

An Efficient Scheme of General Weighted Fair Allocation for Enhanced Best-Effort Service in High Speed Networks

YEN-JEN CHEN AND SUH-YIN LEE*

Department of Computer Science and Information Engineering

National Chiao Tung University

Hsinchu, Taiwan 300, R.O.C.

E-mail: ctchen@csie.nctu.edu.tw

**E-mail: sylee@csie.nctu.edu.tw*

This research is focused on enhanced best-effort service, which provides the minimum bandwidth guarantee and distributes the available bandwidth fairly to packet flows in a high speed network. These goals can be reached when the bandwidth allocation is general weighted (GW) fair among the flows. GW fairness ensures the minimum rate for every flow and, furthermore, allocates the excess bandwidth (the available bandwidth minus the sum of the minimum rates) to the flows in a way that is proportional to their weights. In the proposed scheme, switches coordinate to efficiently achieve GW fair allocation. In the process of achieving fairness, a switch not only takes $O(\log N)$ time for bandwidth allocation among N active flows when receiving feedback from its neighbor node, but also keeps the packet queue length below a target threshold. As a result, better performance in terms of end-to-end delay, packet loss ratio, and throughput can be achieved. A proof to show that the scheme can achieve GW fair allocation is presented. Finally, simulation results illustrate and suggest a good and justifiable system design.

Keywords: general weighted fairness, weighted max-min fairness, unconstrained flow, normalized fair share, critical flow

1. INTRODUCTION

Quality of Service (QOS) is one of the main considerations in current packet networks [1, 2]. Service quality is evaluated in terms of packet delay, delay jitter, and loss ratio [3]. Services classified according to quality include Constant Bit Rate (CBR), Variable Bit Rate (VBR), Available Bit Rate (ABR), and Unspecified Bit Rate (UBR) services in the ATM network [2], Guaranteed Quality (GQ) and Controlled Load (CL) services in the integrated service model [4] of the Internet, and so on. CBR, VBR, and GQ offer a stringent guarantee of bandwidth and delay while ABR and CL offer an enhanced best-effort treatment that tries to provide users with access to the available bandwidth in order to achieve low data loss. Many schemes have been proposed for implementing these services, but more research is needed to improve performance.

Received January 14, 2000; revised May 25, 2000; accepted July 25, 2000.
Communicated by Nen-Fu Huang.

This paper will focus on enhanced best-effort service, which provides a minimum bandwidth guarantee [5] for packet flows in a network and allocates fairly the available bandwidth to them. Here, we assume that a packet flow sources from a user application and passes along a fixed network route to its destination. Though the proposed framework is based on the ATM network, the proposed scheme can be applied to other high speed networks supporting fixed-route flows, such as the MPLS (Multi-protocol Label Switching) enabled network [6]. If a network provides enhanced best-effort service, users will perceive guaranteed and fair service for time-insensitive applications, such as Web Browsing, File Transfer, Telnet, and so on. This service is also suitable for the time-sensitive applications dividing the data to be sent into several levels with different priorities. The high-priority levels of data are sent with guaranteed bandwidth while the low-priority levels of data will be sent if there is excess bandwidth for remaining levels.

In addition to the minimum bandwidth guarantee, the performance issues of concern in the implementation of enhanced best-effort service include utilization, fairness, efficiency, packet loss, delay, and throughput [7]. The available bandwidth for enhanced best-effort service should be *utilized* as much as possible and shared *fairly*. The scheme should also *efficiently* re-achieve fair allocation after some changes are made to the network conditions. The time from the network change to the re-establishment of fairness, called the *fairness response time*, is used to evaluate the *efficiency* of bandwidth allocation schemes. In addition, packet loss and delay should be as small as possible to achieve high *throughput*.

In enhanced best-effort service, a user specifies a minimum rate (MR) when he initiates his flow. To guarantee MR, the allowed rate (AR) of a flow can never be less than MR. In light of this, the excess bandwidth (the available bandwidth minus the sum of the MRs) should be fairly allocated to flows. In [8], *general weighted (GW) fairness* was proposed to allocate excess bandwidth proportionally to predefined weights of flows. GW fairness is defined based on *max-min fairness* [5, 9], which does not specify MR and weight and thus treats flows equally. To maximize network utilization, max-min fairness maximizes the minimum bandwidth among those allocated to flows, then maximizes the second minimum one, and then the next. Based on max-min fairness, *weighted max-min fairness* forces the allocation of normalized bandwidths to flows to be max-min fair, where the normalized bandwidth of a flow is its bandwidth divided by its weight. A scheme will achieve GW fairness if the allowed rate (AR) of each flow can never be less than its respective MR, and if the excess bandwidth (the available bandwidth minus the sum of the MRs) is allocated to the flows in a weighted max-min fair manner.

A number of rate-adaptive schemes [7, 8, 10-22] capable of achieving max-min fairness have been proposed. A common characteristic of these schemes is that they can work well with the simple queuing discipline, single FIFO queuing. That is, all flows of enhanced best-effort service passing through an output link are aggregated into a single FIFO queue. In contrast to single FIFO queuing, some sophisticated queuing disciplines [23-30] are associated with the rate adaptive scheme. WFQ (Weighted Fair Queuing) [23] is a per-flow based queuing discipline, in which each flow has a separate FIFO queue. The WFQ scheduler dynamically assigns the priorities of flow queues on each packet transmission in order to schedule the next flow to send packets. Clearly, a per-flow queue structure is more complex to implement and to maintain than a single

queue structure. The scalability of a per-flow queue structure is not very good when the number of flows becomes large. Thus, in this paper, we will focus on schemes based on single FIFO queuing and will not discuss those based on sophisticated queuing.

A distributed rate-adaptive scheme to achieve max-min fairness that emulates a centralized scheme was presented in [13]. With some improvement, simpler distributed schemes were proposed in [11, 15-17]. Fairness in the presence of MCR (minimum cell rate) guarantees was discussed in [18, 19]. Recently, a weighted-based max-min fairness policy and its implementation in the ABR service of the ATM network were given in [20]. A discussion of the generalized definition of max-min fairness and its distributed implementation was given in [8, 21, 22]. Some well-known schemes to achieve max-min fairness are listed as below. Congestion Avoidance using Proportional Control (CAPC) [10] and Explicit Rate Indication for Congestion Avoidance (ERICA) [7, 11] were proposed through the standardization process in ATM Forum [31]. The Max-Min scheme [12] is more efficient in achieving fairness than are CAPC and ERICA. However, some of the schemes either work well in general but in some situations do not achieve max-min fair allocation, or they achieve fair allocation but not quickly enough.

In the CAPC scheme, flows increase or decrease in rate depending on the traffic load. Oscillations may occur due to simultaneous rate increase or decrease of flows. CAPC takes a long time to achieve fair allocation, as verified via simulation in [14]. In the early version of the ERICA scheme [11], by utilizing the information of traffic loads and fair shares, flows adjust their rates accordingly, thus avoiding rate oscillations in the process of achieving max-min fairness. Therefore, they achieve a state of fair allocation more quickly. However, over avoidance of oscillations leads to failure in achieving max-min fairness when flows are initialized at some special rates [14]. The recent version of ERICA [7] allows a proper amount of oscillations if it detects that unfair allocation may result. This solves the special problem with the early version described above, but the cost is a longer fairness response time. A scheme recently proposed in [8] achieves general weighted fairness by using the ERICA scheme with performance as good as that of ERICA.

The Max-Min scheme coordinates switches in order to compute the max-min fair shares for flows; thus, oscillations are short and rare. The analysis and simulation results given in [12] show that this scheme is much quicker than ERICA in achieving max-min fair allocation. However, there are three aspects of the Max-Min scheme which need to be improved. First, a switch needs to take $O(N)$ time for bandwidth allocation, whenever feedback is received from its neighbor node, among N active flows passing through the switch. Secondly, the scheme sometimes fails to achieve fairness after a flow becomes idle. Thirdly, there is no control of the queue length. As to the third issue, a modified version of the Max-Min scheme [12] seeks to prevent the occurrence of a long queue by delaying the rate increase of local flows until the rates of remote flows decrease when there are changes in network conditions. A short delay for local flows cannot fully prevent a long queue from occurring while a long delay will affect utilization. In addition, it takes a switch $O(N)$ time to compute delays for N active flows, which is not a small computation load.

In this paper, we propose a new scheme based on the Max-Min scheme, called the Weighted Max-min Fairness with Queue Control scheme (WMFQC). WMFQC improves Max-Min in the ways mentioned above and, furthermore, achieves general

weighted fairness. The scheme consists of two mechanisms: Bandwidth Allocation (BA) and Queue Control (QC). In BA, the distributed switches allocate excess bandwidth to achieve weighted max-min fairness. This enables WMFQC to achieve GW fair allocation. The improvement over the Max-Min scheme is that 1) BA can react to the network change when a flow becomes active or idle and re-establish fair allocation, and 2) in BA, a switch takes $O(\log N)$ time for bandwidth allocation among N active flows when it receives feedback from its neighbor node.

BA does not control the packet queue lengths in the network, and this might lead to uncontrolled delay, packet loss, and throughput. Therefore, the QC mechanism is utilized to keep the queue lengths below a target threshold by using feedback to throttle the sending rates of flow sources. In QC, switches only need to monitor the queue length and set congestion flags as feedback to flow sources. Simulation results show that QC helps to achieve superior performance in delay, loss, and throughput. In addition, with good queue dimensioning, QC results in shorter fairness response time. This means that WMFQC can achieve fair allocation more quickly. When fair allocation is achieved, utilization of the available bandwidths reaches the maximum level under the principle of fairness. This implies that WMFQC achieves high utilization quickly.

