Max-Min Fair Bandwidth Allocation Algorithms for Packet Switches *

Deng Pan
Dept. of Computer Science
State University of New York
Stony Brook, NY 11794, USA

Yuanyuan Yang
Dept. of Electrical & Computer Engineering
State University of New York
Stony Brook, NY 11794, USA

Abstract

With the rapid development of broadband applications, the capability of networks to provide quality of service (QoS) has become an important issue. Fair scheduling algorithms are a common approach for switches and routers to support QoS. All fair scheduling algorithms are running based on a bandwidth allocation scheme. The scheme should be feasible in order to be applied in practice, and should be efficient to fully utilize available bandwidth and allocate bandwidth in a fair manner. However, since a single input port or output port of a switch has only the bandwidth information of its local flows (i.e., the flows traversing itself), it is difficult to obtain a globally feasible and efficient bandwidth allocation scheme. In this paper, we show how to fairly allocate bandwidth in packet switches based on the max-min fairness principle. We first formulate the problem, and give the definitions of feasibility and max-min fairness for bandwidth allocation in packet switches. As the first step to solve the problem, we consider the simpler unicast scenarios, and present the max-min fair bandwidth allocation algorithm for unicast traffic. We then extend the analysis to the more general multicast scenarios, and present the max-min fair bandwidth allocation algorithm for multicast traffic. We prove that both algorithms achieve max-min fairness, and analyze their complexity. The proposed algorithms are universally applicable to any type of switches and scheduling algorithms.

1 Introduction

With the rapid development of broadband networks in recent years, a variety of novel network based applications have been developed with different quality of service (QoS) [1] requirements. Network traffic can be broadly classified into two categories: guaranteed performance traffic and best effort traffic. For guaranteed performance traffic, resources are reserved for an allocated transmission rate, and the performance, such as bandwidth and delay, is bounded within a pre-specified range. On the contrary, best effort traffic tries to make the best use of available transmission capacity but has no guarantee to the quality of service.

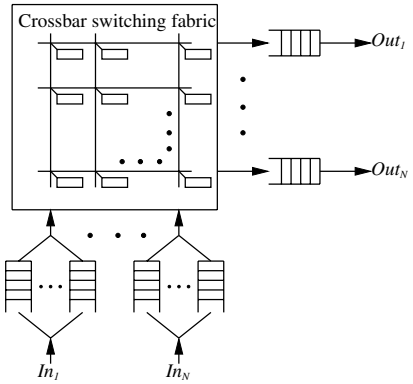

Figure 1. The general structure of a crossbar switch.

The capability to provide QoS support has become an important issue for the design of modern switches and routers [2] [3]. Switches and routers control the departure sequence of the packets stored in their buffers, and the adopted scheduling algorithms largely determine the quality of service that can be provided by the networks. Much research effort has been devoted to the design of fair scheduling algorithms for packet switches to provide QoS support, and many algorithms have been proposed in the literature for different switch architectures.

We now briefly review the switch architectures and fair scheduling algorithms, using the most popular crossbar switches as examples. In a crossbar switch, input ports and output ports are connected by a non-blocking crossbar switching fabric, as shown in Figure 1. The crossbar may be running faster than the input ports and output ports, in which case the crossbar is said to have speedup larger than one. Depending on the exact speedup value, temporarily blocked packets in a crossbar switch can be buffered at either the output ports, input ports, or crosspoints. Out-

*This research was supported in part by the U.S. National Science Foundation under grant number CCR-0207999.

put queued (OQ) switches have buffer space only at output ports. New incoming packets must be immediately transferred through the crossbar and stored in the output buffers. Since there is no buffer at the input side, if multiple input ports have packets arriving at the same time that are destined to the same output port, all the packets must be transmitted simultaneously. Speedup of M is necessary for an OQ switch with M input ports to achieve 100% throughput. Fair scheduling algorithms for OQ switches, such as WFQ [5], WF2Q [6], DRR [7] and FMCF [8], work at each output port to determine the transmission sequence of the packets in its buffer, and emulate the ideal Generalized Processor Sharing (GPS) fairness model [4]. They can provide different levels of service guarantee using different approaches. It has recently been shown [9] that there is a fundamental tradeoff between the delay bound that an algorithm can achieve and its computational complexity.

