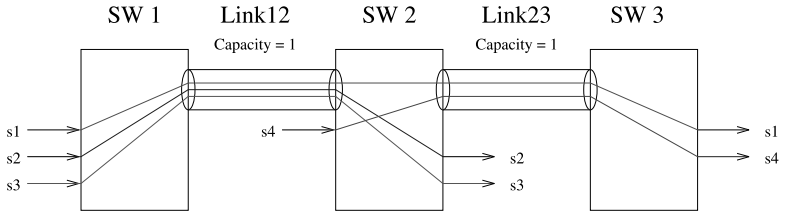


Fig. 1. Four sessions sharing a three-node network.

We use the following simple mathematical notation to model max–min rate allocation in a network. Assuming that a network $\mathcal{N}$ is characterized by interconnecting switches with a set of links $\mathcal{L}$, a set of sessions $\mathcal{S}$ are using the network and each session $s \in \mathcal{S}$ traverses one or more links in $\mathcal{L}$. Letting r_s denote the allocated rate for session s and $\mathcal{S}_\ell$ the set of sessions traversing link ℓ, the aggregated flow on link ℓ of the network is then $F_\ell = \sum_{s \in \mathcal{S}_\ell} r_s$.

Let C_ℓ be the capacity of link ℓ, we have the following constraints on the vector $r = \{r_s | s \in \mathcal{S}\}$ of allocated rates: (1) $r_s \geqslant 0$ for all $s \in \mathcal{S}$; and (2) $F_\ell \leqslant C_\ell$ for all $\ell \in \mathcal{L}$. A vector satisfying these constraints is said to be *feasible*.

A rate vector r is said to be max–min if it is feasible and for each $s \in \mathcal{S}$, r_s cannot be increased while maintaining feasibility without decreasing r_t for some session t for which $r_t \leqslant r_s$. More formally, r is max–min if it is feasible, and for each $s \in \mathcal{S}$ and feasible $\hat{r}$ for which $r_s < \hat{r}_s$, there exists some session $t \in \mathcal{S}$ such that $r_s \geqslant r_t$ and $r_t > \hat{r}_t$.

Given a feasible rate vector r, we say that a link $\ell \in \mathcal{L}$ is a bottleneck link with respect to r for a session s traversing ℓ if $F_\ell = C_\ell$ and $r_s \geqslant r_t$ for all sessions t traversing link ℓ.

In Example 1, the bottleneck links of sessions 1, 2, 3, and 4 are Link12, Link12, Link12, and Link23, respectively. Link23 is not a bottleneck link for session 1 since sessions 1 and 4 share this link and session 4 has a larger rate than session 1.

It turns out that in general, each session has a bottleneck link and a feasible rate vector r is max–min if and only if each session has a bottleneck link with respect to r.

In the following, we give an algorithm for computing max–min rate vector. The idea of the algorithm is to start with all-zero rate vector and to increase the rates on all sessions together until $F_\ell = C_\ell$ for one or more links ℓ. At this point, each session using a saturated link (i.e., a link with $F_\ell = C_\ell$) has the same rate as every other session using that link. Thus, these saturated links serves as bottleneck links for all sessions using them. At the next step of the algorithm, all sessions not using the saturated links are incremented equally in rate until one or more links become saturated. Note that the sessions using the previously saturated links might also be using these newly saturated links. The newly saturated links serve as bottleneck link for these sessions that pass through them but do not use the previously saturated links. The algorithm continues from step to step, always equally incrementing all sessions not passing through any saturated link; when all sessions pass through at least one saturated link, the algorithm stops.

Algorithm 1 (*Max–min rate allocation*).

1. Start the rate allocation of each session with zero.
2. Increase the rate of each session with the smallest rate such that some link becomes saturated.
3. Remove those sessions that traverse saturated links and the capacity associated with such sessions from the network.
4. If there is no session left, the algorithm terminates; otherwise, go back to Step 2 for the remaining sessions and network capacity.

Applying the above algorithm for the four-session three-node network in Example 1, it is easy to show that sessions 1, 2, and 3 get a rate of $1/3$ and session 4 gets a rate of $2/3$.

It can be easily seen that the max–min rate allocation is fair in the sense that all sessions constrained by a particular bottleneck link get an equal share of this bottleneck capacity. It is also efficient in the sense that given the max–min rate allocation, no session can push more flow of data through the network, since each session traverses at least one fully saturated link.

An important limitation associated with the classical max–min rate allocation is that it does not address how to support minimum rate and peak rate constraints from each session. For the remaining of this section, we show how to extend the classical max–min with the GMM [22] and WPMM [23].

2.2. Generalized max–min

Let MR_s and PR_s be the minimum rate requirement and the peak rate constraint for each session $s \in \mathcal{S}$. We assume that the sum of all sessions' minimum rate traversing any link does not exceed the link's capacity, i.e., $\sum_{s \in \mathcal{S}_\ell} \mathrm{MR}_s \leqslant C_\ell$ for

every $\ell \in \mathscr{L}$. This assumption can be enforced by admission control at call setup time to determine whether or not to accept a new connection.

We say that a rate vector $r = \{r_s | s \in \mathscr{S}\}$ is *MP-feasible* if it satisfies the minimum rate and peak rate constraints for each session *and* it is feasible, i.e., (1) $\mathrm{MR}_s \leqslant r_s \leqslant \mathrm{PR}_s$ for all $s \in \mathscr{S}$; and (2) $F_\ell \leqslant C_\ell$ for all $\ell \in \mathscr{L}$.

The GMM rate allocation holds the same fundamental concept as the classical max–min policy, i.e., maximizing the minimum rate among all sessions (while satisfying each session's minimum rate requirement and peak rate constraint), given the best smallest rate allocation, maximize the second smallest rate allocation, and so forth.

Algorithm 2 (*GMM rate allocation*).
1. Start the rate of each session with its MR.
2. Increase the rate of the session with the smallest rate among all sessions until one of the following events takes place: (a) The rate of such session reaches the second smallest rate among the sessions; or (b) Some link saturates; or (c) The session's rate reaches its peak rate (PR).
3. If some link saturates or the session's rate reaches its PR in Step 2, remove the sessions that either traverse the saturated link or reach their PRs, respectively, as well as the network capacity associated with such sessions from the network.
4. If there is no session left, the algorithm terminates; otherwise, go back to Step 2 for the remaining sessions and network capacity.

Example 2 (*GMM rate allocation*). As an example, we use the same four-session three-node network shown in Fig. 1. The minimum rate requirement and peak rate constraint for each session are listed in Table 1.

Table 1
Minimum rate requirement, peak rate constraint, and GMM rate allocation for each session in the three-node network

Session	MR	PR	GMM rate allocation
$s1$	0.40	1.00	0.40
$s2$	0.10	0.25	0.25
$s3$	0.05	0.50	0.35
$s4$	0.10	1.00	0.60

The iterative steps to achieve the GMM rate allocation are described below, with a graphical display shown in Fig. 2.
- *Step* 1: As shown in Fig. 2, we start the rate of each session with its MR (shown in the darkest shaded areas in Fig. 2).
- *Step* 2: Since the rate of $s3$ (0.05) is the smallest among all sessions, we increase it until it reaches the second smallest rate, which is 0.1 ($s2$ and $s4$).
- *Step* 3: The rates of $s2$, $s3$ and $s4$ all being 0.1, we increase them together until $s2$ reaches its PR constraint of 0.25.
- *Step* 4: Remove $s2$ (with a rate of 0.25) from future iterations and we now have the rates of 0.40, 0.25, and 0.25 for $s1$, $s3$ and $s4$, respectively, with a remaining capacity of 0.10 and 0.35 on Link12 and Link23, respectively.
- *Step* 5: Since $s3$ and $s4$ both have a smaller rate (0.25) than $s1$ (0.4), we increase the rates of $s3$ and $s4$ to 0.35 and Link12 saturates.
- *Step* 6: Remove $s1$ (with a rate of 0.40) and $s3$ (with a rate of 0.35) from future iterations and we now have $s4$ as the remaining session (with a rate of 0.35) and remaining capacity on Link23 is 0.25.
- *Step* 7: Increase the rate of $s4$ to 0.60 and Link23 saturates. The final rates are 0.40, 0.25, 0.35, and 0.60 for $s1$, $s2$, $s3$, and $s4$, respectively.

Formally, we say that a rate vector r is GMM if it is MP-feasible, and for every $s \in \mathscr{S}$ and every MP-feasible rate vector $\hat{r}$ in which $\hat{r}_s > r_s$, there exists some session $t \in \mathscr{S}$ such that $r_s \geqslant r_t$, and $r_t > \hat{r}_t$ [22].

Given an MP-feasible rate vector r, a link $\ell \in \mathscr{L}$ is a GMM-bottleneck link with respect to r for a session s traversing ℓ if $F_\ell = C_\ell$ and $r_s \geqslant r_t$ for every session t traversing link ℓ for which $r_t > \mathrm{MCR}_t$.