The remainder of this paper is organized as follows. The Max-Min scheme is introduced in Section 2 as background. Based on the Max-Min scheme, we describe the proposed WMFQC scheme and its bandwidth allocation and queue control mechanisms in Section 3. In Section 4, we prove that WMFQC achieves general weighted fairness. In Section 5, based on simulation results, we verify the performance of WMFQC in terms of delay, loss, throughput, and fairness response time. According to these results, we recommend a good system design with a queue control mechanism. Finally, Section 6 concludes the paper.

2. BACKGROUND

This section introduces the Max-Min scheme, which our scheme is based on. In the Max-Min scheme, switches exchange information of bandwidth allocation with each other using a rate-adaptive mechanism, which is famous for congestion control in ABR service (enhanced best-effort service in ATM networks). We first introduce the rate-adaptive mechanism, then the model of switches, and finally the Max-Min scheme. The terminology associated with flow rate is introduced as follows.

- **Allowed Rate (AR)**: the rate of a flow which is allowed by the network.
- **Minimum Rate (MR)**: the minimum desired rate of a flow.
- **Peak Rate (PR)**: the maximum sending rate of a flow.
- **Supported Rate (SR)**: the rate supported for a flow by the destination node of the flow.

2.1 Rate-Adaptive Mechanism

The rate-adaptive mechanism associated with the Max-Min scheme provides the source node of each flow with a way to adjust its allowed rate (AR) to its deserved band-

bandwidth share according to feedback from the network. Network feedback from switches to the source node is carried in *control packets* containing information about the congestion status and the bandwidth allocated to the flow. *Control packets* are initiated periodically by the source node, travel and gather information along the flow path down to the destination node and then are sent back to the source node along the original path (see Fig. 1). *Forward control packets* are those flowing from the source to the destination while *backward control packets* are those returning from the destination to the source node.

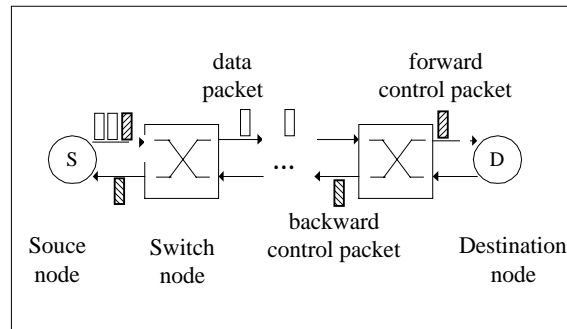


Fig. 1. The control packets of a flow.

A control packet contains two fields, Explicit Rate (ER) and Congestion Indication (CI) flag, which enable the switches to provide feedback to the source node. The ER field indicates the bandwidth that the network can offer to a flow at a particular time instant. When starting at the source node, the ER field is initially set to the peak rate (PR) of the flow, and the CI flag is clear. Via ER, the source node informs the switches of the largest bandwidth the flow desires. Along the path of a flow, each switch sets ER to the bandwidth allocated to the flow and sets the congestion flag CI if necessary. At the destination node, it updates ER by taking into consideration the supported rate (SR) of the flow. When the source node receives a backward control packet, it computes its allowed rate AR using ER and/or CI of the packet.

The above mechanism reserves flexibility in parameter setting for bandwidth allocation schemes. In the Max-Min scheme, the ER field in the control packets of a flow is set as follows. (The CI flag is not used for rate adaptation in this scheme.)

- Starting at the source node, $ER = PR$.
- At the switch node, $ER =$ the bandwidth allocated to the flow. (The model and bandwidth allocation mechanism of a switch will be presented in Sections 2.2 and 2.3.)
- At the destination node, $ER = \min \{PR, SR\}$.
- When a backward control packet is received by the source node, $AR = ER$.

2.2 Switch Model

The switch model shown in Fig. 2 shows how the received packets of different services are queued and forwarded. For each kind of service, there is a separate FIFO

queue at the output link under the control of a scheduling algorithm. The details of the scheduling algorithm are not our concern here; rather our concern is the knowledge of the bandwidth available for enhanced best-effort service, based on which the Max-Min scheme works. The queue for enhanced best-effort service, labeled EBEQ, utilizes the leftover bandwidth after the packets in the queues for other higher-priority services are transmitted.

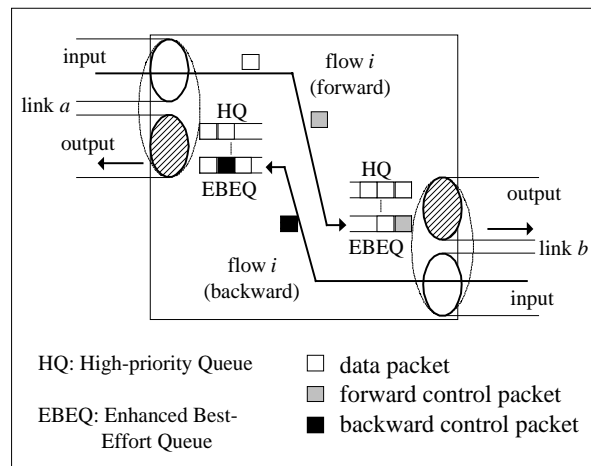


Fig. 2. Switch model.

In Fig. 2, the packets of flow i in the forward direction enter queue EBEQ of output link b . Thus, the bandwidth to be allocated to the flow belongs to output link b . The EBEQ length of the output link b also indicates whether congestion occurs or not. Thus, this queue length is checked when a backward control packet of flow i is received. If congestion occurs, the CI flag of the control packet is set as feedback indicating congestion. On the other hand, the switch monitors the load on the output link over intervals of time to determine the bandwidth available for EBEQ and whether a flow is idle or active. A flow is said to be idle if no data packet is received during the measuring interval. An idle flow will become active once a data packet is received. Bandwidth is allocated to active flows only.

2.3 Max-Min Scheme

The goal of the Max-Min scheme is to quickly achieve max-min fair allocation when the network condition changes. The basic idea is that, initially, every switch allocates equal shares to the active flows passing through its link. Thus, a flow will have different shares at the switches it passes through. The smallest of the shares is called the *bottleneck share* of the flow. The switch at which the flow has the smallest share is called the *bottleneck switch* of the flow. Other switches the flow passes through are called its *non-bottleneck switches*. Due to the bandwidth constraint at the bottleneck, the shares of the flow at its non-bottleneck switches are reduced to the amount of its bottleneck share. Thus, the flow at its non-bottleneck switch is said to be *constrained*.

However, the flow at its bottleneck switch is *unconstrained* since the bottleneck share is not constrained by the share of the flow at any switch.

For a switch, an active flow passing through it is either constrained or unconstrained. The switch reallocates the bandwidth to the constrained flows by setting their shares to be the same as their bottleneck shares, and it then distributes the leftover bandwidth evenly to the unconstrained flows. For a flow, if its share at the switch changes after reallocation, then the shares of the flow at all the switches it passes through will be compared in order to select a new bottleneck share and will then converge to the new bottleneck share. The processes of reallocation, selection, and convergence repeat until the bottleneck shares of all the flows no longer change.

To enable each share of a flow converge to the bottleneck share, each switch the flow passes through allocates to the flow a share that does not exceed the shares allocated in its upstream and downstream switches. This can be accomplished by using the bandwidth information carried in the Explicit Rate (ER) field of control packets. Every switch maintains the information ER_f , ER_b , and ER_m for each flow passing through it. ER_f and ER_b denote the ER value carried in the forward and backward control packet recently received by the switch. ER_m is the minimum of ER_f and ER_b , and serves as the *bandwidth limit* for the flow.

A switch implements the Max-Min scheme as follows. It periodically monitors the load on each output link and determines the available bandwidth ABW and the number of active flows. Initially, ER_f , ER_b , and ER_m of each flow at a link are set to be ABW of the link, and all the flows are marked “unconstrained”. When a forward (backward) control packet is received, the corresponding ER_f (ER_b) is updated to the ER value in the packet. ER_m is updated to $\min\{ER_f, ER_b\}$ accordingly. We say that the ER_m of a flow changes if the new value of ER_m is not equal to the old one. When the ER_m of a flow changes or a flow becomes active or idle, the fair share of the “unconstrained” flows, denoted by A , is recomputed as

$$A = \frac{ABW - \sum_{i: \text{"constrained" flow}} CA_i}{\text{number of "unconstrained" flows}}. \quad (1)$$

Here, CA_i denotes the bandwidth share CA of flow i . The “unconstrained” flows whose bandwidth limits ER_m s are not greater than A are now remarked as “constrained” and their shares CA s are thus reset to their ER_m s. The “constrained” flows whose ER_m s are greater than A are now remarked as “unconstrained,” and their CA s are thus reset to A . Then, Eq. (1) is computed again, and the flows are remarked following the above rule. This process is repeated until the marks of all the flows no longer change. Let A_{end} be the final value of A , which is the bandwidth share CA allocated to the “unconstrained” flows. As for “constrained” flows, their shares CA are equal to their ER_m s. It has been shown in [13, 14] that a switch takes $O(N)$ time to compute the bandwidth shares for N active flows.

In the Max-Min scheme, the ER value carried in control packets from a flow is set as follows. When starting at the source node, ER is set to the peak rate PR of the flow. At the destination node, ER is set to be the minimum value of the PR and the supported rate SR of the flow. Along the flow path, a switch sets ER to be the bandwidth share

CA allocated to the flow.

