On Rate Allocation Policies and Explicit Feedback Control Algorithms for Packet Networks

Y. Thomas Hou[a], Henry Tzeng[b], Shivendra S. Panwar[c], and Vijay P. Kumar[b]

[a] Fujitsu Labs of America, Sunnyvale, CA, USA

[b] Bell Labs, Lucent Technologies, Holmdel, NJ, USA

[c] Polytechnic University, Brooklyn, NY, USA

ABSTRACT

This paper presents an in-depth survey on network bandwidth allocation policies and discuss design methodologies of distributed rate calculation algorithms in packet-switched networks. In particular, we discuss two rate allocation policies: the *generalized max-min (GMM)* and the generic *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. We compare the design tradeoffs between these two classes of algorithms in terms of performance objectives and implementation complexities and discuss important extensions within each class of algorithms.

Keywords: Max-min fairness, minimum rate, peak rate, rate allocation policy, congestion and flow control algorithms, explicit rate feedback, ATM, ABR, per flow accounting, per flow queuing, packet switching networks

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 repeat the ATM ABR details and standards, which we refer to overview papers such as Ref. 2,9,12,15,29, but rather, to capture the underlying principles of such rate-based feedback control. This paper is targeted to readers who already have some familiarity with rate-based flow control and would like to have an in-depth understanding of such traffic control algorithms. In particular, we offer a systematic study on the network bandwidth allocation policies and discuss the design methodologies of distributed rate calculation algorithms in the broader context of packet switching networks. We show that the materials covered in this paper is very general and can be applied to any flow oriented packet switching networks, even though for convenience, we use ABR terminology to illustrate a particular distributed rate calculation algorithm.

We start with a brief description 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.* 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), 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 generic weight-proportional max-min (WPMM), 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

*We use the terms "flow", "session", "connection", and "virtual connection" interchangeably throughout the paper.

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. Our goal is to properly design this overall networking protocol so that eventually each source's rate conforms to our rate allocation objective (i.e. GMM or WPMM).

Since the focus of this paper in on the design of distributed rate calculation algorithms that can converge to our GMM or WPMM rate allocation policy, we will consider only the so-called *explicit rate* feedback control algorithms and will not discuss any binary feedback algorithm (e.g. Ref. 36,43), which has rate oscillations and strictly speaking, does not 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 to achieve the rate allocation objective. They do not require the switch to maintain the state information of each traversing flow (also called per VC accounting for ABR). We show that such algorithms provides 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 that the design of these two classes of algorithms are tradeoffs between performance objectives and implementation complexity. Both 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 reminder of this paper is organized as follows. Section 2 presents the generalized max-min (GMM) and the generic weight-proportional max-min (WPMM) policies. Section 3 classifies existing distributed rate calculation algorithms in the literature into two broad classes and discuss 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 4 concludes this paper.

2. RATE ALLOCATION POLICIES

We are interested in optimal allocation of network bandwidth for each user connection in a packet switching network. Specifically, we want to have a rate allocation 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 to all sessions and the network to be utilized as much as possible.

In this section, we first briefly review of 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 the following simple example to illustrate the basic concept of max-min rate allocation.

EXAMPLE 1. As shown in Fig. 1, one session (s_1) flows 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 $\frac{1}{3}$ 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 $\frac{1}{3}$. Session 4 might better limited to $\frac{2}{3}$, 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.

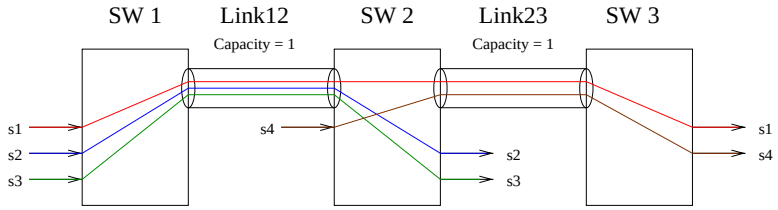

Figure 1. Four sessions sharing a three-node network.

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.

We use the following simple mathematical notation to model max-min rate allocation in a network. We assume 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 $\mathcal{L}$. We denote by r_s 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 $r = \{r_s \mid s \in \mathcal{S}\}$ of allocated rates: 1) $r_s \geq 0$ for all $s \in \mathcal{S}$; and 2) $F_\ell \leq C_\ell$ for all $\ell \in \mathcal{L}$. A vector satisfying these constraints is said to be *feasible*.

A vector of rate r is said to be max-min fair 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 $r_t \leq r_s$. More formally, r is max-min fair 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 $r_s \geq r_t$ 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 $r_s \geq r_t$ 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 fair 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 fair 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. □

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

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 $\frac{1}{3}$ and session 4 gets a rate of $\frac{2}{3}$.

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.