On the other hand, input queued (IQ) switches have buffer space only at input ports, and thus eliminate the speedup requirement. Input buffers are usually organized as multiple virtual output queues (VOQ) [10], to avoid the Head of Line (HOL) blocking caused by the traditional single FIFO queue, which restricts the maximum throughput of the switches [11] [12]. Unfortunately, until now IQ switches are found to be able to achieve 100% throughput only when they work with maximum matching algorithms or their variants [11], which have high time complexity [13] [14]. One type of fair scheduling algorithms for IQ switches try to emulate the corresponding fair scheduling algorithms for OQ switches with iterative matching. For example, iFS [15] emulates WFQ [5] by using time stamps, and iDRR [16] emulates DRR [7] by using round robin pointers. Another algorithm WPIM [17] improves upon PIM [18] by introducing a bandwidth enforcement mechanism to provide probabilistic bandwidth guarantee for input-output connections.

In order to combine the advantages of both output queued switches and IQ switches, combined input-output queued (CIOQ) switches make a tradeoff between the crossbar speedup and the complexity of the scheduling algorithm. They usually have a small, fixed speedup of two, and thus need buffer space at both the input side and output side. Buffered crossbar switches, or combined input-crosspoint-output queued (CICOQ) switches, are a special type of CIOQ switches, where each crosspoint of the crossbar is equipped with a small buffer. Crosspoint buffers eliminate output and input scheduling contention, and enable the switches to work in an asynchronous mode and easily transmit variable length packets. Both CIOQ switches and CIOCQ switches are proved to be able to perfectly emulate OQ switches with a small speedup [19] [20] [21]. Thus, special scheduling algorithms for CIOQ switches or CIOCQ switches can be designed to duplicate the packet departure sequence of existing fair scheduling algorithms for OQ switches, and provide desired service guarantee.

Regardless of the architecture of a switch and the fair scheduling algorithm the switch uses, it is necessary to predefine a feasible and efficient bandwidth allocation scheme as the basis of the scheduling. The bandwidth allocation scheme specifies the amount of bandwidth that an input port or a flow can use at each output port of the switch to transmit packets. On the one hand, the scheme must be feasible in order to be applied in practice. In other words, the total bandwidth allocated to all the flows at any input port or output port cannot exceed its available bandwidth. On the other hand, the scheme should be efficient, which means to fully utilize any potential transmission capacity and allocate bandwidth in a fair manner.

A bandwidth allocation scheme must be carefully designed in order to be feasible and efficient. Before packet transmission, an input port claims portion of the bandwidth of each output port for its traffic. (Alternatively, an output port can allocate its available bandwidth to different input ports.) However, since each input port or output port has only local bandwidth information, it does not know how much bandwidth other input ports claimed at a specific output port (or how much bandwidth other output ports allocated to a specific input port). Thus, this initial bandwidth allocation scheme may be under-utilized or over-utilized. For the under-utilized case, clearly, the unused bandwidth should be allocated to make full use of the transmission capacity, and it has to be carefully handled to allocate the leftover bandwidth in a fair manner. For the over-utilized case, it is also necessary to fairly scale down the claimed bandwidth of each user to make the scheme feasible.

The bandwidth allocation scheme plays several important roles in guaranteeing the high performance of a switch. First of all, the scheme is used as the scheduling criterion by fair scheduling algorithms. The scheduling algorithms make decisions on the departure sequence of packets from different users, so that the bandwidth that each user actually receives is equal to the amount that it is allocated in the scheme. Secondly, the scheme helps to determine the traffic admission policy and buffer management strategy. In order to reduce cost, modern switches usually put packets from different users in shared buffers. Without proper admission control, excessive traffic from one user may cause buffer overflow and packet loss for other users. Also, if the shared buffer is not well managed, and a user has no available packet in the buffer when it is its turn to transmit, there is no way to provide bandwidth or delay guarantee. Thus, it is important to determine how much traffic can be admitted for a user and how the shared buffer should be managed based on the amount of bandwidth allocated to that user in the scheme. Thirdly, an efficient scheme makes it possible for a switch to achieve 100% throughput. Throughput is an important criterion to measure the performance of a switch. There has been much work on how to achieve 100% throughput, and all the proposed approaches require an effi-

cient bandwidth allocation scheme based on which no output port should be under-utilized or over-utilized.