The following scenario illustrates the situation in which the Max-Min scheme fails to re-achieve max-min fair allocation after a flow becomes idle. When max-min fair allocation is achieved, the shares of a flow at the switches it passes through converge to its bottleneck share. Consider a flow i passing through at least two switches. Since the bandwidth shares of flow i at all the switches it passes through are the same, the control packets of the flow going out of the switches have the same ER. Thus, the bandwidth limit ERm of flow i at a switch is equal to the share CA allocated to the flow. As a result, if another flow j becomes idle at some switch that flow i passes through, then the bandwidth released by the idle flow j cannot be distributed to flow i . (That is, the bandwidth of flow i cannot be increased any longer though there is some released bandwidth available.) This is why the Max-Min scheme sometimes fails to achieve max-min fairness after a flow becomes idle.

3. WEIGHTED MAX-MIN FAIRNESS WITH QUEUE CONTROL SCHEME

Although the Max-Min scheme allocates the available bandwidth in a max-min fair manner, it does not have a minimum bandwidth guarantee. Based on the Max-Min scheme, the proposed scheme, WMFQC, was developed to enable the flows in enhanced best-effort service to access the available bandwidth fairly and efficiently. WMFQC guarantees minimum rates of the flows and allocates the excess bandwidth (the available bandwidth minus the sum of the minimum rates) to the flows proportionally to their weights. Thus, WMFQC achieves general weighted (GW) fairness. The details of the mechanism for bandwidth allocation (BA) is given in Section 3.1. In addition, WMFQC controls the lengths of the queues in the network by using feedback to throttle the sending rates of flows, which results in good performance in terms of delay, loss, and throughput. The mechanism for queue control (QC) is presented in Section 3.2.

3.1 Bandwidth Allocation

The basic way to achieve general weighted (GW) fair allocation is to reserve minimum rates (MRs) for all active flows first and to then allocate the excess bandwidth to the flows in a weighted max-min fair manner. Weighted max-min fairness forces the allocation of normalized bandwidths to flows to be max-min fair, where the *normalized bandwidth* of a flow is the bandwidth allocated to the flow divided by its weight. In a switch, the active flows are also divided into constrained and unconstrained flows. The switch allocates the bandwidth to the constrained flows by setting their normalized shares to their normalized bottleneck shares, and it then distributes the leftover bandwidth to the unconstrained flows proportionally to their weights.

Recall the Max-Min scheme. A switch computes the bandwidth shares for the active flows passing through it according to the bandwidth information carried in the Explicit Rate (ER) field of control packets. To achieve GW fair allocation, in addition to the information ERf , ERb , and ERm , each switch also maintains MR and Bo for every flow passing through it. MR and Bo are the minimum rate and the weight of a flow.

(ERf and ERb denote the ER values carried, respectively, in the forward and backward control packets recently received by the switch. ERm is the minimum of ERf and ERb , i.e., the bandwidth limit for a flow.) Now, allocation focuses on the excess bandwidth. For a flow, the maximum excess rate desired in its source node is equal to its peak rate minus its minimum rate (i.e., $PR - MR$), and the maximum excess rate supported in its destination node is its supported rate minus MR (i.e., $SR - MR$). Thus, when starting at the source node of a flow, ER of control packets is set to be $PR - MR$. At the destination node, ER is set to be $SR - MR$. Along the flow path, a switch sets ER by using the following bandwidth allocation mechanism.

A switch periodically monitors the load on each output link and determines the available bandwidth, the number of active flows, and the sum of the MRs of the active flows. To efficiently compute the bandwidth shares for active flows, these flows are maintained in increasing order according to the *normalized bandwidth limit*, i.e., ERm/Bo . The details of the bandwidth allocation (BA) mechanism are given in the following.

Notation:

ABW : the available bandwidth of the output link.

N : the number of active flows at the output link.

SMR : the sum of the minimum rates of the active flows.

C : the excess bandwidth to be allocated to the active flows.

$MR.i$, $Bo.i$, $ERf.i$, $ERb.i$, and $ERm.i$ are the MR , Bo , ERf , ERb , and ERm of flow i .

i_1, i_2, \dots, i_N : active flows listed in increasing order, where $ERm.i_p/Bo.i_p \leq ERm.i_q/Bo.i_q$ for $p < q$.

r : the index of critical flow i_r .

NFS : normalized fair share.

$ER(P)$: the ER field of a control packet P .

When flow i is found to become active:

<1> $ERf.i \leftarrow ERb.i \leftarrow ERm.i \leftarrow ABW$.

When a control packet P of flow i is received:

<2> If it is a forward control packet, then $ERf.i \leftarrow ER(P)$; otherwise, $ERb.i \leftarrow ER(P)$;

<3> $ERm.i \leftarrow \min \{ERb.i, ERm.i\}$;

<4> $C \leftarrow \max \{ABW - SMR, 0\}$;

<5> If C is equal to zero then the process terminates;

<6> Call the subroutine Find_NFS (NFS, r); /* if $r > N$, all flows are constrained */

<7> If ($r \leq N$ and $NFS \leq ERm.i/Bo.i$) then $ER(P) \leftarrow NFS * Bo.i$; /* flow i is unconstrained */

<8> End.

Subroutine Find_NFS ($Temp_NFS$: real number, p : integer)

<S1> $p = 1$; /* All flows are assumed unconstrained */

<S2> $Temp_NFS = \frac{C - \sum_{q=1}^{p-1} ERm.i_q}{\sum_{q=p}^N Bo.i_q}$

```

<S3> If ( $ERm.i_p / Bo.i_p < Temp\_NFS$ ) {
    /* flow  $i_p$  is said constrained */
     $p = p + 1$ ;
    if ( $p \leq N$ ) goto <S2>;
}
<S4> Return ( $Temp\_NFS, p$ );

```

When a control packet is received, the mechanism allocates the excess bandwidth C to the active flows by calling the subroutine Find_NFS. Find_NFS computes the normalized fair share NFS for the flows i_1, i_2, \dots, i_N and divides them into two groups: constrained and unconstrained. A flow is constrained if its normalized bandwidth cannot reach the normalized fair share NFS due to the constraint of its normalized bandwidth limit ERm/Bo ; otherwise, it is unconstrained. Therefore, the normalized bandwidth of a constrained flow is equal to the normalized bandwidth limit ERm/Bo of the flow, and that of an unconstrained flow is equal to the normalized fair share NFS .

To find NFS , Find_NFS assumes that all the flows i_1, i_2, \dots, i_N are initially unconstrained. Thus, their normalized bandwidth shares should be equal to the temporary normalized fair share ($Temp_NFS$), whose value is computed at step <S2> for $p = 1$. The flow i_1 is constrained if its normalized bandwidth limit ERm/Bo is smaller than the normalized bandwidth share just computed for it, i.e., the value in $Temp_NFS$; otherwise, it is unconstrained. If the flow is constrained, the share reallocated to it will be equal to its ERm . The leftover bandwidth will be reallocated to the flows which are still assumed to be unconstrained, in a way that is proportional to their weights. Thus, the flows i_2, i_3, \dots are checked one by one to see if they are constrained or unconstrained using step <S3> until a flow, say i_r , is found to not be constrained. If flow i_r exists (i.e. $r \leq N$), then the flows i_r, i_{r+1}, \dots, i_N are said to be unconstrained. Flow i_r is critical; thus, it is called the *critical flow*. The final value of $Temp_NFS$ is the value of NFS .

In step <7>, the ER field of the control packet received from the flow i is updated to its bandwidth share $NFS * Bo.i$ if flow i is unconstrained; otherwise, ER is not changed. A flow is unconstrained if its normalized bandwidth limit is so large that its normalized bandwidth share can achieve the normalized fair share NFS . This means that the share is not constrained by the shares of this flow at other switches. It is clear that the share is the bottleneck share of this flow. That is, the share of an unconstrained flow is the bottleneck share along the flow path. Since all the shares of the flow are to converge to the bottleneck share, the switches along the flow path only need the information of the bottleneck share to perform bandwidth allocation. Therefore, ER must be updated only at the switch where the flow is unconstrained. The mechanism uses the Boolean expression " $r \leq N$ and $NFS \leq ERm.i/Bo.i$ " in step <7> to check if the flow i is unconstrained at the switch. The term $r \leq N$ means that the critical flow i_r exists. Thus, there is at least one unconstrained flow. (Recall step <S3> and <S4>. If $r > N$, then all the flows are constrained.) The term $NFS \leq ERm.i/Bo.i$ means that flow i is unconstrained. The reason is presented in the following theorem.

Theorem 1. For the ordered flow list i_1, i_2, \dots, i_N , if the critical flow i_r exists (i.e., i_1, i_2, \dots, i_{r-1} are constrained, and i_r, i_{r+1}, \dots, i_N are unconstrained), then $ERm.i_1/Bo.i_1 \leq \dots$

$\leq ERm.i_{r-1}/Bo.i_{r-1} < NFS \leq ERm.i_r/Bo.i_r \leq \dots \leq ERm.i_N/Bo.i_N$. (See the proof in the Appendix.)

The rule that ER must be updated only at a switch where a flow is unconstrained enables new fair allocation to be achieved after some flow becomes idle. (Recall that the Max-Min scheme sometimes fails to achieve fair allocation after a flow becomes idle.) For a flow i , if in steady state, there is only one switch at which the flow is unconstrained, then the bandwidth limit of the flow at the switch is $\min\{PR - MR, SR - MR\}$ since ER is set to be $PR - MR$ and $SR - MR$ at the source and destination nodes, respectively. (PR , SR , and MR are the Peak, Supported, and Minimum Rate, respectively.) Let this switch be switch S . In fact, the bandwidth share of flow i at switch S is the bottleneck share. Since the bottleneck share information is conveyed to other switches that flow i passes through, the bandwidth limits of the flow at these switches are equal to the bottleneck share. When another flow j passing through switch S becomes idle, the bottleneck share of flow i can be increased unless the share has reached its bandwidth limit, $\min\{PR - MR, SR - MR\}$, at switch S . The information of the new bottleneck share will be conveyed from switch S to other switches; thus, the bandwidth limits at the switches except for S change to the new bottleneck share. This scenario shows that the released bandwidth of an idle flow is distributed to an active flow. We will give a proof in Section 4 showing that the WMFQC scheme achieves fair allocation in steady state.