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.,* $\sum_{s \in \mathcal{S}_\ell} \mathrm{MR}_s \leq C_\ell$ for every $\ell \in \mathcal{L}$. This assumption is enforced by admission control at call setup time to determine whether or not to accept a new connection.

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

The generalized max-min (or 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: (1) The rate of such session reaches the second smallest rate among the sessions; or (2) Some link saturates; or (3) 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. We use the same four-session three-node network in Fig. 1. The minimum rate requirement and peak rate constraint for each session are listed in Table 1.

The iterative steps to achieve the GMM rate allocation are listed 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 $s4$).

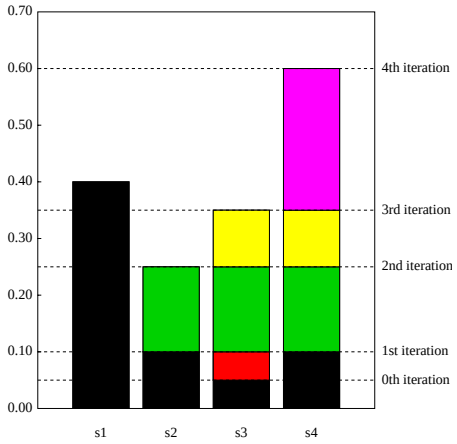

Figure 2. Rate allocation for each session at each iteration for the GMM policy in the three-node network.

- 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. □

REMARK 1. As can be noted in the above example, the GMM policy favors sessions with small MR requirements over those with large MRs in terms of sharing the remaining network capacity (network capacity minus MRs of all sessions). This may appear to be (and in some circumstances is) "unfair" to sessions with large MR requirements. We therefore advocate the use of the GMM policy only when network management policy is to discourage MR-greedy users while being fair to users requesting the smallest required MR for a given application. This situation may arise in networks where users are not explicitly charged for network resources used (e.g. employees within a corporate enterprise network). In such an environment, the GMM policy attempts to achieve equality in bandwidth sharing by first considering the session with the smallest MR. □

Formally, we say that a rate vector r is Generalized Max-Min (GMM) if it is MP-feasible, and for every $s \in \mathcal{S}$ and every MP-feasible rate vector $\hat{r}$ in which $\hat{r}_s > r_s$, there exists some session $t \in \mathcal{S}$ such that $r_s \geq r_t$, and $r_t > \hat{r}_t$.

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

It can be shown that a 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 $s3$ 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 $\ell \in \mathcal{L}$ 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

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

where $\mathcal{U}_\ell$ denotes the set of sessions that are GMM-bottlenecked at link ℓ, and $\mathcal{M}_\ell$ 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 $i \in \mathcal{M}_\ell$.

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 $s \in \mathcal{S}$, the GMM-bottleneck link rate τ_ℓ in Eq. (1) becomes: $\tau_\ell \cdot |\mathcal{U}_\ell| = C_\ell - \sum_{i \in \mathcal{M}_\ell} r_\ell^i$, or $\tau_\ell = \frac{C_\ell - \sum_{i \in \mathcal{M}_\ell} r_\ell^i}{|\mathcal{U}_\ell|}$, where $|\mathcal{U}_\ell|$ denotes the number of sessions bottlenecked at link ℓ. This is exactly 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 $s \in \mathcal{S}$ 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. Generic Weight-Proportional Max-Min

Under this policy, we associate each connection $s \in \mathcal{S}$ with a weight (or priority) w_s.[†] 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.

Our WPMM rate allocation policy presented here generalizes the so-called *MCRadd* and *MCRprop* policies in Ref. 26,44. 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 generically assigned, *i.e.,* independent of (decoupled from) its MR or PR.

The following is a centralized algorithm to compute rate allocation for each connection under the WPMM policy.

ALGORITHM 3. Weight-Proportional Max-Min (WPMM)

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. □

We use the following example to illustrate how the WPMM rate allocation works.

EXAMPLE 3. 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.

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

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 of 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}	Link 12	Link 23
initialization	0.05	0.15	0.20	0.10	0.60	0.85
1st	0.10	0.30	**0.40**	0.20	0.20	0.70
2nd	**0.15**	**0.45**		0.30	0	0.55
3rd				**0.85**		0

- 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 Link 12 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. □

Formally, we say that a rate vector r is *weight-proportional max-min (WPMM)* if it is MP-feasible, and for each $s \in \mathcal{S}$ and every MP-feasible rate vector $\hat{r}$ in which $\hat{r}_s > r_s$, there exists some connection $t \in \mathcal{S}$ such that $\frac{r_s - \mathrm{MR}_s}{w_s} \geq \frac{r_t - \mathrm{MR}_t}{w_t}$ and $r_t > \hat{r}_t$.