In this paper, we present algorithms to compute fair bandwidth allocation schemes based on the max-min fairness principle. The presented algorithms can be universally applied to to any type of switches and scheduling algorithms. Max-min fairness has long been used as a popular fairness principle in resource allocation [22] [23] [24] [25]. In particular, [25] discussed how to compute max-min fair rate allocation for flows in IQ switches. However, it analyzed only the simple unicast scenarios where there are only best effort flows, and did not consider guaranteed performance flows or the more complex multicast scenarios.

We first formulate the problem and define some terminologies that will be used. Then, we give the definitions of feasibility and max-min fairness for bandwidth allocation in packet switches. As the first step of the analysis, we consider the simple unicast scenarios, and present the max-min fair bandwidth allocation algorithm for unicast traffic. We then prove that the presented algorithm achieves max-min fairness. Next, we extend the discussion to the more general multicast scenarios, and present the max-min fair bandwidth allocation algorithm for multicast traffic, which is also proved to achieve max-min fairness. For both algorithms, we give examples to illustrate the operation of the algorithms and analyze their complexity.

2 Max-min Fair Bandwidth Allocation for Unicast Traffic

In this section, we formulate the max-min fair bandwidth allocation problem, and present an algorithm for the simpler scenarios where there is only unicast traffic. We will analyze the more general scenarios with multicast traffic in Section 3.

In order to make the analytical results more widely applicable, we consider a general switch with M input ports and N output ports, without any specific assumption on the switch architecture. For easy representation, denote the i^{th} input port by In_i and the j^{th} output port by Out_j. Input bandwidth vector IB defines the transmission capacity of all input ports, where the i^{th} entry IB_i indicates the available bandwidth of In_i. Similarly, output bandwidth vector OB gives the transmission capacity of all output ports, and OB_j is the available bandwidth of Out_j.

There are guaranteed performance flows as well as best effort flows in the switch. The former require exclusively reserved bandwidth, and the latter utilize the leftover bandwidth from the former. We solve the max-min fair bandwidth allocation problem in two phases. Phase one tries to satisfy the request for exclusive bandwidth of guaranteed performance flows, and phase two equally allocates the remaining bandwidth to best effort flows. Fortunately, both phases can be handled in a similar way.

With pure unicast traffic, any flow has one source input port and one destination output port. Thus, in phase one, all the guaranteed performance flows of the same input-output pair (i.e., leaving from the same input port and destined to the same output port) are subject to the same bandwidth constraints, and can be viewed as a single logical flow in the analysis. We denote the logical flow from In_i to Out_j as F_{ij}. In the following description of phase one, if not specifically noted, a flow means a logical guaranteed performance unicast flow. Similarly, in phase two, all the best effort flows of the same input-output pair can also be considered as a single logical flow.

Before the transmission of packets, each flow claims its desired bandwidth, which we call the requested bandwidth. We use an $M \times N$ matrix R to represent the requested bandwidth of all flows, where entry R_{ij} is the requested bandwidth of flow F_{ij} or the desired bandwidth of In_i at Out_j. If In_i does not have guaranteed performance traffic to Out_j, then R_{ij} is set to zero.

We use another $M \times N$ matrix A to represent the allocated bandwidth of all flows, where entry A_{ij} is the actual bandwidth allocated to flow F_{ij}, or the amount of bandwidth that In_i can use at Out_j.