It takes $O(N)$ time for Find_NFS to find critical flow i_r and NFS while for modified Find_NFS, it takes only $O(\log N)$ time based on the following theorem.

Theorem 2. Let i_p be a flow among i_1, i_2, \dots, i_N , and let $\theta.i_p$ be the normalized share of i_p supposing i_1, \dots, i_{p-1} are constrained and i_p, \dots, i_N are unconstrained, where

$$\theta.i_p = \frac{C - \sum_{q=1}^{p-1} ERm.i_q}{\sum_{q=p}^N Bo.i_q}. \quad (2)$$

In fact, flow i_p is constrained if and only if $ERm.i_p/Bo.i_p < \theta.i_p$. (See the proof in the Appendix.)

According to Theorem 2, the critical flow i_r can be found, if i_r exists, by means of binary search of the flow list i_1, i_2, \dots, i_N . The search is as follows. If $ERm.i_p/Bo.i_p < \theta.i_p$, then i_p is constrained. Thus, i_r is either among $i_{p+1}, i_{p+2}, \dots, i_N$ or does not exist. The search goes through $i_{p+1}, i_{p+2}, \dots, i_N$. On the other hand, if $ERm.i_p/Bo.i_p \geq \theta.i_p$, then i_p is unconstrained. Thus, i_r may be i_p or among i_1, \dots, i_{p-1} . Flow i_p is known as the most possible critical flow (MPCF) till now. Then, the search goes through i_1, \dots, i_{p-1} . If no new MPCF is found, then flow i_p is i_r and $\theta.i_p$ is NFS . Furthermore, the list i_1, i_2, \dots, i_N must be re-sorted whenever the bandwidth limit of a flow changes. Re-sorting of the flows involving one flow insertion and one deletion takes $O(N)$ time. To reduce the complexity of insertion and deletion, we replace the linear list structure for flows i_1, i_2, \dots, i_N with a balanced binary tree structure, such as an AVL tree or Red Black tree [32]. Therefore, it takes $O(\log N)$ time to insert and delete a flow. As for the binary tree search for the critical flow i_r based on Theorem 2, the most complex step is the computation of $\theta.i_p$, which can be done in $O(1)$ time by using two variables to maintain the val-

ues of $\sum_{q=1,p-1} ERM.i_q$ and $\sum_{q=p..N} Bo.i_q$ (more detail in [33]). It, thus, takes $O(\log N)$ time to search for i_r . It is beneficial to develop an $O(\log N)$ algorithm for high speed networks, where the number of flows is large.

We will now summarize the differences between the Max-Min scheme and the bandwidth allocation mechanism of WMFQC. **First**, the Max-Min scheme allocates the available bandwidth to active flows in a max-min fair manner while WMFQC reserves minimum bandwidth and allocates the excess bandwidth in the weighted max-min fair manner. **Secondly**, in Max-Min, ER is set to be the minimum of the peak rate (PR) and supported rate (SR) at destination nodes while in WMFQC, ER is set to be $SR - MR$. In WMFQC, each destination node does not need to store the information of PR, but does need to store the information of MR for the minimum rate guarantee. **Thirdly**, in Max-Min, a switch sets ER to be the allocated share of a flow while in WMFQC, a switch changes ER to be the allocated share of a flow only when the flow is unconstrained. This is the reason why WMFQC re-establishes fair allocation even when a flow becomes idle. **Finally**, in the Max-Min scheme, a switch, when receiving a control packet, takes $O(N)$ time to allocate bandwidth to N active flows while in WMFQC, a switch takes $O(\log N)$ time.

3.2 Queue Control

The queue length deeply affects the network performance in terms of end-to-end delay, packet loss ratio, and network throughput. The fairness response time is also affected since control packets carrying bandwidth information are delayed in the queue. The queue control (QC) mechanism controls the queue by using congestion avoidance. The congestion information is detected by checking the queue length and is carried back to the source nodes by using congestion indication (CI) flags of control packets. Upon receiving a backward control packet, if CI is set, the source node of a flow will reduce its allowed rate (AR) to the minimum rate (MR) in order to relieve congestion; otherwise, it will set AR to the explicit rate (ER) value in the control packet plus MR since ER carries the information of excess bandwidth allocated to the flow. This is summarized as follows:

$$AR = \begin{cases} MR, & \text{if CI}=1; \\ MR + ER, & \text{otherwise.} \end{cases} \quad (3)$$

For each queue, there are two thresholds, target and dropping thresholds, for controlling the queue length. The thresholds are denoted by Q_T and Q_D , and are used for congestion detection and packet dropping. Whenever a switch receives a backward control packet from a flow, it checks the queue of the output link the flow passes through. If the queue length exceeds Q_T , then the CI flag in the control packet is set as feedback for congestion. On the other hand, whenever a data packet arrives, the switch checks if the queue length exceeds Q_D . If it does, the data packet is dropped. Notably, control packets are not dropped unless the link buffer overflows. This is to prevent control packets from being lost during congestion.

To guarantee that the queue can always be restored below the threshold Q_T , a guard

bandwidth (GB) is reserved, which cannot be accessed by any high priority service queue (HQ), in order to drain the queue when congestion occurs. Let GF be the guard-band factor, which is a fraction between 0 to 1. The link capacity should be scheduled as follows:

$$\begin{aligned} \text{link capacity} &= \text{bandwidth reserved for EBEQ} + \text{bandwidth reserved for all HQs;} \\ \text{bandwidth reserved for EBEQ} &= \text{sum of the MRs of the flows through EBEQ} + \text{GB;} \\ \text{GB} &= \text{GF} * \text{sum of the MRs of the flows through EBEQ.} \end{aligned}$$

Since the bandwidth reserved for HQs sometimes is not used, the available bandwidth ABW for EBEQ is at least the sum of GB and the MRs. When the source nodes reduce AR to MR to relieve congestion at EBEQ, there will be a capacity of at least GB to drain the queue. GB should be proportional to the sum of the MRs. The factor GF should be about 5% based on our experience using the scheme ERICA [11], which will be explained next.

To avoid a long queue, ERICA sets the target utilization of ABW to be about 95%. Thus, in steady state, the target available bandwidth is $0.95 * ABW$. The queue tends to be empty in steady state since about 5% of ABW is used to drain the queue off. Five percent is thought to be enough. However, this five percent cannot be used by the flows passing through EBEQ. Of course, it cannot be used by the flows passing through HQs, either. In comparison with GB in steady state, in ERICA, the 5% of ABW is wasted while in WMFQC, GB can be completely used since it is allowed to access hundred percent of ABW .

The recent version [7] of ERICA includes a queue control mechanism, which modified the target utilization factor as a function of the queue length. Because the target utilization factor converges to one in steady state, it is allowed to access hundred percent of ABW in steady state. Recently, ERICA was employed to solve the GW fair allocation problem in [8]. It is believed that it is also necessary for ERICA to reserve a GB for the flows. If no GB is reserved, it is possible that the capacity of ABW will be saturated by the flows with AR being equal to MR. In this case, the queue has no chance to become shorter since no AR can be smaller than MR unless the network condition changes. For example, ABW becomes larger or some flow becomes idle. However, waiting for a change in the network condition is too passive to enable congestion to be relieved. Therefore, both ERICA and WMFQC need GB to strengthen their reliability to relieve congestion. However, the queue control mechanism of WMFQC is simpler than that of ERICA since ERICA needs to compute the target utilization factor according to the queue length at the end of each measurement interval while WMFQC needs to read the queue length only.

4. CORRECTNESS IN FAIRNESS

The allowed rates (ARs) of all active flows reflect the state of the current bandwidth allocation. We will show that bandwidth allocation using the WMFQC scheme is general weighted (GW) fair when the network is in steady state. In this scheme, a switch allocates to flows the excess bandwidth of the output link, i.e., the excess of the available

bandwidth over the sum of the minimum rates (MRs). Thus, a flow receives respective bandwidth shares at the output links it passes through.

Definition 1. (Steady State) A network employing the WMFQC scheme is in steady state if 1) the shares of every flow at the output links it passes through converge to its bottleneck share and 2) the queue lengths of the links are below the target threshold Q_T .

The second entry means that in steady state, there is no congestion; thus, the flow sources can send packets at the explicit rate (ER) recommended by the network, instead of at the minimum rate in order to relieve congestion.

The bandwidth allocation for all the active flows in the network is represented by the vector of ARs, denoted by $\mathbf{AR} = (AR.1, AR.2, \dots, AR.K)$, where 1, 2, ..., and K are the numbers for the active flows. The vectors of minimum rates (MRs) and weights are denoted by $\mathbf{MR} = (MR.1, MR.2, \dots, MR.K)$ and $\mathbf{Bo} = (Bo.1, Bo.2, \dots, Bo.K)$, respectively. With \mathbf{AR} , the normalized vector of excess bandwidths for the flows, denoted by $\lambda = (\lambda.1, \lambda.2, \dots, \lambda.K)$, is defined as follows:

$$\lambda.i = \frac{AR.i - MR.i}{Bo.i}, i = 1, 2, \dots, K. \quad (4)$$

Definition 2. (Feasibility) A normalized vector of excess bandwidths λ is *feasible* if it satisfies the following constraints: 1) the normalized bandwidth $\lambda.i \geq 0$ for any active flow i and 2) $F(l) \leq C(l)$ for any output link l , where $F(l) = \sum_{i \in A(l)} \lambda.i * Bo.i$ and $C(l) = ABW(l) - \sum_{i \in A(l)} MR.i$. Here, $A(l)$ is the set of the active flows passing through link l , and $ABW(l)$ is the available bandwidth at link l . $F(l)$ is the sum of the excess bandwidths for the flows in set $A(l)$, and $C(l)$ is the total excess bandwidth at the link.