Given a MP-feasible rate vector r, a link $\ell \in \mathcal{L}$ is a *WPMM-bottleneck link* with respect to r for a connection s traversing ℓ if $F_\ell = C_\ell$ and $\frac{r_s - \mathrm{MR}_s}{w_s} \geq \frac{r_t - \mathrm{MR}_t}{w_t}$ 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 a 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.

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

3. EXPLICIT 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 switching networks. Mosely's work in Ref. 33 was the first

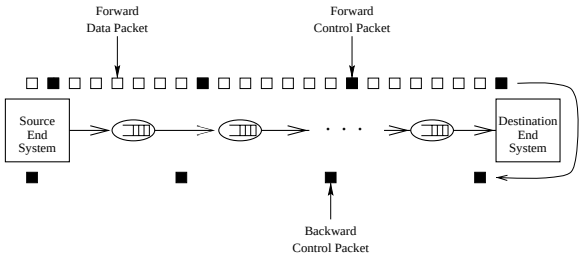

Figure 3. Rate-based end-to-end flow control mechanism for a session.

asynchronous algorithm. Unfortunately, this algorithm could not offer satisfactory convergence performance. The algorithms by Ramakrishnan-Jain-Chiu[36] and Yin[43] relies 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 interest in ATM 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 focus of this paper is on the design of distributed algorithms that have good convergence property to our GMM or WPMM rate allocation, we will consider only the so-called explicit rate feedback control and will not discuss any binary feedback control algorithm, which is oscillatory in nature and has poor convergence property.

We outline two broad classes of explicit rate control 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 term 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 easy to see 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 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 the switch level and conveys information about these controls to the source by means of feedback. Upon the receipt of the feedback signal, the source adjusts its estimate of the allowed transmission rate according to some rule. Specifically, we employ an end-to-end closed-loop feedback scheme, in which 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. Our work is to properly design this overall network protocol so that eventually each source's rate conforms to our predefined centralized objective, *i.e.* the GMM or WPMM rate allocation policy.

Since a major motivation of this work came from the research on the ATM ABR service, we will use some ABR terminologies by ATM Forum when we illustrate various design alternatives and tradeoffs. As we shall see, the design methodology discussed here is very general and is applicable to any flow oriented packet network.

A rate-based closed-loop flow control mechanism is shown in Fig. 3. Special control packets (also called Resource Management (RM) cells in ATM ABR) are employed and are inserted periodically among data packets (or cells for ATM) 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 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

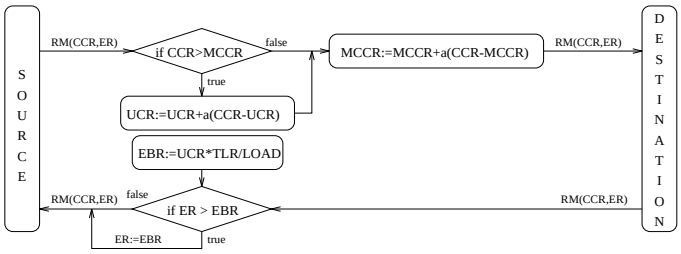

Figure 4. Switch behavior of the Intelligent Marking protocol.

switch, the switch performs some calculations based on the information carried in this RM cell and at the switch. 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 queue length information in conjunction with the CCR value available in the RM cells.[6,13,34,37,38] 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 are based on simple heuristic to achieve the desired rate allocation and do 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 Ref. 38, and further refined in Ref. 39,41 best represents the properties of this class of algorithms. It offers satisfactory performance in achieving the classical max-min policy in a local area environment with minimal implementation complexity. For the purpose of demonstrating the properties of this class of algorithms, we will examine this algorithm in detail.

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 achieve max-min share.

Figure 4 illustrates the switch behavior of the Intelligent Marking technique.[39,41] Four variables MCCR (Mean CCR), UCR (Upper Cell Rate), EBR (Estimated Bottleneck Rate) 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 ($0 < \text{TLR} \leq 1$) 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 $\text{MCCR} := \text{MCCR} + \alpha(\text{CCR} - \text{MCCR})$ 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 achieves max-min share rate for each VC.

Another interesting fact is that the MCCR is larger than the algebraic average of each VC's 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 cost. It does not require each output port of a switch to keep track of each traversing flow's state information (so called per VC accounting) and has $O(1)$ storage requirements and computational complexity.

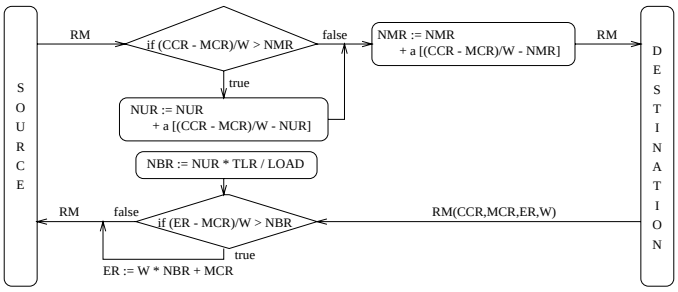

Figure 5. Switch behavior for the WPMM policy.