Furthermore, we define the satisfaction degree of flow F_{ij} to be the ratio of its allocated bandwidth to the requested bandwidth, and denote it as S_{ij}, i.e.,

$$S_{ij} = \frac{A_{ij}}{R_{ij}}$$

When $R_{ij} = 0$, both A_{ij} and S_{ij} are set to zero. Flow F_{ij} is said to be satisfied or a satisfied flow if its allocated bandwidth is equal to its requested bandwidth, or $S_{ij} = 1$, otherwise, F_{ij} is unsatisfied. We call the matrix S formed by all S_{ij} the satisfaction matrix.

Now we can define the feasibility of bandwidth allocation for unicast traffic. We say that an allocation matrix A is feasible with respect to input bandwidth vector IB, output bandwidth vector OB and request matrix R, or simply A is feasible, if no flow is allocated more bandwidth than what it requests, i.e.,

$$\forall i \forall j A_{ij} \leq R_{ij}$$

and there is no oversubscription at any input port or output port, i.e.,

$$\forall i \sum_j A_{ij} \leq IB_i, \forall j \sum_i A_{ij} \leq OB_j$$

Since the satisfaction matrix and allocation matrix have one-to-one correspondence, we also say that S is feasible when its corresponding A is feasible.

Note that feasibility only makes a bandwidth allocation scheme possible to be applied in practice. However, a feasible scheme may not be an efficient one. Thus, we adopt max-min fairness to make the best use of available bandwidth and allocate bandwidth in a fair manner.

We say that an allocation matrix A for unicast traffic is max-min fair with respect to input bandwidth vector IB, output bandwidth vector OB and request matrix R, or simply A is max-min fair, if it is feasible and it is impossible to increase the allocated bandwidth of any flow without reducing the allocated bandwidth of another flow with lower satisfaction degree. Similarly, when A is max-min fair, we also say that its corresponding satisfaction matrix S is max-min fair. Formally, a feasible satisfaction matrix S is max-min fair, if for any feasible satisfaction matrix S' the following condition holds

$$S'_{ij} > S_{ij} \rightarrow \exists i' \exists j' \left(S_{i'j'} \leq S_{ij} \wedge S_{i'j'} > S'_{i'j'} \right)$$

Intuitively, the objective of max-min fairness is two folds: on the one hand, to increase the satisfaction degree of each flow as much as possible, so as to make the best use of available bandwidth; on the other hand, to maximize the minimum satisfaction degree of all the flows to achieve fairness, which also explains the meaning of the term "max-min."

We have the following theorem concerning the max-min fair satisfaction matrix for unicast traffic.

Theorem 1 *The max-min fair satisfaction matrix for unicast traffic is unique.*

Proof: We prove it by contradiction. Suppose that both satisfaction matrices S and S' are max-min fair, and $S \neq S'$.

Without loss of generality, assume that S_{ij} is the smallest entry among all the entries in S that are different from their corresponding entries in S', i.e.,

$$S_{ij} \neq S'_{ij} \wedge \forall i' \forall j' \left(S_{i'j'} \neq S'_{i'j'} \rightarrow S_{i'j'} \geq S_{ij} \right)$$

There are two possible cases as to the relationship between S_{ij} and S'_{ij}: $S_{ij} > S'_{ij}$ and $S'_{ij} > S_{ij}$. In the case that $S'_{ij} > S_{ij}$, since S is max-min fair, according to the definition, there exists i'' and j'' such that $S_{i''j''} \leq S_{ij}$ and $S_{i''j''} > S'_{i''j''}$. Thus, we can always have p and q, such that $S_{ij} \geq S_{pq} \wedge S_{pq} > S'_{pq}$, where $p = i$ and $q = j$ in the case $S_{ij} > S'_{ij}$ and $p = i''$ and $q = j''$ in the case $S'_{ij} > S_{ij}$.