Definition 3. (Weighted Max-min Fairness) In steady state, a vector λ is weighted max-min fair if it is feasible, and if for each active flow i , to maintain *feasibility*, the normalized bandwidth $\lambda.i$ cannot be increased without decreasing $\lambda.j$, where $\lambda.j \leq \lambda.i$ for some active flow j .

Definition 4. (GW Fairness) In steady state, the bandwidth allocation vector \mathbf{AR} is general weighted fair if its associated vector λ is weighted max-min fair.

Lemma 1. According to the bandwidth allocation mechanism of WMFQC, in steady state, if a flow is constrained at every output link it passes through, then the AR is equal to the minimum of the PR (peak rate) and SR (supported rate at the destination node).

Proof: If the flow is constrained at every output link it passes through, the ER field of the control packets, initialized to be PR minus MR, is not updated by the switch from step <7> of the mechanism, as shown in Section 3.1. The ER field is set to be SR minus MR when the control packet is at the destination node and the value is still kept when the control packet goes back to the source node. In steady state, there is no congestion. From Eq. (3), $AR = MR + ER = \min \{PR, SR\}$.

Theorem 3. The normalized vector of excess bandwidths λ introduced by WMFQC is weighted max-min fair in steady state.

Proof: Assume there are K active flows passing through a network which employs WMFQC. When the network stays in steady state, we have $AR = MR + ER$ from Eq. (3). For a flow i , $\lambda_i = (AR_i - MR_i)/Bo_i$ from Eq. (4). In steady state, the normalized shares of the flow at the output links it passes through converge to a common value, λ_i . Let λ be the normalized bandwidth vector of all the active flows; that is, $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_K)$. According to the definition of weighted max-min fairness (Definition 3), we examine each λ_i as follows. From Lemma 1, if active flow i is constrained at every link it passes through, then $AR_i = \min \{PR_i, SR_i\}$. That is, AR is either PR or SR . Clearly, AR cannot be increased; thus, λ_i cannot be increased. On the other hand, assume that flow i is unconstrained at some output link l . The normalized share of the unconstrained flow is equal to the *NFS* (normalized fair share) at link l , which is larger than the normalized share of any constrained flow passing through the same link from Theorem 1. Thus, $\lambda_i \geq \lambda_j$ for any flow j , $j \neq i$, passing through link l . If we try to increase λ_i , we must decrease the normalized bandwidth λ_j of some flow j in order to maintain the feasibility of the network. Thus, λ_i cannot be increased without decreasing λ_j of some flow j with $\lambda_j \leq \lambda_i$. Therefore, from Definition 3, the vector λ is weighted max-min fair. Proven.

According to Definition 4, in steady state the allocation AR achieved by WMFQC is GW fair since its associated vector λ is weighted max-min fair.

5. SIMULATION

We first show the efficiency of WMFQC in Section 5.1, which is compared with the efficiency of the Max-Min scheme. Then, we show the performance in terms of end-to-end delay, data loss, and throughput in Section 5.2. We suggest possible parameters for queue control in the design of the switch nodes to achieve better performance.

5.1 Efficiency of the WMFQC Scheme

To measure the efficiency of WMFQC is to measure the fairness response time. Since the Max-Min scheme has better performance than the CAPC and ERICA schemes in terms of fairness response time (recall our discussion in Section 1), we will compare WMFQC with the Max-Min scheme. For any switch in a large-scale network, WMFQC use less computation time in bandwidth allocation as mentioned in Section 3.1. Here, we assume that the computation time is zero at each switch. Thus, the only factor that affects the fairness response time is the number of runs required to achieve fair allocation. A run is defined as the propagation of bandwidth information (i.e., the ER value carried by a control packet) between two switches. The time needed for a run is the link propagation delay plus the queuing delay. To compare WMFQC with the Max-Min scheme, we set the minimum rate and weight of each flow to be zero and one, respec-

tively, and disable the queue control mechanism in WMFQC since in the Max-Min scheme, there are no support for minimum bandwidth guarantee, excess bandwidth allocation proportional to weight, or queue length control. In this experiment, we assumed that the queue length was unlimited in both schemes; thus, no packet would be dropped.

Since the Max-Min scheme was developed for ATM networks, the simulation model adopted was the ATM network shown in Fig. 3. There were six flows VC1 through VC6 passing through the network for enhanced best-effort service. The available bandwidth at a link was thus the link capacity in this environment. The unit of the link capacity was cell/ms. For example, the capacity of L1 was 117.92 cell/ms, i.e., 50 Mbps. We assumed that the length of each link was 50 km; thus, the propagation delay at each link was 250 μ s. The respective sources of the flows, one by one, started sending cells (i.e., turned on) and then stopped (i.e., turned off) sending them. The (on, off) time pairs of VC1 through VC6 were (0, 200), (10, 250), (20, 300), (30, 350), (40, 400), and (50, 450) in milliseconds (ms). The PR (peak rate) and SR (supported rate) of each flow are shown in Table 1. The IR (initial rate) of each flow was set to be PR.

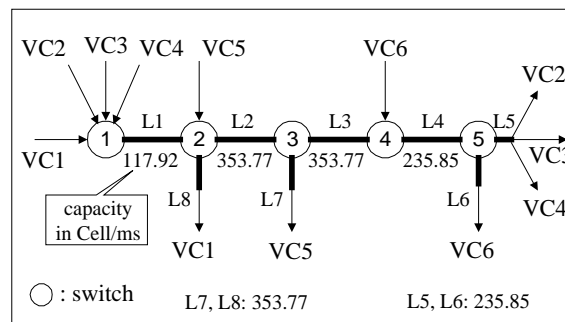


Fig. 3. Simulation model (an ATM network).

Table 1. The PR and SR of flows in cell/ms.

	VC1	VC2	VC3	VC4	VC5	VC6
PR	60	120	120	120	360	250
SR	150	60	120	60	500	300

Tables 2 and 3 list the response time required for each flow i , i.e., VC i , to achieve its individual max-min fair allocation AR_i in the Max-Min and WMFQC schemes, respectively, when some flows turned on or off. For example, in WMFQC, when VC2 turns on, VC1 achieves a new fair allocation after 539.28 μ s (Table 3). When VC1 turns off, VC2 achieves a new one after 968.35 μ s. When a flow turns off, the response times of all the flows in the Max-Min scheme are zero because their ARs do not change. This is because the Max-Min scheme cannot react to the change after a flow becomes idle. However, WMFQC can react and reflect the up-to-date network situation. The tables also show that the respective response times of a flow in the Max-Min and WMFQC schemes are the same when some flow turns on in order to send packets. This

is evidence that Max-Min and WMFQC require the same number of runs to achieve the max-min fairness allocation. When the computational time of bandwidth allocation is added in, WMFQC is more efficient than Max-Min. This is because a switch in WMFQC uses $O(\log N)$ time for bandwidth allocation while a switch in Max-Min uses $O(N)$ time whenever receiving a control packet.

Table 2. The fairness response time of each flow in μs at different times for the Max-Min scheme.

Max-Min	AR.1	AR.2	AR.3	AR.4	AR.5	AR.6
VC1(on)	0.00	N/A	N/A	N/A	N/A	N/A
VC2(on)	539.28	3062.19	N/A	N/A	N/A	N/A
VC3(on)	757.37	643.34	4662.88	N/A	N/A	N/A
VC4(on)	891.02	938.11	793.95	8086.78	N/A	N/A
VC5(on)	0.00	0.00	0.00	0.00	1514.13	N/A
VC6(on)	0.00	0.00	0.00	0.00	0.00	1525.44
VC1(off)	N/A	0.00	0.00	0.00	0.00	0.00
VC2(off)	N/A	N/A	0.00	0.00	0.00	0.00
VC3(off)	N/A	N/A	N/A	0.00	0.00	0.00
VC4(off)	N/A	N/A	N/A	N/A	0.00	0.00
VC5(off)	N/A	N/A	N/A	N/A	N/A	0.00
VC6(off)	N/A	N/A	N/A	N/A	N/A	N/A

Table 3. The fairness response time of each flow in μs at different times for the WMFQC scheme.

WMFQC	AR.1	AR.2	AR.3	AR.4	AR.5	AR.6
VC1(on)	0.00	N/A	N/A	N/A	N/A	N/A
VC2(on)	539.28	3062.19	N/A	N/A	N/A	N/A
VC3(on)	757.37	643.34	4662.88	N/A	N/A	N/A
VC4(on)	891.02	938.11	793.95	8086.78	N/A	N/A
VC5(on)	0.00	0.00	0.00	0.00	1514.13	N/A
VC6(on)	0.00	0.00	0.00	0.00	0.00	1525.44
VC1(off)	N/A	968.35	770.80	1288.08	12692.16	13986.01
VC2(off)	N/A	N/A	1205.82	959.90	12484.11	13912.23
VC3(off)	N/A	N/A	N/A	758.93	11945.34	13439.89
VC4(off)	N/A	N/A	N/A	N/A	781.08	3982.96
VC5(off)	N/A	N/A	N/A	N/A	N/A	0.00
VC6(off)	N/A	N/A	N/A	N/A	N/A	N/A

5.2 Performance of the WMFQC Scheme

The following experiment was designed to show the performance of the WMFQC scheme in terms of end-to-end delay, loss ratio, and throughput. Based on the experimental results, we suggest use of the parameters Q_D (dropping threshold) and Q_T (target threshold) for queue control in the switch design in order to achieve better performance.