It has been shown that the Intelligent Marking technique can be extended to support the GMM and WPMM rate allocation policies in Ref. 21 and Ref. 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:[‡]
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, 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. $\frac{\text{CCR}-\text{MCR}}{w}$, $\frac{\text{ER}-\text{MCR}}{w}$) 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.

Figure 5 illustrates our switch algorithm for the WPMM policy. Four variables named LOAD, NMR (Normalized Mean Rate), NUR (Normalized Upper Rate) and NBR (Normalized Bottleneck Rate) are defined at each output port of an ATM 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 ($0 < \text{TLR} \leq 1$) 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 $(\text{CCR} - \text{MCR})/\text{w} > \text{NMR}$, then NUR := NUR + α [(CCR − MCR)/w - NUR];
NMR := NMR + α [(CCR − MCR)/w - NMR];
Forward RM(CCR, MCR, ER, w) to its destination;

[‡] 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 Ref. 4.

Table 4. Simulation parameters.

End System	PCR	PCR
	MCR	MCR
	ICR	MCR
	Nrm	32
	AIR (= PCR · RIF)	3.39 Mbps
Link	Speed	150 Mbps
Switch	Cell Switching Delay	4 μSec
	TLR	1
	α	0.125
	Load/Utilization Measurement Interval	500 μSec
	Queue Threshold for ER Adjustment	50 cells
	Output Buffer Size	2000 cells

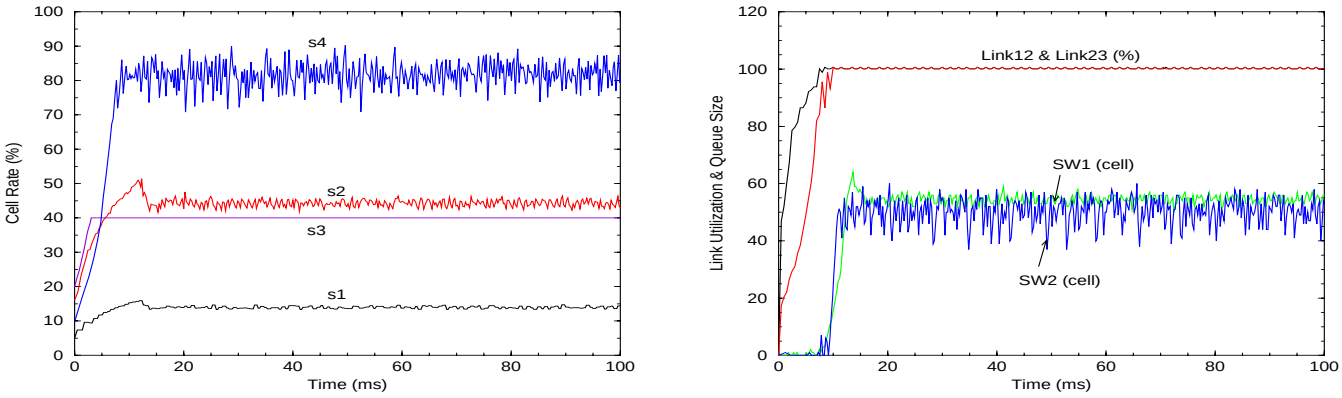

Figure 6. The cell rates of all connections, the link utilization and the queue size of the congested switches in the three-node network configuration.

Upon the receipt of RM(CCR, MCR, ER, w) from the destination of a VC
NBR := NUR * TLR / LOAD;
if (QS > QT),[§] then NBR := (QT / QS) * NBR;
if (ER − MCR)/w > NBR, then ER := w × NBR + MCR;
Forward RM(CCR, MCR, ER, w) to its source. □

We use a set of simulation results to demonstrate the performance of our 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 our 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 we assume that the propagation delay is 5 μs per km. The initial values of NMR and NUR at each switch output port are set to 0.

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), our simple heuristic algorithm shown here for WPMM and Class 1 algorithms in general require careful system parameter tuning to minimize oscillations. Thus, a more sophisticated algorithm using per flow accounting will be much more effective, as we shall show in the next section. But in a LAN environment,

[§]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.

where implementation cost 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 Accounting

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.[3,8,10,17,28,30,35,40] In particular, the algorithm by Charny *et al.* in Ref. 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 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 a 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 Ref. 22 and Ref. 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 our 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 VC accounting) and performs the switch algorithm (Algorithm 7) at this output port.