It can be shown that an MP-feasible rate vector r is GMM if and only if each session has either a GMM-bottleneck link with respect to r or a rate allocation equal to its PR [22].

In Example 2, Link12 is a GMM-bottleneck link for both $s1$ and $s3$. On the other hand, $s1$ and

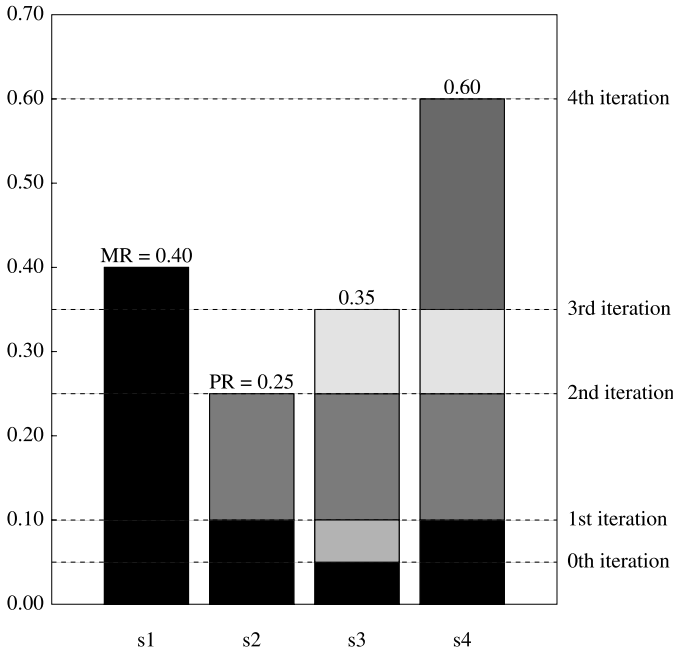


Fig. 2. Rate allocation for each session at each iteration for the GMM policy in the three-node network.

$s3$ have different rate allocation (0.4 for $s1$ and 0.35 for $s3$). Thus, it is essential to have a precise definition of *GMM-bottleneck link rate* here.

Let $1^+\{\text{event A}\}$ be the indicator function with the following definition:

$$1^+\{\text{event A}\} = \begin{cases} 1 & \text{if event A is true,} \\ 0 & \text{otherwise.} \end{cases}$$

Given a GMM rate vector r, suppose that link $\ell \in \mathscr{L}$ is a GMM-bottleneck link with respect to r and let τ_ℓ denote the GMM-bottleneck link rate at link ℓ. Then τ_ℓ satisfies

$$\tau_\ell \sum_{i\in\mathscr{U}_\ell} 1^+\{\mathrm{MR}^i \leqslant \tau_\ell\} + \sum_{i\in\mathscr{U}_\ell} \mathrm{MR}^i \; 1^+\{\mathrm{MR}^i > \tau_\ell\} = C_\ell - \sum_{i\in\mathscr{M}_\ell} r_\ell^i, \tag{1}$$

where $\mathscr{U}_\ell$ denotes the set of sessions that are GMM-bottlenecked at link ℓ, and $\mathscr{M}_\ell$ denotes the set of sessions that are either GMM-bottlenecked elsewhere or have a rate allocation equal to their PRs and $r_\ell^i < \tau_\ell$ for every $i \in \mathscr{M}_\ell$.

With the above clarification, it is easy to show that in Example 2 the GMM-bottleneck link rates are 0.35 at Link12 and 0.60 at Link23.

Note that in the special case when $\mathrm{MR}^s = 0$ for every $s \in \mathscr{S}$, the GMM-bottleneck link rate τ_ℓ in Eq. (1) becomes:

$$\tau_\ell \; |\mathscr{U}_\ell| = C_\ell - \sum_{i\in\mathscr{M}_\ell} r_\ell^i, \quad \text{or} \quad \tau_\ell = \frac{C_\ell - \sum_{i\in\mathscr{M}_\ell} r_\ell^i}{|\mathscr{U}_\ell|},$$

where $|\mathscr{U}_\ell|$ denotes the number of sessions bottlenecked at link ℓ. This is precisely the expression for the classical max–min bottleneck link rate definition at a saturated link ℓ.

It should be clear that by Algorithm 2 and the GMM-bottleneck link rate definition in Eq. (1), the GMM rate allocation for a session $s \in \mathscr{S}$ can only be one of the following: (1) A rate equal to its MR; or (2) A rate equal to its PR; or (3) A rate equal to its GMM-bottleneck link rate.

2.3. Weight-proportional max–min

This is another way (perhaps most popular in terms of its many variants) to extend the classical max–min rate allocation with minimum rate and peak rate constraints. Under this policy, we associate each connection $s \in \mathscr{S}$ with a weight (or

priority) w_s. [2] Informally, the WPMM policy first allocates to each connection its MR. Then from the remaining network capacity, it allocates additional bandwidth for each connection using a proportional version of the max–min policy based on each connection's weight while satisfying its PR constraint. The final bandwidth for each connection is its MR plus an additional "weighted" rate share [24].

The WPMM rate allocation policy presented here generalizes the so-called *MCRadd* and *MCRprop* policies [26,45], which are quite popular. Both MCRadd and MCRprop first guarantee the minimum rate of each connection. Under MCRadd, the remaining network bandwidth is shared among all connections using the max–min policy, i.e., equal weight for all connections; while under MCRprop, the remaining bandwidth is shared among all connections using MCR-proportional max–min policy. Both the MCRadd and MCRprop policies are special cases of WPMM policy since the weight for each connection under WPMM can be arbitrary positive number and independent of (decoupled from) its MR or PR.

The following algorithm shows how to compute rate allocation for each connection under the WPMM policy.

Algorithm 3 (*WPMM rate allocation*).

1. Start the rate allocation of each connection with its MR.
2. Increase the rate of each connection with an increment proportional to its weight until either some link becomes saturated or some connection reaches its PR, whichever comes first.
3. Remove those connections that either traverse saturated links or have reached their PRs and the capacity associated with such connections from the network.
4. If there is no connection left, the algorithm terminates; otherwise, go back to Step 2 for the remaining connections and remaining network capacity.

[2] We assume a positive weight assignment for each connection.

The following example illustrates how the WPMM rate allocation works.

Example 3 (*WPMM rate allocation*). Again, we use the four-session three-node network in Fig. 1. The minimum rate requirement, peak rate constraint and weight for each session are listed in Table 2, as well as the WPMM rate allocation for each session. Table 3 shows the results of rate allocation for each session at the end of each iteration of Algorithm 3, which are described as follows.

- *Step* 1: As shown in the initialization procedure of Table 3, we start the rate of each session with its MR.
- *Step* 2: We increase the rate of each session with an increment proportional to its weight (1, 3, 4, and 2 for $s1$, $s2$, $s3$ and $s4$, respectively) until session $s3$ reaches its PR constraint (0.40).
- *Step* 3: Remove $s3$ (with a rate of 0.40) from future iterations and we now have the rates of 0.10, 0.30, and 0.20 for $s1$, $s2$ and $s4$, respectively, with a remaining capacity of 0.20 and 0.70 on Link12 and Link23, respectively.
- *Step* 4: We increase the rates of the remaining sessions ($s1$, $s2$, and $s4$), each with an increment proportional to its weight until Link12 saturates.
- *Step* 5: Remove $s1$ (with a rate of 0.15) and $s2$ (with a rate of 0.45) from future iterations and we now have $s4$ as the remaining session (with a rate of 0.30) and remaining capacity on Link23 is 0.55.
- *Step* 6: Increase the rate of $s4$ to 0.85 and Link23 saturates. The final rates are 0.15, 0.45, 0.40, and 0.85 for $s1$, $s2$, $s3$, and $s4$, respectively.

Table 2
Minimum rate requirement, peak rate constraint, weight, and rate allocation for each session for the WPMM policy in the three-node network

Session	MR	PR	Weight	WPMM rate allocation
$s1$	0.05	0.75	1	0.15
$s2$	0.15	0.90	3	0.45
$s3$	0.20	0.40	4	0.40
$s4$	0.10	1.00	2	0.85

Table 3
Rate allocation for each session after each iteration under WPMM algorithm in the three-node network

Iterations	Session{(MR, PR), w}				Remaining capacity	
	$s1$ {(0.05, 0.75), 1}	$s2$ {(0.15, 0.90), 3}	$s3$ {(0.20, 0.40), 4}	$s4$ {(0.10, 1.00), 2}	Link12	Link23
Initialization	0.05	0.15	0.20	0.10	0.60	0.85
First	0.10	0.30	0.40	0.20	0.20	0.70
Second	0.15	0.45		0.30	0	0.55
Third				0.85		0

Formally, we say that a rate vector r is WPMM if it is MP-feasible, and for each $s \in \mathscr{S}$ and every MP-feasible rate vector $\hat{r}$ in which $\hat{r}_s > r_s$, there exists some connection $t \in \mathscr{S}$ such that $(r_s - \mathrm{MR}_s)/w_s \geqslant (r_t - \mathrm{MR}_t)/w_t$ and $r_t > \hat{r}_t$ [24].