To find a better (Q_D, Q_T) pair that would result in better performance in terms of end-to-end delay, throughput, and loss ratio, we first fixed Q_D and then varied the ratio of Q_T to Q_D from 93% to 40%. Finally, two kinds of tests were compared in terms of the fairness response time, one with better (Q_D, Q_T) and the other with queue control disabled. The purpose was to verify that the mechanism of queue control was really helpful to the performance in terms of the fairness response time.

The simulation for WMFQC was based on the same model shown in Fig. 3. However, the weights of the flows were no longer the same, and the link lengths were parameterized for LAN and WAN for the sake of completeness. For simplicity but without loss of generality, the weight was set to be the minimum rate (MR) for each flow as listed in Table 4. The peak rate (PR) and supported rate (SR) are also listed. The length of each link was set to be 2 km for LAN and 50 km for WAN; thus, the propagation delay per link, PD , was 10 and 250 μ s, respectively, as listed in Table 5.

Table 4. The MR, PR, and SR of flows in cell/ms.

	VC1	VC2	VC3	VC4	VC5	VC6
MR	10	20	30	40	90	60
PR	60	120	120	120	360	250
SR	150	60	120	60	500	300

Table 5. Parameters of queue control. (Q_D, Q_T in cells)

	PD	Q_D	Q_T				
LAN	10 μ s	75 100%	70 93%	60 80%	50 67%	40 53%	30 40%
WAN	250 μ s	500 100%	450 90%	400 80%	350 70%	300 60%	250 50%

As listed in Table 5, the dropping threshold Q_D at each link was fixed at 75 and 500 cells for LAN and WAN, respectively. The target threshold Q_T varied from 70 to 30 cells for LAN (i.e., Q_{TD} was from 93% to 40%, where Q_{TD} denotes the ratio of Q_T to Q_D) and varied from 450 to 250 cells (from 90% to 50%) for WAN. Each test for WMFQC was labeled according to its environment (LAN or WAN) and the ratio of Q_T to Q_D . For example, the LAN test using Q_T of 70 was labeled LQC93%; the WAN test with queue control disabled was labeled WQCX.

The maximum queue lengths at links L1 through L5 for LAN and WAN are shown in Figs. 4 and 5. It is reasonable that the tests, LQCX and WQCX, have the longest maximum queue lengths in LAN and WAN, respectively, because their queue control was disabled. In other tests (where queue control was enabled), the smaller Q_{TD} (the ratio of Q_T to Q_D) was, the shorter was the maximum queue length in general. The results for the average end-to-end delays of VC1 through VC6 for LAN and WAN are

shown in Figs. 6 and 7. The tests with enabled queue control also produced better performance than did those with disabled queue control. In the tests, the smaller Q_{TD} was, the shorter was the average end-to-end delay.

The throughput is the total number of cells successfully received by the destination nodes during the life times of the flows. The loss ratio is the number of lost cells over the sum of the lost and successful cells. The number of successful cells and the loss ratio are listed in Tables 6 and 7 in order to illustrate the throughput and cell loss. There was no cell loss in LQCX or WQCX since the queue length was assumed to be unlimited in the tests with queue control disabled. However, the number of successful cells in LQCX is not the highest among the LAN tests (Table 6) since many cells were queued in switches when the life times of the flows expired. A similar situation also occurred in WQCX in the WAN tests (Table 7).

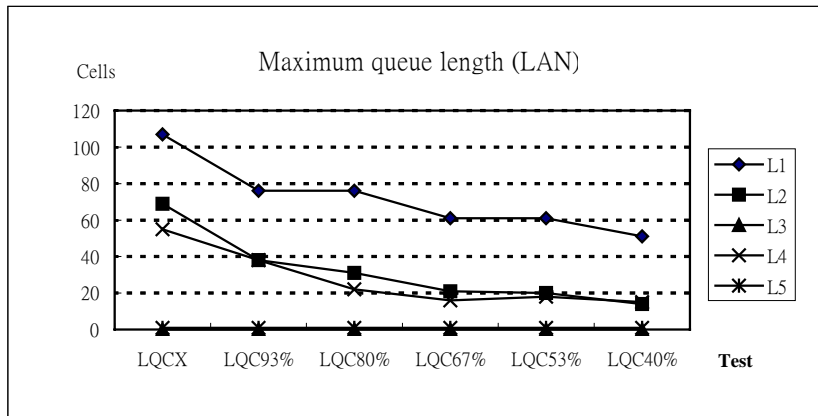


Fig. 4. The maximum link queue length in LAN.

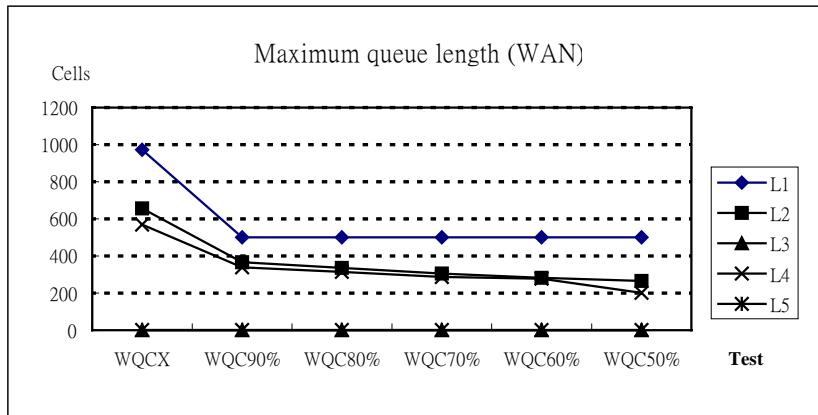


Fig. 5. The maximum link queue length in WAN.

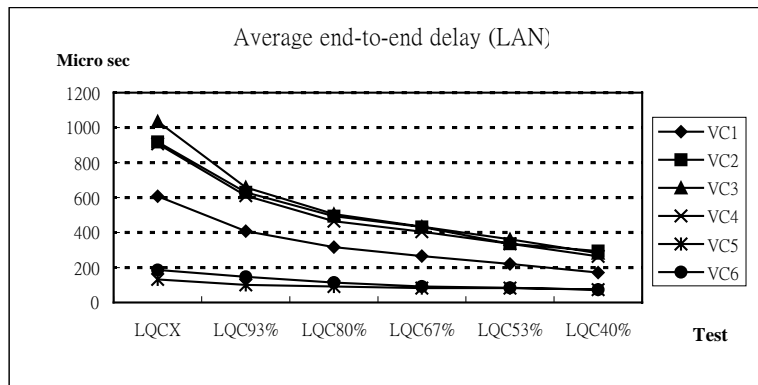


Fig. 6. The average end-to-end delay in LAN.

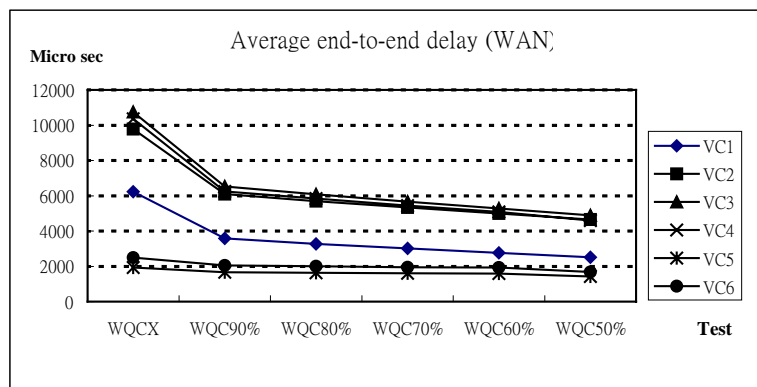


Fig. 7. The average end-to-end delay in WAN.

Table 6. The throughput and loss ratio in LAN.

LAN	Succ. cells	Lost cells	Loss ratio
LQCX	190409	0	0
LQC93%	190411	44	2.3×10^{-4}
LQC80%	190418	41	2.2×10^{-4}
LQC67%	190422	0	0
LQC53%	190418	0	0
LQC40%	190410	0	0

Table 7. The throughput and loss ratio in WAN.

WAN	Succ. cells	Lost cells	Loss ratio
WQCX	189904	0	0
WQC90%	190036	402	2.1×10^{-3}
WQC80%	190055	236	1.2×10^{-3}
WQC70%	190072	171	9.0×10^{-4}
WQC60%	190091	110	5.8×10^{-4}
WQC50%	189979	25	1.3×10^{-4}

The throughput (the number of successful cells) is mainly affected by the target threshold of the queue length, Q_T , and by the fairness response time. If Q_T is too short, before the flows achieve fair allocation, queues may be drained off; thus, there is a period of time during which there is no cell to transmit. This leads to bad throughput. On the other hand, the shorter the fairness response time, the larger the throughput. The reason is as follows. If a state of fair allocation has not been achieved, then the traffic injected according to the allowed rates of flows either overloads or under-loads the network. Let *normal throughput* be the throughput of the network in the state in which fair allocation is achieved. When the network is overloaded, the throughput is not necessarily larger than the normal throughput since many cells may be queued in switches. However, when the network is under-loaded, the throughput must be smaller than the normal throughput. Thus, to achieve large throughput, the flows need to achieve fair allocation rapidly; that is, it is desirable for the fairness response time to be as short as possible.