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

C_ℓ: Capacity of link ℓ, $\ell \in \mathcal{L}$.
RC_ℓ: Remaining Capacity variable at link ℓ used for μ_ℓ calculation in Algorithm 8.
$\mathcal{S}_\ell$: Set of sessions traversing link ℓ, $\ell \in \mathcal{L}$.
n_ℓ: Number of sessions in $\mathcal{S}_\ell$, $\ell \in \mathcal{L}$, i.e., $n_\ell = |\mathcal{S}_\ell|$.
r_ℓ^i: CCR value of session $i \in \mathcal{S}_\ell$ at link ℓ.
MCR^i: MCR requirement of session i.
b_ℓ^i: Bit used to mark session $i \in \mathcal{S}_\ell$ at link ℓ. $b_\ell^i = 1$ if session $i \in \mathcal{S}_\ell$ is marked at link ℓ, or 0 otherwise.
$\mathcal{M}_\ell$: Set of sessions marked at link ℓ, i.e. $\mathcal{M}_\ell = \{i \,|\, i \in \mathcal{S}_\ell \text{ and } b_\ell^i = 1\}$.
$\mathcal{U}_\ell$: Set of sessions unmarked at link ℓ, i.e. $\mathcal{U}_\ell = \{i \,|\, i \in \mathcal{S}_\ell \text{ and } b_\ell^i = 0\}$, and $\mathcal{M}_\ell \cup \mathcal{U}_\ell = \mathcal{S}_\ell$.
μ_ℓ: Advertised rate at link ℓ.

The following algorithm shows the switch behavior for our GMM rate allocation, with each output link $\ell \in \mathcal{L}$ initialized with: $\mathcal{S}_\ell = \emptyset$; $n_\ell = 0$; $\mu_\ell = C_\ell$.

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[¶] {
        $\mathcal{S}_\ell := \mathcal{S}_\ell - \{i\}$; $n_\ell := n_\ell - 1$;
        table_update();
        }
    if RM cell signals session initiation {
        $\text{MCR}^i := \text{MCR}$;
        /* 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 re in increasing order. */
        $\mathcal{S}_\ell := \mathcal{S}_\ell \cup \{i\}$; $n_\ell := n_\ell + 1$; $r_\ell^i := \text{CCR}$; $b_\ell^i := 0$;
        table_update();
        }
    else /* i.e. RM cell belongs to an ongoing session. */ {
        $r_\ell^i := \text{CCR}$; if $(r_\ell^i < \mu_\ell)$ then $b_\ell^i := 1$;
        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 {
    $\text{ER} := \max\{\min\{\text{ER}, \mu_\ell\}, \text{MCR}\}$;
    Forward RM(CCR, MCR, ER) towards its source;
}

table_update()
{
    rate_calculation_1: use Algorithm 8 to calculate advertised rate μ_ℓ^1;
    Unmark any marked session $i \in \mathcal{S}_\ell$ at link ℓ with $r_\ell^i \geq \mu_\ell^1$;
    rate_calculation_2: use Algorithm 8 to calculate advertised rate μ_ℓ;
    if $(\mu_\ell < \mu_\ell^1)$, then {
        Unmark any marked session $i \in \mathcal{S}_\ell$ at link ℓ with $r_\ell^i \geq \mu_\ell$;
        rate_calculation_3: use Algorithm 8 to calculate advertised rate μ_ℓ again;
        }
}

□

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

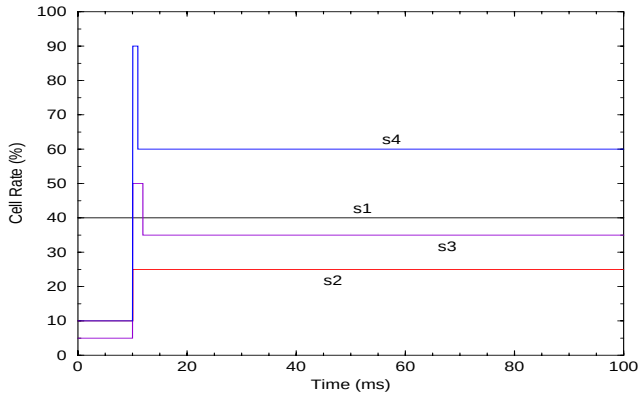

Figure 7. The cell rates of all connections in the three-node network under the distributed GMM rate calculation algorithm.