Given an MP-feasible rate vector r, a link $\ell \in \mathscr{L}$ is an *WPMM-bottleneck link* with respect to r for a connection s traversing ℓ if

$$F_\ell = C_\ell \quad \text{and} \quad \frac{r_s - \mathrm{MR}_s}{w_s} \geqslant \frac{r_t - \mathrm{MR}_t}{w_t}$$

for all connections t traversing link ℓ.

It can be shown that an MP-feasible rate vector r is WPMM if and only if each connection has either an WPMM-bottleneck link with respect to r or a rate assignment equal to its PR. In the special case, when (1) each session's MR is zero; (2) there is no PR constraint; and (3) each session has equal weight, the WPMM rate allocation degenerates into the classical max–min rate allocation.

Thus far, we have presented two rate allocation polices that extend the classical max–min. In the next section, we will focus on the design of distributed rate allocation algorithms to achieve these policies in a fully distributed networking environment. In particular, we will study the explicit rate feedback control algorithms.

3. Explicit rate feedback control algorithms

There have been extensive efforts on the design of distributed algorithms to achieve the classical max–min rate allocation. The algorithms by Hayden [20], Jaffe [27], Gafni [16], and Abraham [1] required synchronization of all nodes for each iteration, which is impractical in real world packet networks. Mosely's work in [33] was the first asynchronous algorithm. Unfortunately, this algorithm could not offer satisfactory convergence performance. The algorithms by Ramakrishnan et al. [36] and Yin [44] relied on using a single bit as feedback to indicate congestion. Due to the binary nature of such algorithms, the source's rate exhibited oscillations.

Recent interests in ABR service have led to many contributions to the design of distributed algorithms to achieve the classical max–min. In this section, we first briefly describe the rate-based flow control mechanism, where special control packets are employed to exchange the rate information between a source and the network. Since the purpose of this paper is to share our experience on the design of distributed algorithms that have good convergence property to either GMM or WPMM rate allocation, we will focus only on the so-called *explicit rate* feedback control algorithms.

We outline two broad classes of explicit rate calculation algorithms. Both algorithms have the common property that only a simple shared buffer and FIFO scheduling are required at the switch output port. We will discuss the design tradeoffs between these two classes of algorithms in terms of performance objectives and implementation complexity. After we show the fundamental properties of each class of algorithms, it becomes straightforward for us to discuss how more sophisticated buffering strategy (e.g., per flow queuing) and scheduling policies can be incorporated to further improve the performance of each class of algorithms.

3.1. Rate-based flow control mechanism

It is clear that any distributed flow control algorithm achieving the GMM or WPMM policy must employ some kind of cooperation between the sources and the network. Such cooperation

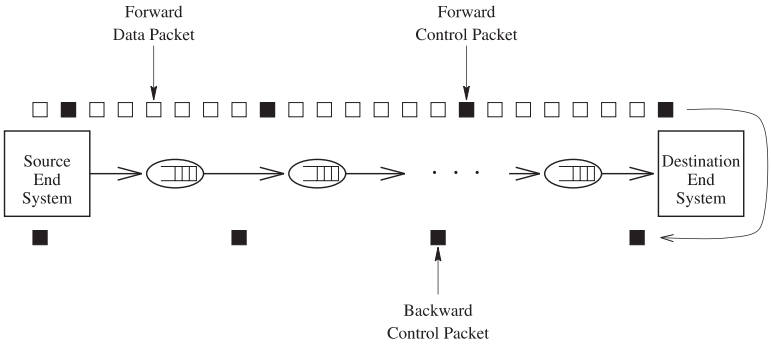


Fig. 3. Rate-based end-to-end flow control mechanism for a session.

should include the following two key components: (1) Information exchange between each source and the network; and (2) Source rate adaptation upon receiving feedback from the network.

Our design of distributed flow control algorithm maintains some link controls at each switch and conveys information about these controls to the source via feedback. Upon the receipt of the feedback signal, the source adjusts its estimate of the allowed transmission rate. Specifically, we employ an end-to-end closed-loop feedback scheme, where control packets are used in both forward and backward paths. A source uses control packets in the forward path to inform the switches about the source's rate information. The switches perform calculations and use the control packets in the backward path to advise the sources on how to adjust their rates. The goal is to properly design this flow control protocol so that eventually each source's rate conforms to the predefined rate allocation policy (i.e., GMM or WPMM).

Since a major driving force behind recent research efforts on rate-based feedback control came from the ATM forum, we will use ABR protocol and terminology when we illustrate specific algorithms. As we shall see, the design methodology discussed here is very general and is applicable to any flow oriented packet-switched network. [3]

[3] In IP networks, the path for a flow may be set up and maintained by multi-protocol label switching [38].

A rate-based flow control is shown in Fig. 3. Special control packets (also called Resource Management (RM) cells in ABR) are employed and are inserted periodically among data packets to exchange information between a source and the network. Note that these control packets are used once every, say N_{rm}, data packets, rather once for every fixed time interval. Such choice will guarantee that the control overhead for each session is fixed at a marginal percentage (i.e., $1/(N_{rm}+1)$) and thus is scalable with network capacity as the number of sessions in the network increases.

To achieve information exchange, the source sets the fields in the forward RM cells to inform the network about the source's rate information (e.g., MCR, PCR, CCR) and weight assignment. For each traversing RM cell at a switch, the switch performs rate calculation based on the information carried in this RM cell. We let the network (switches) set the fields in the backward RM cells to inform the source. To achieve source rate adaptation, the source adjusts its transmission rate upon receiving a backward RM cell.

3.2. A class of heuristic algorithms

Algorithms in this category are designed to approximate the desired rate allocation for each session by using congestion information (e.g., queue length, target queue threshold, load) in conjunction with the CCR value available in the RM cells [6,13,34,37,39]. A switch maintains a running average variable to calculate the share rate

for each session, based on the level of congestion and CCR. This class of algorithms is based on simple heuristic to achieve the desired rate allocation and does not require the switch to maintain a table and to keep track of the state information of each traversing flow.

The *Intelligent Marking* technique by Siu and Tzeng, originally proposed in [39], and further refined in [40,42] best represents the properties of this class of algorithms. We use this simple scheme to illustrate the one end of the spectrum (with the other end taken by per-flow based schemes that will be discussed in Section 3.3).

3.2.1. Intelligent marking for max–min

The key idea of Intelligent Marking technique is to employ a small number of variables and use a small number of computations at each switch output port to estimate the max–min bottleneck link rate. Using a simple feedback mechanism, the ER field of a returning RM cell is set to the minimum of all the estimated bottleneck link rates on all its traversing links to approximate max–min share.

Fig. 4 illustrates the switch behavior of the Intelligent Marking technique [40,42]. Four variables MCCR (Mean CCR), upper cell rate (UCR), estimated bottleneck rate (EBR) and LOAD are defined for the following purpose: (1) MCCR contains an estimated average cell rate of all VCs traversing this link; (2) UCR contains an estimated upper limit of the cell rates of all VCs traversing this link; (3) EBR contains an estimated bottleneck link rate; and (4) LOAD corresponds to the aggregated cell rate entering the queue normalized with respect to the link capacity and is measured over a period of time. Furthermore, two parameters TLR and α are defined at each output port, where the value of TLR is the desired or Targeted Load Ratio ($0 < \mathrm{TLR} \leqslant 1$) and $0 < \alpha < 1$.

The Intelligent Marking algorithm is a heuristic algorithm. We can only give an intuitive explanation on how it works. The RM cells from all VCs participate in the exponential averaging for MCCR with $\mathrm{MCCR} := \mathrm{MCCR} + \alpha(\mathrm{CCR} - \mathrm{MCCR})$ while only those VCs with CCR greater than MCCR (potentially VCs bottlenecked at *this* link) participate in UCR averaging. EBR is used to estimate the max–min bottleneck link rate and is based on UCR and LOAD variables. Since (1) there can be only one max–min bottleneck rate at a link and it is greater than or equal to any of the VC's rate traversing this link; and (2) the returning RM cell's ER field is set to the minimum of all the bottleneck link rates along its path, the final rate allocation through the Intelligent Marking approximates max–min share rate for each flow.

Another interesting fact is that the MCCR is larger than the algebraic average of each VCs CCR traversing this link. This is because MCCR is updated more frequently by those VCs with relatively larger CCR than those with relatively smaller CCR traversing the same link.