Based on the above analysis, consider the numbers of successful cells in the tests listed in Tables 6 and 7. The number of successful cells did not vary significantly in LAN because the flows in LAN could achieve fair allocation quickly. In WAN, however, the number of successful cells increased with decreasing Q_{TD} (the ratio of Q_T to Q_D) except for the case of WQC50%. A small Q_{TD} resulted in a short queue length and, thus, a short fairness response time. This led to high throughput. However, a ratio that was too small means a very short queue length; thus, the queue may be drained off before fair allocation is achieved. In this situation, the loss was greater than the gain in throughput. This is the reason why the number of successful cells in WQC50% was less than that in WQC60%. As for the loss ratio, it decreased with decreasing Q_{TD} . The reason is obvious.

It is clear that the maximum queuing delay is equal to the dropping threshold Q_D divided by the bandwidth reserved for the queue. Q_D can be determined accordingly. We suggest that the ratio Q_{TD} should be small based on the results obtained for the maximum queue length, average end-to-end delay, and loss ratio. However, Q_{TD} cannot be infinitely small based on the throughput results. From Tables 6 and 7, the best performance results seem to be those for LQC67% in LAN and WQC60% in WAN. Thus, an acceptable Q_{TD} is about 60%, taking into consideration the various aspects of performance.

The dynamics of the allowed rates (ARs) of the flows in LQC67% and WQC60% are shown in Figs. 8 and 9, respectively. The figs. show that the WMFQC scheme could re-establish fair allocation by adapting to network change when flows became active on or inactive off, thus overcoming the problem with the Max-Min scheme. The experimental results also show that the queue control mechanism of WMFQC does not make the flow rates oscillate. These properties indicate that WMFQC performs well in terms of fairness and stability.

Finally, we will present the difference in the fairness response time for mechanisms with “enabled” and “disabled” queue control. For LAN, we compare LQC67% and LQCX. For WAN, we compare WQC60% and WQCX. LQC67% and WQC60% with the best performances in terms of delay, loss, and throughput, were selected to test the effectiveness of the queue control mechanism in terms of fairness response time. The response times of all the flow needed to achieve individual fair allocation in these tests are listed in Tables 8 through 11. For each event, when a flow turned on or off, the

Table 9. The individual response times in μs for LQC67% in LAN ($Q_T = 50, Q_D = 75$).

LQC67%	AR.1	AR.2	AR.3	AR.4	AR.5	AR.6	max
VC1(on)	0.00	N/A	N/A	N/A	N/A	N/A	0.00
VC2(on)	303.09	182.19	N/A	N/A	N/A	N/A	303.09
VC3(on)	1182.98	1333.44	1401.28	N/A	N/A	N/A	1401.28
VC4(on)	1698.23	2705.17	2145.49	1755.40	N/A	N/A	2705.17
VC5(on)	0.00	0.00	0.00	0.00	74.13	N/A	74.13
VC6(on)	0.00	0.00	0.00	0.00	0.00	85.44	85.44
VC1(off)	N/A	1256.57	671.45	255.93	1067.64	1170.04	1256.57
VC2(off)	N/A	N/A	809.67	627.35	658.89	668.02	809.67
VC3(off)	N/A	N/A	N/A	0	264.28	428.88	428.88
VC4(off)	N/A	N/A	N/A	N/A	129.73	210.95	210.95
VC5(off)	N/A	N/A	N/A	N/A	N/A	0.00	0.00
VC6(off)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Average							808.35

Table 10. The individual responses time in μs of each VC for WQCX in WAN.

WQCX	AR.1	AR.2	AR.3	AR.4	AR.5	AR.6	max
VC1(on)	0.00	N/A	N/A	N/A	N/A	N/A	0.00
VC2(on)	3328.86	3062.19	N/A	N/A	N/A	N/A	3328.86
VC3(on)	799.78	626.38	4196.48	N/A	N/A	N/A	4196.48
VC4(on)	1908.62	887.23	632.83	6746.93	N/A	N/A	6746.93
VC5(on)	0.00	0.00	0.00	0.00	1514.93	N/A	1514.93
VC6(on)	0.00	0.00	0.00	0.00	0.00	1523.88	1523.88
VC1(off)	N/A	1460.36	620.84	752.28	9281.16	10931.64	10931.64
VC2(off)	N/A	N/A	712.42	653.06	9313.38	10374.50	10374.50
VC3(off)	N/A	N/A	N/A	0	8915.94	9957.28	9957.28
VC4(off)	N/A	N/A	N/A	N/A	827.11	3256.35	3256.35
VC5(off)	N/A	N/A	N/A	N/A	N/A	0.00	0.00
VC6(off)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Average							5758.98

maximum of the response times of all the flows, also called the fairness response time of the scheme at that event, is listed in the columns marked “max” in these tables. The average of fairness response times at all events is listed in the lower-right corner. Notably, the entries containing zeros or “N/A” in the “max” column were not used to obtain the average since a zero entry means that the allowed rate of the flow is not affected by the corresponding event while an “N/A” entry means the flow is not active. From the average values listed in the four tables (8~11), we observe that LQC67% and WQC60% achieved 16% and 34% improvement over LQCX and WQCX in terms of the fairness response time, respectively. This indicates that the queue control mechanism does achieve good performance in terms of efficiency, delay, loss, and throughput.

Table 11. The individual response times in μs for WQC60% in WAN ($Q_T = 300, Q_D = 500$).

WQC60%	AR.1	AR.2	AR.3	AR.4	AR.5	AR.6	max
VC1(on)	0.00	N/A	N/A	N/A	N/A	N/A	0.00
VC2(on)	3430.62	3062.19	N/A	N/A	N/A	N/A	3430.62
VC3(on)	6583.16	6884.65	6452.17	N/A	N/A	N/A	6884.65
VC4(on)	1891.66	1396.03	1099.23	5186.61	N/A	N/A	5186.61
VC5(on)	0.00	0.00	0.00	0.00	1514.93	N/A	1514.93
VC6(on)	0.00	0.00	0.00	0.00	0.00	1525.44	1525.44
VC1(off)	N/A	1741.76	1449.20	1071.84	3910.47	4857.25	4857.25
VC2(off)	N/A	N/A	853.90	658.86	3334.95	4529.07	4529.07
VC3(off)	N/A	N/A	N/A	0	2962.95	3959.21	3959.21
VC4(off)	N/A	N/A	N/A	N/A	824.28	2210.63	2210.63
VC5(off)	N/A	N/A	N/A	N/A	N/A	0.00	0.00
VC6(off)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Average							3788.71

6. CONCLUSIONS

This research focused on enhanced best-effort service, which provides minimum bandwidth guarantee and distributes the available bandwidth fairly to the flows. The goal has been accomplished by means of the proposed WMFQC scheme since it can achieve general weighted (GW) fair allocation. GW fairness guarantees the minimum rate for every flow and allocates the excess bandwidth (the available bandwidth minus the sum of the minimum rates) to the flows in proportion to their weights. In addition, WMFQC employs a queue control mechanism to keep the queue length below a target threshold and to drop data packets when a dropping threshold is met. The mechanism ensures good performance in terms of efficiency (measured based on fairness response time), end-to-end delay, packet loss ratio, and throughput. From the simulation results, better performance appears to be achieved when the ratio of the target threshold to the dropping threshold is set to be about 60%.

The simulation results also show that the WMFQC and Max-Min schemes require the same response time to achieve fair allocation when the computation time is not considered. In addition, a switch in the WMFQC scheme, when receiving a control packet, uses $O(\log N)$ time for bandwidth allocation among N active flows while a switch in the Max-Min scheme uses $O(N)$ time. Thus, WMFQC achieves fair allocation more efficiently than the Max-Min scheme does. It is found that the Max-Min scheme fails to re-achieve new fair allocation when some flow becomes idle. WMFQC overcomes this problem by distributing the released bandwidth of an idle flow to the unconstrained flows at the switches that the idle flow passes through and thus re-achieves new fair allocation. Finally, we conclude that the contributions of WMFQC are as follows: 1) it achieves GW fair allocation efficiently; 2) it re-establishes fair allocation by adapting to network changes when flows become active or idle; 3) it allocates bandwidth in logarithmic time; 4) it controls the queue length to achieve superior performance in terms of delay, packet loss, and throughput.

ACKNOWLEDGEMENT

The authors are very grateful to the anonymous reviewers for their comments and suggestions that helped improve the quality of this paper.

REFERENCES

1. X. Xiao and L. M. Ni, "Internet QoS: a big picture," *IEEE Network Magazine*, Vol.13, No. 2, 1999, pp. 8-18.
2. M. de Prycker, *Asynchronous Transfer Mode: Solution for Broadband ISDN*, Ellis Horwood, 1993.
3. A. Campbell *et al.*, "Integrated quality of service for multimedia communications," in *Proceedings of IEEE INFOCOM '93*, 1993, pp. 732-739.
4. R. Braden *et al.*, "Resource reservation protocol (RSVP) — version 1 functional specification," RFC 2205, Sep. 1997.
5. F. Bonomi and K. W. Fendick, "The rate-based flow control framework for the available bit rate ATM service," *IEEE Network Magazine*, Vol. 9, No. 2, 1995, pp. 25-39.
6. E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," Internet draft, draft-ietf-mpls-arch-01.txt, March 1998.
7. S. Kalyanaraman *et al.*, "The ERICA switch algorithm for ABR traffic management in ATM networks," *IEEE/ACM Transactions on Networking*, Vol. 8, No. 1, 2000, pp. 87-98.
8. B. Vandalore *et al.*, "General weighted fairness and its support in explicit rate switch algorithms," *Computer Communications*, Vol. 23, No. 2, 2000, pp. 149-161.
9. D. Bartsekas and R. Gallager, *Data Networks*, Prentice Hall, 1992.
10. A. W. Barnhart, "Explicit rate performance evaluation," ATM Forum 94-0983R1, 1994.
11. R. Jain *et al.*, "A sample switch algorithm," ATM Forum 95-0178R1, 1995.
12. 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," in *Proceedings of IEEE INFOCOM '96*, 1996, pp. 1174-1181.
13. A. Charny, "An algorithm for rate allocation in a packet-switching network with feedback," MIT/LCS/TR-601, 1994.
14. D. H. K. Tsang *et al.*, "A fast switch algorithm for ABR traffic to achieve max-min fairness," in *Proceedings of IEEE International Zurich Seminar on Digital Communications*, 1996, pp. 19-23.
15. L. Kalampoukas, A. Varma, and K. K. Ramakrishnan, "An efficient rate allocation algorithm for ATM networks providing max-min fairness," in *Proceedings of the 6th IFIP International Conference on High Performance Networking*, 1995, pp. 143-154.
16. K. Siu and T. Tzang, "Intelligent congestion control for ABR service in ATM networks," *Computer Communication Review*, Vol. 24, No. 5, 1994, pp. 81-106.
17. Y. Afek, Y. Mansour, and Z. Ostfeld, "Phantom: A simple and effective flow control scheme," in *Proceedings of ACM SIGCOMM '96 Conference*, 1996, pp. 169-182.