ALGORITHM 8. μ_ℓ Calculation

if $n_\ell = 0$, then $\mu_\ell := C_\ell$;
else if $n_\ell = |\mathcal{M}_\ell|$, then $\mu_\ell := C_\ell - \sum_{i \in \mathcal{S}_\ell} r_\ell^i + \max_{i \in \mathcal{S}_\ell} r_\ell^i$;
else {

$RC_\ell := C_\ell - \sum_{i \in \mathcal{M}_\ell} r_\ell^i$;
if ($RC_\ell < \sum_{i \in \mathcal{U}_\ell} \mathrm{MCR}^i$) then $\mu_\ell := 0$;
else /* i.e. $RC_\ell \geq \sum_{i \in \mathcal{U}_\ell} \mathrm{MCR}^i$. */ {
/* Due to our particular VC table creation scheme, the unmarked sessions $i \in \mathcal{U}_\ell$ are already in increasing order of their MCRs, *i.e.*, $\mathrm{MCR}[1] \leq \mathrm{MCR}[2] \leq \cdots \leq \mathrm{MCR}[|\mathcal{U}_\ell|]$. */
$k := |\mathcal{U}_\ell|$; $\mu_\ell := \frac{RC_\ell}{k}$;
while ($\mu_\ell < \mathrm{MCR}[k]$) {
$RC_\ell := RC_\ell - \mathrm{MCR}[k]$; $k := k - 1$; $\mu_\ell := \frac{RC_\ell}{k}$;
}
}‖
}

□

It can be shown that after the number of active sessions in the network stabilizes, the rate allocation for each session by our distributed flow control algorithm converges to the GMM rate allocation.[22] Furthermore, an upper bound for the convergence time to the final GMM rate allocation by our 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 less than or equal to the number of sessions in the network, N, and is often significantly less.

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 $C_\ell = 0.95 \times 150$ Mbps $= 142.5$ Mbps at every link $\ell \in \mathcal{L}$ 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 we assume that the propagation delay is 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.

‖The combined steps in the bracket for "else" are equivalent to find the GMM-bottleneck link rate μ_ℓ for the set of unmarked sessions $\mathcal{U}_\ell$ such that $\mu_\ell \cdot \sum_{i \in \mathcal{U}_\ell} 1^+\{\mathrm{MCR}^i \leq \mu_\ell\} + \sum_{i \in \mathcal{U}_\ell} \mathrm{MCR}^i \cdot 1^+\{\mathrm{MCR}^i > \mu_\ell\} = RC_\ell$. In the special case when $\mathrm{MCR}^i = 0$ for every $i \in \mathcal{U}_\ell$, $\mu_\ell = \frac{RC_\ell}{|\mathcal{U}_\ell|}$, i.e. the max-min share rate.

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

3.4. Design Tradeoff Between Performance and Complexity

The two specific flow control algorithms described in Sections 3.2 and 3.3 best represent two broad classes of distributed rate calculation algorithms in the literature. They reflect the design tradeoffs between performance objectives and implementation complexity. Table 5 summarizes important tradeoffs between these two classes of algorithms. In the following, we elaborate each item listed in Table 5 and discuss important extensions within each class of algorithms.

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.

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 our optimal rate allocation). This rate decoupling property is a consequence of our 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 has important applications in transporting rate-adaptive compressed video using a feedback control mechanism. In Ref. 25, we proposed a novel source rate adaptation algorithm, which exploits

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, can make the source's transmission rate converge smoothly to the final optimal rate allocation without going through 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 uses the buffer occupancy and load information, and a source's true transmission rate must be used in the ER calculation for the proper operation of the overall flow control algorithm.

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.

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 become 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 each flow at a switch output port and it is not possible to isolate flows and tune the system parameters for each flow. Later, 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.

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 is 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 Ref. 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 enables 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 networks. Note that such a flow aggregation technique is itself a tradeoff between performance latitude and implementation complexity.

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 has 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 replace 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]

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 for 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 the 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 our rate allocation objective.[7,14,19,32] In particular, it has been shown in Ref. 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 tune system parameters for each individual flow. In Ref. 7, it has been shown that once flows are isolated with per flow queuing, a control theoretic approach may be employed for rate calculation.

Other Extensions

Even though the specific distributed flow control algorithms that we presented use ATM ABR terminology, it should be clear that the general methodology of this work is very general and is applicable to network traffic management of any flow oriented packet switching 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 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 to support 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 Ref. 18. Unlike RCBR, the distributed flow control algorithms we discussed in this paper are capable to further explore 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 straight-forward 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 Ref. 42 that under very general requirements, 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.

4. CONCLUDING REMARKS

This paper presented an in-depth study on the network bandwidth allocation policies and their distributed flow control algorithms for packet networks. A major motivation of this work came from the intense research efforts focused on ATM ABR service for the past several years. However, the objective of this paper is not to reiterate the details of ABR standards, but rather to show some fundamental traffic management principles behind ABR, which is general enough in the broader context of flow oriented packet switching networks.

We examined the classical max-min rate allocation and presented two rate allocation policies that generalize the max-min with a minimum rate support and a peak rate constraint for each connection. We classified the many algorithms in the literature 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 show that the choice of a particular algorithm is largely a tradeoff between performance objectives and implementation complexities. Furthermore, we showed how sophisticated 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 interest. 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.