The most attractive feature of the Intelligent Marking technique is its low implementation complexity. It does not require each output port of a switch to keep track of each traversing flow's state information (so called per flow accounting)

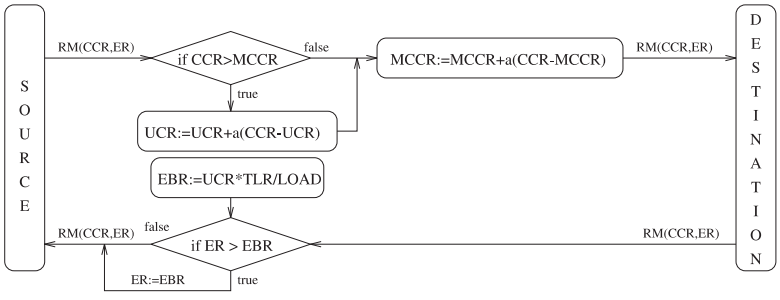


Fig. 4. Switch behavior of the Intelligent Marking protocol.

and has O(1) storage requirements and computational complexity.

It has been shown that the Intelligent Marking technique can be extended to support both GMM and WPMM rate allocation policies in [21,23], respectively. Due to paper length limitation, we will illustrate here how to extend Intelligent Marking for the WPMM rate allocation [23].

3.2.2. Extending intelligent marking for WPMM

We first specify the source and destination's behaviors of each connection.

Algorithm 4 (*End system behavior*).
Source behavior: [4]

1. The source starts to transmit at ACR := ICR, which is greater than or equal to its MCR;
2. For every N_{rm} transmitted data cells, the source sends a forward RM(CCR, MCR, ER, w) cell with: CCR := ACR; MCR := MCR; ER := PCR; $w := w$;
3. Upon the receipt of a backward RM(CCR, MCR, ER, w) cell from the destination, the ACR at the source is adjusted to: ACR := $\max\{\min\{(\text{ACR} + \text{AIR}), \text{ER}\}, \text{MCR}\}$.

Destination behavior: The destination end system of a connection simply returns every RM cell back towards the source upon receiving it.

Since the WPMM policy first allocates each session with its MCR, and then allocates the remaining network bandwidth to each session using the w-proportional max–min policy (Algorithm 3), this motivates us to let the CCR and ER fields of a traversing RM cell be first offsetted by its MCR, and then normalized with respect to the connection's weight w (i.e., $(\text{CCR} - \text{MCR})/w$, $(\text{ER} - \text{MCR})/w$) to participate in the Intelligent Marking algorithm.

Note that we let the source set the weight of a connection into some unspecified field in the forward RM cell. Therefore, in a manner similar to the Intelligent Marking technique, there is no need here to use per flow accounting to keep track of the weight information of each flow at the switch output port.

Fig. 5 illustrates a switch algorithm for the WPMM policy. Four variables named LOAD, normalized mean rate (NMR), normalized upper rate (NUR) and normalized bottleneck rate (NBR) are defined at each output port of a switch. The value of LOAD corresponds to the aggregated cell rate entering the output queue normalized with respect to the link capacity. It is measured at the switch output port over a period of time. The value of NMR contains an exponential averaging of $(\text{CCR} - \text{MCR})/w$ for all VCs traversing this link; the value of NUR contains an exponential averaging of $(\text{CCR} - \text{MCR})/w$ only for VCs with $(\text{CCR} - \text{MCR})/w > \text{NMR}$; and NBR contains an estimated normalized WPMM bottleneck link rate. Here, NMR, NUR and NBR are all dimensionless. TLR is the Targeted Load Ratio ($0 < \text{TLR} \leqslant 1$) at the switch output port and $0 < \alpha < 1$.

Algorithm 5 (*Switch behavior for WPMM rate allocation*).

Upon the receipt of RM (CCR, MCR, ER, w) from the source of a VC
  if $(\text{CCR} - \text{MCR})/w > \text{NMR}$, then
    NUR := NUR+
    $\alpha[(\text{CCR} - \text{MCR})/w - \text{NUR}]$;
  NMR := NMR+
  $\alpha[(\text{CCR} - \text{MCR})/w - \text{NMR}]$;
  Forward RM(CCR, MCR, ER, w) to its destination;
Upon the receipt of RM(CCR, MCR, ER, w) from the destination of a VC
  NBR := NUR $*$ TLR/LOAD;
  if (QS > QT), [5] then NBR := (QT/QS) $*$ NBR;

[4] We use a simplified version of source and destination behavior, which does not include the use-it-or-lose-it option [4]. The AIR parameter is also denoted as PCR · RIF in [4].

[5] This step is a finer adjustment of NBR calculation based on buffer occupancy information and is not shown in Fig. 5 due to space limitation. QS is the Queue Size of the output link and QT is a predefined Queue Threshold.

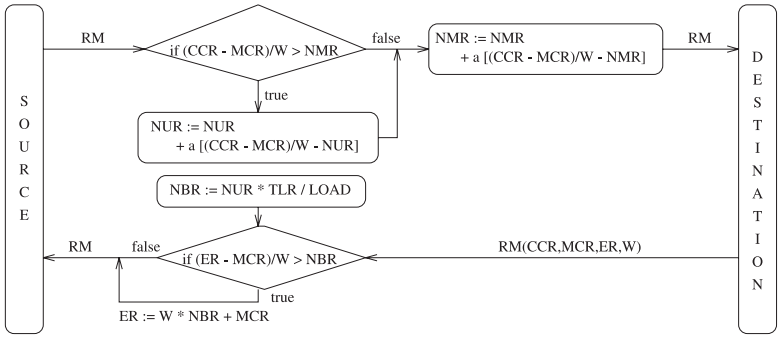


Fig. 5. Switch behavior for the WPMM policy.

if $(\mathrm{ER} - \mathrm{MCR})/w > \mathrm{NBR}$, then $\mathrm{ER} := w \times \mathrm{NBR} + \mathrm{MCR}$;
Forward RM(CCR, MCR, ER, w) to its source.

We use a set of simulation results to demonstrate the performance of this distributed algorithm for the WPMM rate allocation. The network configuration is the four-session three-node network shown in Fig. 1, with the minimum rate requirement and peak rate constraint for each session listed in Table 2. Table 4 lists the parameters used in the simulation. The distance from an end system (source or destination) to the switch is 100 m and the link distance between switches is 10 km (corresponding to a local area network) and the propagation delay is assumed to be 5 μs per km. The initial values of NMR and NUR at each switch output port are set to 0.

Table 4
Simulation parameters

End system	PCR	PCR
	MCR	MCR
	ICR	MCR
	N_{rm}	32
	AIR ($= \mathrm{PCR} \cdot \mathrm{RIF}$)	3.39 Mbps
Link	Speed	150 Mbps
Switch	Cell switching delay	4 μs
	TLR	1
	α	0.125
	Load/utilization measurement interval	500 μs
	Queue threshold for ER adjustment	50 cells
	Output buffer size	2000 cells

Simulation results for the cell rate of each session, the bottleneck link utilization and buffer occupancy are shown in Fig. 6. We see that after the initial transient period, the cell rate of each session matches with the rate listed in Table 2. Also, the bottleneck links are 100% utilized with reasonably small buffer occupancies.

For a wide area network (WAN), this simple heuristic algorithm shown here for WPMM and Class 1 algorithms in general require careful system parameter tuning to minimize oscillations. On the other hand, a more sophisticated algorithm using per flow accounting will be much more effective, as we will discuss in the next section. But in a LAN environment, where implementation complexity may well be the most important criterion in the choice of a switch algorithm, Class 1 algorithms offer satisfactory performance with minimal implementation complexity.

3.3. Using per flow management

The distributed flow control algorithms that fall into this category, as the name implies, employ a table at each output port of a switch to keep track of the state information of each flow [3,8,10,17,28,30,35,41]. In particular, the algorithm by Charny et al. [10] was one of the few algorithms that were proven to converge to max–min through distributed and asynchronous iterations. This algorithm has been widely referred in the literature

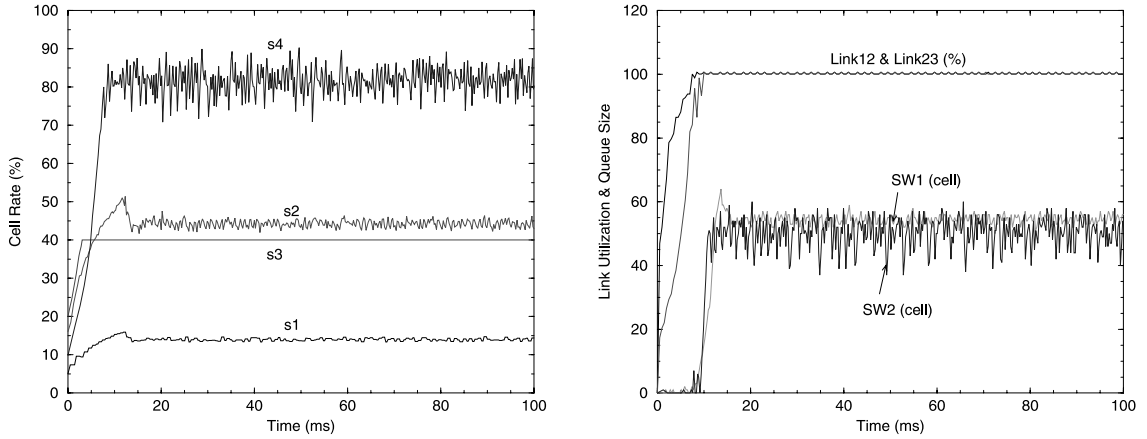


Fig. 6. The normalized cell rates of all connections, the link utilization and the queue size of the congested switches in the three-node network configuration.

and is regarded as a major milestone in the design of rate-based control algorithm for max–min. We will use this algorithm as an example to show the fundamental properties of this class of algorithms that employ per flow accounting.