18. D. Hughes, "Fair share in the context of MCR," ATM Forum/AF-TM 94-0977, 1994.
19. N. Yin, "Max-min fairness vs. MCR guarantee on bandwidth allocation for ABRx," in *Proceedings of IEEE ATM '96 Workshop*, 1996.
20. Y. T. Hou, H. Tzeng and S. S. Panwar, "A simple ABR switch algorithm for the weighted max-min fairness policy," in *Proceedings of IEEE ATM'97 Workshop*, 1997, pp. 329-338.
21. S. P. Abraham and A. Kumar, "A stochastic approximation approach for a max-min fair adaptive rate control of ABR sessions with MCRs," in *Proceedings of IEEE INFOCOM'98*, 1998, pp. 1358-1365.
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," in *Proceedings of IEEE INFOCOM '98*, 1998, pp. 1366-1375.
23. A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks — the single node case," *IEEE/ACM Transactions on Networking*, Vol. 1, No. 3, 1993, pp. 344-357.
24. A. Demers, S. Keshav, and S. Shenkar, "Analysis and simulation of a fair queuing algorithm," *Internetworking: Research & Experience*, Vol. 1, No. 1, 1990, pp. 3-26.
25. S. J. Golestani, "A self-clocked fair queuing scheme for broadband applications," in *Proceedings of IEEE INFOCOM '94*, 1994, pp. 636-646.
26. D. Stiliadis and A. Varma, "Latency-rate servers: a general model for analysis of traffic scheduling algorithms," in *Proceedings of IEEE INFOCOM '96*, 1996, pp. 111-119.
27. D. Stiliadis and A. Varma, "Efficient fair queuing algorithms for packet-switched networks," *IEEE/ACM Transactions on Networking*, Vol. 6, No. 2, 1998, pp. 175-185.
28. J. C. R. Bennet and H. Zhang, "Hierarchical packet fair queuing algorithm," *IEEE/ACM Transactions on Networking*, Vol. 5, No. 5, 1997, pp. 675-689.
29. F. M. Chiussi, and A. Francini, "Minimum-delay self-clocked fair queuing algorithm for packet-switched networks," in *Proceedings of IEEE INFOCOM '98*, 1998, pp. 1112-1121.
30. Y.-J. Chen and S.-Y. Lee, "Bounded tag fair queuing for broadband packet switching networks," *Computer Communications*, Vol. 23, No. 1, 2000, pp. 45-61.
31. S. S. Sathaye, "ATM forum traffic management specification version 4.0," <ftp://ftp.atmforum.com/pub/approved-specs/af-tm-0056.0000.pdf>, 1996.
32. M. A. Weiss, *Data Structures and Algorithm Analysis in C*, Addison-Wesley, 1997.
33. Y.-J. Chen and S.-Y. Lee, "A scheme to achieve weighted max-min fairness allocation for high speed networks," in *Proceedings of Workshop on Computer Networks, Internet, and Multimedia in ICS '98*, 1998, pp. 258-265.

APPENDIX

Theorem 1. For the ordered flow list i_1, i_2, \dots, i_N , if the critical flow i_r exists (i.e., i_1, i_2, \dots, i_{r-1} are constrained and i_r, i_{r+1}, \dots, i_N are unconstrained), then $ERm.i_1/Bo.i_1 \leq \dots \leq ERm.i_{r-1}/Bo.i_{r-1} < NFS \leq ERm.i_r/Bo.i_r \leq \dots \leq ERm.i_N/Bo.i_N$.

Proof: It is clear that NFS (Normalized Fair Share) $\leq ERm.i_r/Bo.i_r$ (the normalized bandwidth limit of the critical flow i_r) from step <6> of the bandwidth allocation algorithm presented in Section 3.1 and steps <S3> and <S4> of the subroutine Find_NFS. In addition, the flows i_1, i_2, \dots, i_N are in increasing order according to their bandwidth limits, ERm/Bo . Therefore, we have $NFS \leq ERm.i_r/Bo.i_r \leq \dots \leq ERm.i_N/Bo.i_N$. On the other hand, we can show that $ERm.i_{r-1}/Bo.i_{r-1} < NFS$. From Find_NFS, NFS is the final

value of the variable $Temp_NFS$. Thus, $NFS = \frac{(C - \sum_{q=1}^{r-1} ERm.i_q)}{\sum_{q=r}^N Bo.i_q}$. From step <S3>,

we have $\frac{ERm.i_{r-1}}{Bo.i_{r-1}} < \frac{(C - \sum_{q=1}^{r-2} ERm.i_q)}{\sum_{q=r-1}^N Bo.i_q}$. Now, $NFS - ERm.i_{r-1}/Bo.i_{r-1}$ can be written as

follows:

$$NFS - ERm.i_{r-1}/Bo.i_{r-1} = \frac{(C - \sum_{q=1}^{r-2} ERm.i_q) - ERm.i_{r-1}}{\sum_{q=r-1}^N Bo.i_q - Bo.i_{r-1}} - \frac{ERm.i_{r-1}}{Bo.i_{r-1}}.$$

Since for any real numbers, a, b, c , and d , $\frac{c-a}{d-b} - \frac{a}{b} > 0$ if $\frac{c}{d} > \frac{a}{b}$ and $d > b > 0$, the result of the right side of the “=” sign in the above expression is positive. We have $ERm.i_{r-1}/Bo.i_{r-1} < NFS$. Obviously, $ERm.i_1/Bo.i_1 \leq \dots \leq ERm.i_{r-1}/Bo.i_{r-1} < NFS$. Proven.

Theorem 2. Let i_p be a flow among i_1, i_2, \dots, i_N , and let $\theta.i_p$ be the normalized share of i_p , assuming that i_1, \dots, i_{p-1} are constrained and i_p, \dots, i_N are unconstrained, where

$$\theta.i_p = \frac{C - \sum_{q=1}^{p-1} ERm.i_q}{\sum_{q=p}^N Bo.i_q}. \quad (A1)$$

In fact, flow i_p is constrained if and only if $ERm.i_p/Bo.i_p < \theta.i_p$.

Proof: If i_p is constrained, then we have $ERm.i_p/Bo.i_p < Temp_NFS$ from step <S3> of the subroutine Find_NFS presented in Section 3.1. From step <S2> and Eq. (A1), $Temp_NFS$ is equivalent to $\theta.i_p$. Thus, $ERm.i_p/Bo.i_p < \theta.i_p$. Next, we will show that if i_p is not constrained (i.e., unconstrained), then $ERm.i_p/Bo.i_p \geq \theta.i_p$. If i_p is unconstrained, its normalized bandwidth share, denoted by $R.i_p$, is $\frac{C - \sum_{q=1}^{r-1} ERm.i_q}{\sum_{q=r}^N Bo.i_q}$

and $p \geq r$, where i_r is the critical flow. Thus, we have

$$R.i_p - \theta.i_p = \frac{(C - \sum_{q=1}^{r-1} ERm.i_q)}{\sum_{q=r}^N Bo.i_q} - \frac{(C - \sum_{q=1}^{r-1} ERm.i_q) - \sum_{q=r}^{p-1} ERm.i_q}{\sum_{q=r}^N Bo.i_q - \sum_{q=r}^{p-1} Bo.i_q}.$$

Since $\frac{(C - \sum_{q=1}^{r-1} ERm.i_q)}{\sum_{q=r}^N Bo.i_q} = R.i_p = R.i_r \leq \frac{ERm.i_q}{Bo.i_q}$ for $q \geq r$, we have $R.i_p - \theta.i_p \geq 0$.

Since i_p is unconstrained, $ERm.i_p/Bo.i_p \geq R.i_p$ from Theorem 1. Therefore, $ERm.i_p/Bo.i_p \geq \theta.i_p$. Proven.



Yen-Jen Chen (陳延禎) received the B.S. degree in computer science and information engineering from Tatung Institute of Technology, Taiwan, in 1989. Currently, he is a Ph.D. candidate in the Institute of Computer Science and Information Engineering, National Chiao Tung University, Taiwan. His research interests include packet scheduling, flow control, and bandwidth allocation algorithm and quality of service in computer networks.



Suh-Yin Lee (李素瑛) received the B.S. degree in electrical engineering from National Chiao Tung University, Taiwan, in 1972, the M.S. degree in computer science from University of Washington, U.S.A., in 1975, and the Ph.D. degree in computer science from Institute of Electronics, National Chiao Tung University, Taiwan, in 1981. Currently, she is a Professor in the Department of Computer Science and Information Engineering at National Chiao Tung University. Her research interests include multimedia information system, computer network, mobile computing, data mining, and intelligent agent on web.