ACKNOWLEDGMENTS

This work was supported by a National Science Foundation Graduate Research Traineeship and in part by the New York State Center for Advanced Technology in Telecommunications (CATT) at Polytechnic University, Brooklyn, NY, USA.

REFERENCES

1. S. P. Abraham and A. Kumar, "Max-Min Fair Rate Control of ABR Connections with Nonzero MCRs," *Proc. IEEE GLOBECOM'97,* pp. 498–502, Nov. 1997.
2. A. Arulambalam, X. Chen, and N. Ansari, "Allocating Fair Rates for Available Bit Rate Service in ATM Networks," *IEEE Commun. Magazine,* pp. 92–100, Nov. 1996.
3. A. Arulambalam, X. Chen, and N. Ansari, "An Intelligent Explicit Rate Control Algorithm for ABR Service in ATM Networks," *Proc. IEEE ICC'97,* pp. 200–204, June 1997.
4. ATM Forum Technical Committee, "Traffic Management Specification, Version 4.0," *ATM Forum Contribution, AF-TM 96-0056.00,* April 1996.
5. D. Bertsekas and R. Gallager, *Data Networks,* Chapter 6, Prentice Hall, 1992.
6. A. W. Barnhart, "Explicit Rate Performance Evaluations," *ATM Forum Contribution, AF-TM 94-0983,* Sept. 1994.
7. L. Benmohamed and Y. T. Wang, "A Control-Theoretic ABR Explicit Rate Algorithm for ATM Switches with Per-VC Queuing," *Proc. IEEE INFOCOM'98,* pp. 183–191, March 1998.
8. G. Bianchi, L. Fratta, and L. Musumeci, "Congestion Control Algorithms for the ABR Service in ATM Networks," *Proc. IEEE GLOBECOM'96,* pp. 1080–1084, Nov. 1996.
9. F. Bonomi and K. W. Fendick, "The Rate-Based Flow Control Framework for the Available Bit Rate ATM Service," *IEEE Network,* vol. 9, no. 2, pp. 25–39, March/April 1995.
10. A. Charny, D. Clark, and R. Jain, "Congestion Control with Explicit Rate Indication," *Proc. IEEE ICC'95,* pp. 1954–1963, June 1995.
11. A. Charny, K. K. Ramakrishnan, and A. Lauck, "Time Scale Analysis and Scalability Issues for Explicit Rate Allocation in ATM Networks," *IEEE/ACM Trans. on Netwroking,* vol. 4, no. 4, pp. 569–581, August 1996.
12. T. M. Chen, S. S. Liu, and V. K. Samalam, "The Available Bit Rate Service for Data in ATM Networks," *IEEE Commun. Magazine,* pp. 56–71, May 1996.
13. F. M. Chiussi, Y. Xia, and V. P. Kumar, "Dynamic Max Rate Control Algorithm for Available Bit Rate Service in ATM Networks," *Proc. IEEE GLOBECOM'96,* pp. 2108–2117, Nov. 1996.
14. F. M. Chiussi and Y. T. Wang, "An ABR Rate-Based Congestion Control Algorithm for ATM Switches with Per-VC Queuing," *Proc. IEEE GLOBECOM'97,* pp. 771–778, Nov. 1997.
15. K. W. Fendick, "Evolution of Controls for the Available Bit Rate Service," *IEEE Commun. Magazine,* pp. 35–39, Nov. 1996.
16. E. M. Gafni, "The Integration of Routing and Flow Control for Voice and Data in a Computer Communication Network," *Ph.D. Thesis,* Dept. of Elec. Eng. and Comp. Sci., MIT, Cambridge, MA, August 1982.
17. N. Ghani and J. W. Mark, "Dynamic Rate-Based Control Algorithm for ABR Service in ATM Networks," *Proc. IEEE GLOBECOM'96,* pp. 1074–1079, Nov. 1996.
18. M. Grossglauser, S. Keshav, D. Tse, "RCBR: A Simple and Efficient Service for Multiple Time-Scale Traffic," *IEEE/ACM Trans. on Networking,* vol. 5, no. 6, pp. 741–755, Dec. 1997.
19. E. L. Hahne, "Round-Robin Scheduling for Max-Min Fairness in Data Networks," *IEEE J. Select. Areas Commun.,* vol. 9, no. 7, pp. 1024–1039, Sept. 1991.
20. H. P. Hayden, "Voice Flow Control in Integrated Packet Networks," *M.S. Thesis*, Dept. of Elec. Eng. and Comp. Sci., MIT, Cambridge, MA, June 1981.