In Charny's algorithm, each switch monitors its traffic by keeping track of the state information of each traversing connection. Also, each output port of a switch maintains a variable called the *advertised rate* to calculate available bandwidth for each connection. When an RM cell arrives at the switch, the CCR value of the connection is stored in a VC table. If this CCR value is less than or equal to the current advertised rate, then the associated connection is assumed to be bottlenecked either at this link or elsewhere and a corresponding bit for this connection is marked at the VC table. Then the following equation is used to update the advertised rate:

$$\text{advertised rate} = \frac{C_\ell - \sum \text{rates of marked connections}}{n_\ell - \sum \text{marked connections}}, \tag{2}$$

where C_ℓ and n_ℓ are the link capacity and the number of connections at link ℓ. Then the VC table is examined again. For each marked session, if its recorded CCR is larger than this newly calculated advertised rate, this session is then unmarked and the advertised rate is calculated again. The ER field of an RM cell is then set to the minimum of all advertised rates along its traversing links. Upon convergence, each session is allocated with a max–min rate and is marked along every link it traverses.

It has been shown that the above session marking technique can be extended to design distributed flow control algorithms for both the GMM and WPMM policies in [22,24], respectively. Due to paper length limitation, we will only illustrate how to extend Charny's session marking technique to achieve the GMM rate allocation [22].

To extend Charny's algorithm for GMM, it is obvious the advertised rate calculation in Eq. (2) has to be modified to reflect the GMM-bottleneck link rate calculation in Section 2.2. However, with such a GMM-bottleneck link rate definition, it is not clear how to perform session marking for each traversing session. If we mark a session when its CCR is less than or equal to the advertised rate as in Charny's technique, this may bring the advertised rate into a state of oscillation that will never converge (due to some session having a large MCR)!

A deeper look at Charny's original algorithm for max–min shows that *a session traversing its own max–min bottleneck link does not need to be marked at this link*. That is, at a saturated link, only sessions bottlenecked elsewhere need to be marked. A small modification as it may appear to be, this new marking criterion brings a whole

new marking property for sessions upon convergence. In fact, this is the key to resolve the difficulty of marking sessions that are GMM-bottlenecked at the same link but with different rates. In conjunction with the GMM-bottleneck link rate definition and advertised rate calculation, this new marking technique leads to a fundamental generalization of Charny's Consistent Marking technique [22].

We first specify the end system behavior of the protocol.

Algorithm 6 (*End system behavior*).
Source behavior:
1. The source starts to transmit at ACR := ICR, which is greater than or equal to its MCR;
2. For every N_{rm} transmitted ATM data cells, the source sends a forward RM(CCR, MCR, ER) cell with: CCR := ACR; MCR := MCR; ER := PCR;
3. Upon the receipt a backward RM(CCR, MCR, ER) cell from the destination, the ACR at source is adjusted to: ACR := ER.

Destination behavior: The destination returns every RM cell back towards the source upon receiving it.

The switch maintains a table at each output port to keep track of the state information of each traversing flow (so-called per flow accounting) and performs the switch algorithm (Algorithm 7) at this output port.

The following are the link parameters and variables used in the switch algorithm:

C_ℓ	capacity of link ℓ, $\ell \in \mathscr{L}$
RC_ℓ	remaining capacity variable at link ℓ used for μ_ℓ calculation in Algorithm 8
$\mathscr{S}_\ell$	set of sessions traversing link ℓ, $\ell \in \mathscr{L}$
n_ℓ	number of sessions in $\mathscr{S}_\ell$, $\ell \in \mathscr{L}$, i.e., $n_\ell = \lvert\mathscr{S}_\ell\rvert$
r_ℓ^i	CCR value of session $i \in \mathscr{S}_\ell$ at link ℓ
MCR^i	MCR requirement of session i
b_ℓ^i	bit used to mark session $i \in \mathscr{S}_\ell$ at link ℓ. $b_\ell^i = 1$ if session $i \in \mathscr{S}_\ell$ is marked at link ℓ, or 0 otherwise
$\mathscr{M}_\ell$	set of sessions marked at link ℓ, i.e., $\mathscr{M}_\ell = \{ i \mid i \in \mathscr{S}_\ell \text{ and } b_\ell^i = 1\}$
$\mathscr{U}_\ell$	set of sessions unmarked at link ℓ, i.e., $\mathscr{U}_\ell = \{ i \mid i \in \mathscr{S}_\ell \text{ and } b_\ell^i = 0\}$, and $\mathscr{M}_\ell \cup \mathscr{U}_\ell = \mathscr{S}_\ell$
μ_ℓ	advertised rate at link ℓ.

The following two algorithms show the switch behavior for the GMM rate allocation, with each output link $\ell \in \mathscr{L}$ initialized with $\mathscr{S}_\ell = \emptyset$; $n_\ell = 0$; $\mu_\ell = C_\ell$.

Algorithm 7 (*Switch behavior for GMM rate allocation*).

```
Upon the receipt of a forward RM(CCR, MCR,
ER) cell from the source of session i {
  if RM cell signals session exit [6] {
    S_l := S_l - {i}; n_l := n_l - 1;
    table_update();
    }
  if RM cell signals session initiation {
    MCR^i := MCR;
    /* Insert a new record for this session in
    the table (a linked list of records) such that
    the MCR fields of the linked list of records
    are in increasing order. */
    S_l := S_l ∪ {i};   n_l := n_l + 1;
    r_l^i := CCR;   b_l^i := 0;
    table_update();
    }
  else /* i.e., RM cell belongs to an ongoing
  session. */ {
    r_l^i := CCR;
    if (r_l^i < μ_l) then b_l^i := 1;
    table_update();
    }
  Forward RM(CCR, MCR, ER) towards its
  destination;
}
Upon the receipt of a backward RM(CCR,
MCR, ER) cell from the destination of session
i {
  ER := max{min{ER, μ_l}, MCR};
  Forward RM(CCR, MCR, ER) towards its
  source;
}
```

[6] This information is conveyed through some unspecified bits in the RM cell, which can be set either at the source or the UNI.

```
table_update()
{
   rate_calculation_1: use Algorithm 8 to calculate advertised rate μ_ℓ^1;
   Unmark any marked session i ∈ S_ℓ at link ℓ with r_ℓ^i ⩾ μ_ℓ^1;
   rate_calculation_2: use Algorithm 8 to calculate advertised rate μ_ℓ;
   if (μ_ℓ < μ_ℓ^1), then {
      Unmark any marked session i ∈ S_ℓ at link ℓ with r_ℓ^i ⩾ μ_ℓ;
      rate_calculation_3: use Algorithm 8 to calculate advertised rate μ_ℓ again;
      }
}
```

Algorithm 8 (μ_ℓ *Calculation*).

```
if n_ℓ = 0, then μ_ℓ := C_ℓ;
else if n_ℓ = |M_ℓ|, then μ_ℓ := C_ℓ − Σ_{i∈S_ℓ} r_ℓ^i + max_{i∈S_ℓ} r_ℓ^i;
else {
   RC_ℓ := C_ℓ − Σ_{i∈M_ℓ} r_ℓ^i;
   if (RC_ℓ < Σ_{i∈U_ℓ} MCR^i) then μ_ℓ := 0;
   else /* i.e., RC_ℓ ⩾ Σ_{i∈U_ℓ} MCR^i. */ {
      /* Due to the particular VC table creation scheme, the unmarked sessions i ∈ U_ℓ are
      already in increasing order of their MCRs, i.e.,
      MCR[1] ⩽ MCR[2] ⩽ ··· ⩽ MCR[|U_ℓ|]. */
      k := |U_ℓ|;   μ_ℓ := RC_ℓ/k;
      while (μ_ℓ < MCR[k]) {
         RC_ℓ := RC_ℓ − MCR[k];        k := k − 1;
         μ_ℓ := RC_ℓ/k;
         }
      } [7]
   }
```

It can be shown that after the number of active sessions in the network stabilizes, the rate allocation for each session by this distributed rate calculation algorithm converges to the GMM rate allocation [22]. Furthermore, an upper bound for the convergence time to the final GMM rate allocation by this distributed protocol from the time when the number of active sessions in the network stabilizes is given by $2.5KD$, where K is the number of levels of bottleneck link rates and D is an upper bound for the round-trip delay among all sessions [22]. Note that K is bounded by (usually substantially less than) the number of sessions in the network, N.

We use a set of simulation results to illustrate the performance of the above distributed algorithm. The network configuration we use is the same four-session three-node network shown in Fig. 1, with the minimum rate requirement and peak rate constraint for each session listed in Table 1. The link speed is 150 Mbps. For stability, we set the target link utilization to be 0.95. That is, we set $C_\ell = 0.95 \times 150 = 142.5$ Mbps at every link $\ell \in \mathscr{L}$ for the ER calculation. This will ensure that the potential buffer build up during transient period will be eventually emptied upon convergence. The distance from source/destination to the switch is 1 km and the link distance between switches is 1000 km (corresponding to a wide area network) and the propagation delay is assumed to be 5 μs per km. The simulation results for the rate of each session are shown in Fig. 7. We find that after the initial transient period, the rate of each session converges to the GMM rate allocation listed in Table 1 without any oscillations.

Thus far, we have used two specific flow control algorithms (described in Sections 3.2 and 3.3) to illustrate two broad classes of distributed rate calculation algorithms. They show the design tradeoffs between performance objectives and implementation complexity. In the next section, we summarize our experience on the design of such rate calculation algorithms.

4. Design space and tradeoffs

Table 5 summarizes important tradeoffs between two classes of algorithms discussed in this paper. In the following, we elaborate each item

[7] The combined steps in the bracket for "else" are equivalent to finding the GMM-bottleneck link rate μ_ℓ for the set of unmarked sessions $\mathscr{U}_\ell$ such that $\mu_\ell \sum_{i\in\mathscr{U}_\ell} 1^+\{\mathrm{MCR}^i \leqslant \mu_\ell\} + \sum_{i\in\mathscr{U}_\ell} \mathrm{MCR}^i\, 1^+\{\mathrm{MCR}^i > \mu_\ell\} = RC_\ell$. In the special case when $\mathrm{MCR}^i = 0$ for every $i \in \mathscr{U}_\ell$, $\mu_\ell = RC_\ell/|\mathscr{U}_\ell|$, i.e., the max–min share rate.

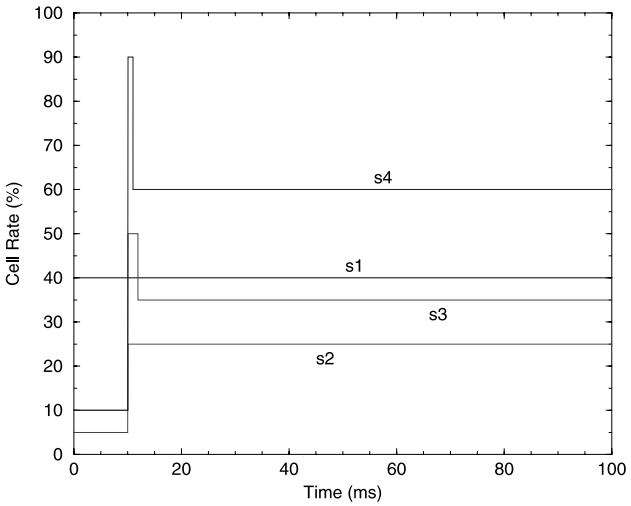


Fig. 7. The normalized cell rates of all connections in the three-node network under the distributed GMM rate calculation algorithm.

Table 5
Design tradeoffs between performance objectives and implementation complexity for the two classes of explicit feedback control algorithms

Type of algorithms		Class 1: exponential averaging without per flow accounting	Class 2: using per flow accounting
Performance features	Convergence property	Approximate	Guaranteed convergence
	Rate decoupling property	No	Yes
	Sensitivity to system parameters	Yes	No
	Applicable networks	LAN	LAN and WAN
Implementation characteristics	State requirement	O(1)	O(N)
	Computational complexity	O(1)	O(N)
	Buffering and scheduling	One shared queue FIFO	One shared queue FIFO

listed in Table 5 and discuss important extensions within each class of algorithms.

4.1. Performance objectives

We compare the two classes of algorithms in terms of convergence property, rate decoupling property, sensitivity to system parameters, and applicable networks as follows.

4.1.1. Convergence property

At steady state, the rate allocated to each flow through a distributed rate control algorithm should match the intended rate allocation policy (e.g., WPMM or GMM) from any initial network conditions.

Class 1 heuristic algorithms, strictly speaking, do not converge to the rate allocation policy. At best, they only approximate to the particular rate allocation objective. The accuracy of such approximation relies on the overall system parameter tuning for the particular network configuration and the set of link distances. On the other hand, Class 2 algorithms can provide guaranteed convergence to the predefined rate allocation policy under any network configuration and any set of link distances.

4.1.2. Rate decoupling property

For Class 2 algorithms discussed in Section 3.3, note that in the source algorithm (Algorithm 6), the ACR of a source is adjusted immediately upon receiving a returning RM cell. A closer look at the mechanics of the switch algorithm (Algorithm 7) reveals that the ACR variable at a source (recorded as CCR in the forward RM cell) is used as

a variable solely for the purpose of distributed protocol convergence iterations and a source's true transmission rate does not affect the convergence property. That is, a source's true transmission rate does not have to be identical to its ACR at all times. For example, as long as a source's true transmission rate is between its MCR and ACR, the overall feedback control protocol can still operate properly (i.e., the ACR for each connection will converge to the optimal rate allocation). This rate decoupling property is a consequence of the special design of the switch algorithm where a table is employed to keep track of the traversing connections and their rate information, and the fact that the buffer occupancy and the load at the output port, and therefore the source's true transmission rate, do not play any role in the ER calculation.

Such rate decoupling property in Class 2 algorithms has important applications in transporting rate-adaptive compressed video using a feedback control mechanism. In [25], we proposed a novel source rate adaptation algorithm, which exploited such decoupling property of a source's true transmission rate and ACR variable used for protocol convergence. We demonstrated that such rate decoupling property, once employed by a source, could make the source's transmission rate converge smoothly to the final optimal rate allocation without undergoing frequent rate fluctuations during a transient period, which is undesirable for video applications.

Class 1 algorithms are unable to offer such a rate decoupling property since the explicit rate calculation for each flow relies on the congestion information (e.g., buffer occupancy, load), which are determined by the source's true transmission rate.

4.1.3. Sensitivity to system parameters

There are many ad hoc system parameters in Class 1 heuristic algorithms such as α for exponential averaging, buffer threshold for ER adjustment and AIR. The performance of Class 1 algorithms under a particular network configuration and the set of link distances are sensitive to the settings of these parameters.

On the other hand, Class 2 algorithms (e.g., Algorithm 7) do not use such ad hoc parameters and have guaranteed performance to the particular rate allocation policy (GMM or WPMM) under any network configuration and any set of link distances.

4.1.4. Applicable networks

Class 1 heuristic algorithms are shown to be a viable solution to achieve the rate allocation in a local area network environment, where the set of link distances are small, and therefore, the systems parameters are fairly easy to set. As the set of link distances increase, the proper setting of system parameters becomes increasingly difficult and thus, the performance of Class 1 algorithms also degrades (e.g., large oscillations in a source's rate). The fundamental difficulty lies in the fact that we use one common shared queue for all flows at a switch output port and it is not possible to isolate flows and tune the system parameters for each individual flow. Later on, we will discuss how per flow queuing may be employed to alleviate this problem and help to improve the performance of Class 1 algorithms in a wide area network.

Since Class 2 algorithms (e.g., Algorithm 7 for GMM) provide guaranteed convergence to the predefined rate allocation policy under any network configuration and any set of link distances, they can be used for both LAN and WAN. The only problem associated with such algorithms is the scalability issue associated with the state table at each output port, which we discuss as follows.

4.2. Implementation complexity

We compare the implementation complexity between the two classes of algorithms in terms of state requirement and scalability, computational complexity, queuing and scheduling as follows.

4.2.1. State requirement and scalability issues

State requirement refers to the number of variables required at each output port of a switch for the purpose of explicit rate calculation. As shown in Section 3.2, Class 1 algorithms such as Algorithm 5 for WPMM require only a constant number of variables at each switch output port and are thus scalable to the number of traversing flows. On the other hand, Class 2 algorithms in

Section 3.3 (e.g., Algorithm 7 for GMM) have to maintain a table for keeping track of the state information of each individual traversing flow. Since each flow occupies one entry in the table, the table size will grow as the number of traversing flows increases.

Such one flow per entry requirement for Class 2 algorithms is driven by the fact that we allow the rate of each flow take any value from a continuous real value interval (and thus infinite states). In [11], a scheme to reduce the state information was introduced by restricting the set of supported rates to a fixed and countable number of discrete states. Such compromise enabled the switch to maintain a fixed size table (determined by the number of discrete rate levels) instead of per flow rates. This is analogous to defining a discrete number of service classes instead of having a continuous range of service class in the broader context of service options in packet-switched networks. Note that such a flow aggregation technique is itself a tradeoff between performance latitude and implementation complexity.

4.2.2. Computational complexity

Computational complexity refers to the number of times and type of mathematical operations required to perform explicit rate calculation for each traversing control packet.