21. Y. T. Hou, H. Tzeng, and S. S. Panwar, "A Generalized Max-Min Network Capacity Assignment Policy with a Simple ABR Implementation for an ATM LAN," *Proc. IEEE GLOBECOM'97*, pp. 503–508, Phoenix, AZ, USA, Nov. 3–8, 1997.
22. Y. T. Hou, H. Tzeng, and S. S. Panwar, "A Generalized Max-Min Rate Allocation Policy and Its Distributed Implementation Using the ABR Flow Control Mechanism," *Proc. IEEE INFOCOM'98*, pp. 1366–1375, San Francisco, CA, USA, March 29 – April 2, 1998.
23. Y. T. Hou, H. Tzeng, S. S. Panwar, and V. P. Kumar, "ATM ABR Traffic Control with a Generic Weight-Based Bandwidth Sharing Policy: Theory and a Simple Implementation," *IEICE Trans. on Commun.,* vol. E81-B, no. 5, pp. 958–972, May 1998.
24. Y. T. Hou, H. Tzeng, S. S. Panwar, "A Generic Weight Based Network Bandwidth Sharing Policy for ATM ABR Service," *Proc. IEEE ICC '98*, pp. 1492–1499, Atlanta, GA, June 7–11, 1998.
25. Y. T. Hou, S. S. Panwar, Z.-L. Zhang, H. Tzeng, and Y.-Q. Zhang, "On Network Bandwidth Sharing for Transporting Rate-Adaptive Packet Video Using Feedback," *Proc. IEEE GLOBECOM'98*, Sydney, Australia, Nov. 8–12, 1998.
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. on Commun.,* vol. COM-29, no. 7, pp. 954–962, July 1981.
28. R. Jain, S. Kalyanaraman, R. Goyal, S. Fahmy, and 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,* vol. 28, no. 13, pp. 1723–1738, October 1996.
30. L. Kalampoukas, A. Varma, and K. K. Ramakrishnan, "An Efficient Rate Allocation Algorithm for ATM Networks Providing Max-Min Fairness," *Proc. 6th IFIP Int. Conf. High Performance Networking (HPN '95),* pp. 143–154, Sept. 1995.
31. D. Lee, K. K. Ramakrishnan, W. M. Moh, and A. U. Shankar, "Performance and Correctness of the ATM ABR Rate Control Scheme," *Proc. IEEE INFOCOM'97*, pp. 785–794, Kobe, Japan, April 7–12, 1997.
32. D. Lin, "Constant-Time Dynamic ATM Bandwidth Scheduling for Guaranteed and Best Effort Services with Overbooking," *Proc. IEEE INFOCOM'97*, pp. 398–405, Kobe, Japan, April 7–12, 1997.
33. J. Mosely, "Asynchronous Distributed Flow Control Algorithms," *Ph.D. Thesis,* Dept. of Elec. Eng. and Comp. Sci., MIT, Cambridge, MA, June 1984.
34. S. Muddu, F. Chiussi, C. Tryfonas, and V. P. Kumar, "Max-Min Rate Control Algorithm for Available Bit Rate Service in ATM Networks," *Proc. IEEE ICC'96,* pp. 412–418, June 1996.
35. S. Prasad, K. Kiasaleh, and P. Balsara, "LAPLUS: An Efficient, Effective and Stable Switch Algorithm for Flow Control of the Available Bit Rate ATM Service," *Proc. IEEE INFOCOM'98,* pp. 174–182, March 1998.
36. K. K. Ramakrishnan, R. Jain, and 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. K.-Y. Siu and H.-Y. Tzeng, "Intelligent Congestion Control for ABR Service in ATM Networks," *ACM SIGCOMM Computer Commun. Review*, vol. 24, no. 5, pp. 81–106, October 1994.
39. K.-Y. Siu and H.-Y. Tzeng, "Limits of Performance in Rate-based Control Schemes," *ATM Forum Contribution, AF-TM 94-1077,* November 1994.
40. D. H. K. Tsang and W. K. F. Wong, "A New Rate-Based Switch Algorithm for ABR Traffic to Achieve Max-Min Fairness with Analytical Approximation and Delay Adjustment," *Proc. IEEE INFOCOM'96,* pp. 1174–1181, March 1994.
41. H.-Y. Tzeng and K.-Y. Siu, "Comparison of Performance Among Existing Rate Control Schemes," *ATM Forum Contribution, AF-TM 94-1078,* November 1994.
42. H.-Y. Tzeng and K.-Y. Siu, "On Max-Min Fair Congestion Control for Multicast ABR Service in ATM," *IEEE J. Select. Areas Commun.,* vol. 15, no. 3, pp. 545–556, April 1997.
43. N. Yin and M. G. Hluchyj, "On Closed-Loop Rate Control for ATM Cell Relay Networks," *Proc. IEEE INFOCOM '94*, pp.99–108, June 1994.
44. N. Yin, "Max-Min Fairness vs. MCR Guarantee on Bandwidth Allocation for ABR," *Proc. IEEE ATM'96 Workshop,* San Francisco, CA, August 25–27, 1996.