Class 1 exponential averaging based heuristic algorithms have the attractive feature of O(1) computation complexity since only a constant small number of switch variables are used to calculate explicit rate.

Class 2 algorithms such as Algorithm 7 for GMM have $O(N)$ computational complexity. However, if we can compromise the infinite rate states for each flow with a fixed and countable number in a discrete state space, the computational complexity can be reduced to $O(\log(N))$ [11].

4.2.3. Queuing and scheduling

We refer *queuing* as the buffering strategy for the packets arriving at the same output port from multiple input flows and attribute *scheduling* to the service discipline for the packets stored at the output port.

The queuing (or buffering) strategy employed by both Class 1 and Class 2 algorithms is one common shared queue for all flows and the scheduling policy used is the simple first-in-first-out (FIFO) service discipline. It should be clear that since both classes of algorithms are rate calculation algorithms and are not concerned with rate enforcement, many other scheduling policies can also be used. Furthermore, it has been shown that the rate-based feedback control mechanism is robust to occasional loss of packets, i.e., some packets loss will not alter the final rate allocation for each flow and the stability of the algorithm [31].

We would like to point out that if we use a sophisticated buffering strategy such as *per flow queuing* in combination with an appropriate scheduling mechanism, we may design a flow control algorithm to achieve the rate allocation objective [7,14,19,32]. In particular, it has been shown in [14] that by using per flow queuing, greater control can be exercised for each flow and an exponential averaging type heuristic algorithm (Class 1) can be easily extended for rate calculations on each flow for improved performance in a wide area network. This is because per flow queuing enables us to set/tune system parameters for each individual flow. In [7], it has been shown that once flows are isolated with per flow queuing, a control theoretic approach may be employed for rate calculation.

4.3. Other extensions

Even though the specific distributed flow control algorithms that we presented used ABR terminology, it should be clear that the general methodology of this work is very general and is applicable to network traffic management for any flow oriented packet-switched networks. For example, the fixed-sized packet requirement for ATM can be relaxed in packet networks with variable-sized packets without affecting the overall rate convergence property. Also, it is not necessary to have the returning control packets to follow the same path as the forward path. Should the control packets use a different returning path, we can set the explicit rate information in the forward control packets.

Both the WPMM and GMM rate allocation policies support a minimum rate requirement for

each flow. However, it is sometimes difficult for each flow to have an accurate prior estimate of its minimum required rate. Therefore, it will be very useful that a user can renegotiate its minimum rate requirement should he/she find it necessary. The distributed flow control algorithms we presented are capable of supporting such minimum rate renegotiation options [25]. Since a minimum rate guarantee provides some kind of constant bit rate (CBR)-like service for each flow, the minimum rate renegotiation option is very similar to the renegotiation CBR (RCBR) concept introduced in [18]. Unlike RCBR, the distributed flow control algorithms we discussed in this paper are capable of further exploring any additional network bandwidth and can achieve a rate allocation objective (i.e., WPMM or GMM) for each flow.

The rate allocation policies discussed in this paper supports point-to-point packet flow from one source to one destination. It is straightforward to define point-to-multipoint (or multicast) versions of WPMM or GMM rate allocation policies using similar concepts in the centralized rate allocation algorithms, i.e., Algorithm 3 for WPMM and Algorithm 2 for GMM. For the design of distributed flow control algorithms for a multicast rate allocation policy, it has been shown in [43] that under very general settings, a unicast rate-based flow control algorithm can be extended to support multicast rate allocation policy with guaranteed performance and minimal additional complexity in the control packets.

5. Concluding remarks

This paper summarized our experience in the design of network bandwidth allocation policies and distributed rate calculation algorithms for packet-switched networks. A major motivation of this work came from the intense research efforts on ATM ABR service over the past several years. However, this paper is, by no means, an exhaustive review of ABR, which can be found in many overview papers in the literature, but rather, to share with the readers our own experience on some important principles behind such rate-based feedback control. This paper is intended to offer the readers, who already have some familiarity with rate-based flow control, a broader and deeper perspective on such traffic control algorithms.

We examined the classical max–min rate allocation and showed how to extend the classical max–min with two rate allocation policies that support minimum rate and peak rate constraints from each connection. We classified the many well-known algorithms into two broad classes based on how much state information is required at the switch for feedback rate calculation and discussed the tradeoffs between these two classes of algorithms in terms of convergence property, rate decoupling property, sensitivity to system parameters, applications, state requirement, computational complexity, and queuing and scheduling disciplines. We showed that the choice of a particular algorithm is largely a tradeoff between performance objectives and implementation complexities. Also, we discussed how per flow queuing and scheduling discipline may be incorporated into both classes of algorithms for improved performance.

As networking technologies keep advancing and new user application requirements grow, research on feedback-based congestion and flow control will continue to attract interests. We hope the experience we summarized in this paper on rate allocation policies and distributed rate calculation algorithms will serve as a valuable reference for both researchers and network planners when they design or deploy a particular feedback control algorithm for their network.

References

[1] S.P. Abraham, A. Kumar, Max–min fair rate control of ABR connections with nonzero MCRs, in: Proceedings of the IEEE GLOBECOM'97, November 1997, pp. 498–502.

[2] A. Arulambalam, X. Chen, N. Ansari, Allocating fair rates for available bit rate service in ATM networks, IEEE Commun. Magazine (November 1996) 92–100.

[3] A. Arulambalam, X. Chen, N. Ansari, An intelligent explicit rate control algorithm for ABR service in ATM networks, in: Proceedings of the IEEE ICC'97, June 1997, pp. 200–204.

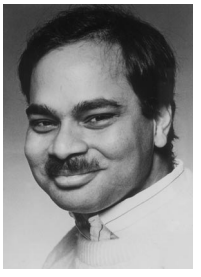
[4] ATM Forum Technical Committee, Traffic Management Specification, Version 4.0, ATM Forum Contribution, AF-TM 96-0056.00, April 1996.

[5] D. Bertsekas, R. Gallager, Data Networks, Prentice Hall, Englewood Cliffs, NJ, 1992 (Chapter 6).
[6] A.W. Barnhart, Explicit Rate Performance Evaluations, ATM Forum Contribution, AF-TM 94-0983, September 1994.
[7] L. Benmohamed, Y.T. Wang, A control-theoretic ABR explicit rate algorithm for ATM switches with per-VC queuing, in: Proceedings of the IEEE INFOCOM'98, March 1998, pp. 183–191.
[8] G. Bianchi, L. Fratta, L. Musumeci, Congestion control algorithms for the ABR service in ATM networks, in: Proceedings of the IEEE GLOBECOM'96, November 1996, pp. 1080–1084.
[9] F. Bonomi, K.W. Fendick, The rate-based flow control framework for the available bit rate ATM service, IEEE Network (March/April 1995) 25–39.
[10] A. Charny, D. Clark, R. Jain, Congestion control with explicit rate indication, in: Proceedings of the IEEE ICC'95, June 1995, pp. 1954–1963.
[11] A. Charny, K.K. Ramakrishnan, A. Lauck, Time scale analysis and scalability issues for explicit rate allocation in ATM networks, IEEE/ACM Trans. Networking 4 (1996) 569–581.
[12] T.M. Chen, S.S. Liu, V.K. Samalam, The available bit rate service for data in ATM networks, IEEE Commun. Magazine (May 1996) 56–71.
[13] F.M. Chiussi, Y. Xia, V.P. Kumar, Dynamic max rate control algorithm for available bit rate service in ATM networks, in: Proceedings of the IEEE GLOBECOM'96, November 1996, pp. 2108–2117.
[14] F.M. Chiussi, Y.T. Wang, An ABR rate-based congestion control algorithm for ATM switches with per-VC queuing, in: Proceedings of the IEEE GLOBECOM'97, November 1997, pp. 771–778.
[15] K.W. Fendick, Evolution of controls for the available bit rate service, IEEE Commun. Magazine (November 1996) 35–39.
[16] E.M. Gafni, The integration of routing and flow control for voice and data in a computer communication network, Ph.D. Thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, 1982.
[17] N. Ghani, J.W. Mark, Dynamic rate-based control algorithm for ABR service in ATM networks, in: Proceedings of the IEEE GLOBECOM'96, November 1996, pp. 1074–1079.
[18] M. Grossglauser, S. Keshav, D. Tse, RCBR: a simple and efficient service for multiple time-scale traffic, IEEE/ACM Trans. Networking 5 (1997) 741–755.
[19] E.L. Hahne, Round-robin scheduling for max–min fairness in data networks, IEEE J. Select. Areas Commun. 9 (1991) 1024–1039.
[20] H.P. Hayden, Voice flow control in integrated packet networks, M.S. Thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, 1981.
[21] Y.T. Hou, H. Tzeng, S.S. Panwar, A generalized max–min network capacity assignment policy with a simple ABR implementation for an ATM LAN, in: Proceedings of the IEEE GLOBECOM'97, November 1997, pp. 503–508.
[22] Y.T. Hou, H. Tzeng, S.S. Panwar, A generalized max–min rate allocation policy and its distributed implementation using the ABR flow control mechanism, in: Proceedings of the IEEE INFOCOM'98, March 1998, pp. 1366–1375.
[23] Y.T. Hou, H. Tzeng, S.S. Panwar, V.P. Kumar, ATM ABR traffic control with a generic weight-based bandwidth sharing policy: theory and a simple implementation, IEICE Trans. Commun. E81-B (1998) 958–972.
[24] Y.T. Hou, H. Tzeng, S.S. Panwar, A generic weight based network bandwidth sharing policy for ATM ABR service, in: Proceedings of the IEEE ICC'98, June 1998, pp. 1492–1499.
[25] Y.T. Hou, S.S. Panwar, Z.-L. Zhang, H. Tzeng, Y.-Q. Zhang, Network bandwidth sharing for transporting rate-adaptive packet video using feedback, in: Proceedings of the IEEE GLOBECOM'98, November 1998, pp. 1547–1555.
[26] D. Hughes, Fair Share in the Context of MCR, ATM Forum Contribution, AF-TM 94-0977, October 1994.
[27] J.M. Jaffe, Bottleneck flow control, IEEE Trans. Commun. COM-29 (1981) 954–962.
[28] R. Jain, S. Kalyanaraman, R. Goyal, S. Fahmy, R. Viswanathan, ERICA Switch Algorithm: A Complete Description, ATM Forum Contribution, AF-TM 96-1172, August 1996.
[29] R. Jain, Congestion control and traffic management in ATM networks: recent advances and a survey, Computer Networks and ISDN Systems 28 (1996) 1723–1738.
[30] L. Kalampoukas, A. Varma, K.K. Ramakrishnan, An efficient rate allocation algorithm for ATM networks providing max–min fairness, in: Proceedings of the Sixth IFIP International Conference on High Performance Networking (HPN'95), September 1995, pp. 143–154.
[31] D. Lee, K.K. Ramakrishnan, W.M. Moh, A.U. Shankar, Performance and correctness of the ATM ABR rate control scheme, in: Proceedings of the IEEE INFOCOM'97, April 1997, pp. 785–794.
[32] D. Lin, Constant-time dynamic ATM bandwidth scheduling for guaranteed and best effort services with overbooking, in: Proceedings of the IEEE INFOCOM'97, April 1997, pp. 398–405.
[33] J. Mosely, Asynchronous distributed flow control algorithms, Ph.D. Thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, 1984.
[34] S. Muddu, F. Chiussi, C. Tryfonas, V.P. Kumar, Max–min rate control algorithm for available bit rate service in ATM networks, in: Proceedings of the IEEE ICC'96, June 1996, pp. 412–418.
[35] S. Prasad, K. Kiasaleh, P. Balsara, LAPLUS: an efficient, effective and stable switch algorithm for flow control of the available bit rate ATM service, in: Proceedings of the IEEE INFOCOM'98, March 1998, pp. 174–182.
[36] K.K. Ramakrishnan, R. Jain, D.-M. Chiu, Congestion avoidance in computer networks with a connectionless network layer – part IV: a selective binary feedback scheme

for general topologies methodology, DEC-TR-510, Digital Equipment Corporation, 1987.
[37] L. Roberts, Enhanced PRCA (Proportional Rate Control Algorithm), ATM Forum Contribution 94-0735R1, August 1994.
[38] E.C. Rosen, A. Viswanathan, R. Callon, Multiprotocol Label Switching Architecture, Internet Draft, work in progress, Internet Engineering Task Force, August 1999.
[39] K.-Y. Siu, H.-Y. Tzeng, Intelligent congestion control for ABR service in ATM networks, ACM SIGCOMM Comput. Commun. Rev. 24 (5) (1994) 81–106.
[40] K.-Y. Siu, H.-Y. Tzeng, Limits of Performance in Rate-based Control Schemes, ATM Forum Contribution, AF-TM 94-1077, November 1994.
[41] D.H.K. Tsang, W.K.F. Wong, A new rate-based switch algorithm for ABR traffic to achieve max–min fairness with analytical approximation and delay adjustment, in: Proceedings of the IEEE INFOCOM'96, March 1994, pp. 1174–1181.
[42] H.-Y. Tzeng, K.-Y. Siu, Comparison of Performance Among Existing Rate Control Schemes, ATM Forum Contribution, AF-TM 94-1078, November 1994.
[43] H.-Y. Tzeng, K.-Y. Siu, On max–min fair congestion control for multicast ABR service in ATM, IEEE J. Select. Areas Commun. 15 (1997) 545–556.
[44] N. Yin, M.G. Hluchyj, On closed-loop rate control for ATM cell relay networks, in: Proceedings of the IEEE INFOCOM'94, June 1994, pp. 99–108.
[45] N. Yin, Max–min fairness vs. MCR guarantee on bandwidth allocation for ABR, in: Proceedings of the IEEE ATM'96 Workshop, San Francisco, CA, August 1996.

Yiwei Thomas Hou obtained his B.E. degree (*Summa Cum Laude*) from the City College of New York in 1991, the M.S. degree from Columbia University in 1993, and the Ph.D. degree from Polytechnic University, Brooklyn, New York, in 1997, all in Electrical Engineering. He was awarded a five-year National Science Foundation Graduate Research Traineeship for pursuing Ph.D. degree in high speed networking, and was recipient of the Alexander Hessel award for outstanding Ph.D. dissertation (1997–1998 academic year) from Polytechnic University. While a graduate student, he worked at AT&T Bell Labs, Murray Hill, New Jersey, during the summers of 1994 and 1995, on internetworking of IP/ATM networks; he conducted research at Lucent Technologies Bell Labs, Holmdel, NJ, during the summer of 1996, on fundamental problems on network traffic management. Since September 1997, Dr. Hou has been a Research Scientist at Fujitsu Laboratories of America, Sunnyvale, California. His current research interests are in the areas of scalable architecture, protocols, and implementations for differentiated services Internet; terabit switching; and quality of service (QoS) support for multimedia over IP networks. Dr. Hou is a member of the IEEE, ACM, Sigma Xi and New York Academy of Sciences.

Bo Li received the B.S. (*cum laude*) and M.S. degrees in Computer Science from Tsinghua University, China, in 1987 and 1989, respectively, and the Ph.D. degree in Computer Engineering from University of Massachusetts at Amherst in 1993. Between 1994 and 1996, he worked on high performance routers and ATM switches in IBM Networking System Division, Research Triangle Park, North Carolina. He joined the faculty of the Computer Science Department of the Hong Kong University of Science and Technology in January 1996. Dr. Li has been on the editorial board for ACM Mobile Computing and Communications Review and Journal of Communications and Networks. He will be serving as an Editor for ACM/Baltzer Journal of Wireless Networks. He served as a Guest Editor for special issues for IEEE Communications Magazine, IEEE Journal on Selected Areas in Communications and the upcoming SPIE/Baltzer Optical Networks Magazine. He has been involved in organizing many conferences such as IEEE Infocom, ICDCS and ICC. He will be the international vice-chair for Infocom'2001. Dr. Li's current research interests include wireless mobile networking supporting multimedia, voice and video (MPEG-2 and MPEG-4) transmission over the Internet and all optical networks using WDM.



Shivendra S. Panwar received the B.Tech. degree in Electrical Engineering from the Indian Institute of Technology, Kanpur, in 1981, and the M.S. and Ph.D. degrees in Electrical and Computer Engineering from the University of Massachusetts at Amherst, in 1983 and 1986, respectively. He joined the Department of Electrical Engineering at the Polytechnic University, Brooklyn, NY, and is now an Associate Professor. Since 1996, he has served as Director of the New York State Center for Advanced Technology in Telecommunications (CATT). His research interests include performance analysis and design of high speed networks. Dr. Panwar is a Senior Member of the IEEE and a member of Tau Beta Pi and Sigma Xi. He was co-editor of two books, Network Management and Control, Vol. II, and Multimedia Communications and Video Coding, both published by Plenum Press, NY.

Henry Tzeng received his B.S. degree from the Tatung Institute of Technologies, Taiwan, Republic of China, in 1988, and his M.S. and Ph.D. degrees in Electrical Engineering from the University of California, Irvine, in 1993 and 1995, respectively. He was a recipient of the University of California Regent's Dissertation Fellowship in 1995 and the 1997 IEEE Browder J. Thompson Memorial Prize Award. Dr. Tzeng is currently Principle Engineer at Amber Networks Inc., Santa Clara, CA, and is working on high performance routing technologies for the Internet. Prior to joining Amber Networks, he was Member of Technical Staff at Lucent Technologies Bell Labs, Holmdel, NJ, during 1995 to 1999. Dr. Tzeng was a Guest Editor for the IEEE Journal on Selected Areas in Communications special issue on next generation IP switches and routers (June 1999).