Since $S_{pq} > S'_{pq}$ and S' is max-min fair, by the definition, there must exist p' and q' such that $S'_{p'q'} \leq S'_{pq}$ and $S'_{p'q'} > S_{p'q'}$, and thus $S'_{pq} > S_{p'q'}$. Noticing that $S_{p'q'} \neq S'_{p'q'}$ and that S_{ij} is the smallest different entry in S, we can obtain $S_{p'q'} \geq S_{ij}$. Combining with the previous inequality $S'_{pq} > S_{p'q'}$, we have $S'_{pq} > S_{ij}$, which is a contradiction with the fact that $S_{ij} > S'_{pq}$.

Therefore, S and S' must be equal, and the max-min fair satisfaction matrix is unique. ■

Next, we give the definition of bottleneck ports. Given a satisfaction matrix, a port is the bottleneck port of a flow if the flow has the highest satisfaction degree among all the flows traversing the port, and the bandwidth of the port is fully allocated. Formally, In_i is a bottleneck port of flow F_{ij} in satisfaction matrix S if

$$\forall j' S_{ij} \geq S_{ij'} \wedge \sum_q S_{iq} R_{iq} = IB_i$$

and Out_j is a bottleneck port of F_{ij} in S if

$$\forall i' S_{ij} \geq S_{i'j} \wedge \sum_p S_{pj} R_{pj} = OB_j$$

Theorem 2 *A feasible satisfaction matrix for unicast traffic is max-min fair if and only if each unsatisfied flow has a bottleneck port.*

Proof: First, we prove the "if" part. Assume S is feasible and each unsatisfied flow has a bottle port in S. We will prove that S is max-min fair using the definition of max-min fairness.

Suppose S' is a feasible satisfaction matrix and $S'_{ij} > S_{ij}$. Then we know that $S_{ij} < S'_{ij} \leq 1$, and F_{ij} is an unsatisfied flow in S. Since each unsatisfied flow has a bottleneck port in S, we first assume that In_i is a bottleneck port of F_{ij} in S. By the definition of bottleneck ports, we know that $\forall j' S_{ij} \geq S_{ij'}$ and $\sum_q S_{iq} R_{iq} = IB_i$. On the other hand, since S' is feasible, we have $\sum_q S'_{iq} R_{iq} \leq IB_i$, and it follows that $\sum_q S'_{iq} R_{iq} \leq \sum_q S_{iq} R_{iq}$. Because $S'_{ij} > S_{ij}$, there must exist j' such that $S_{ij'} > S'_{ij'}$, otherwise we can obtain the contradiction that $\sum_q S'_{iq} R_{iq} > \sum_q S_{iq} R_{iq}$. Noticing that $S_{ij} \geq S_{ij'}$, we have found $i' = i$ and j' such that $S_{i'j'} \leq S_{ij}$ and $S_{i'j'} > S'_{i'j'}$, and thus S is max-min fair. Similar reasoning can be applied to the case that Out_j is a bottleneck port of F_{ij} in S.

Now we prove the "only if" part by contradiction. Assume that S is max-min fair, but an unsatisfied flow F_{ij} (thus $S_{ij} < 1$) has no bottleneck port in S.

First, consider two possible cases that In_i is not a bottleneck port of F_{ij} in S.

Case 1: There exists another flow from In_i with higher satisfaction degree, i.e., $\exists j' S_{ij'} > S_{ij}$. Let

$$D'_{xy} = \begin{cases} \frac{S_{ij'} - S_{ij}}{R_{ij'} + R_{ij}} R_{ij'} R_{ij}, & \text{if } x = i \text{ and } y = j \\ -\frac{S_{ij'} - S_{ij}}{R_{ij'} + R_{ij}} R_{ij'} R_{ij}, & \text{if } x = i \text{ and } y = j' \\ 0, & \text{otherwise} \end{cases}$$

Case 2: The bandwidth of In_i is not fully allocated, i.e., $IB_i > \sum_q S_{iq} R_{iq}$. Let

$$D'_{xy} = \begin{cases} \min\{R_{ij}(1 - S_{ij}), IB_i - \sum_q S_{iq} R_{iq}\}, & \\ & \text{if } x = i \text{ and } y = j \\ 0, & \text{otherwise} \end{cases}$$

By adding $\frac{D'_{xy}}{R_{xy}}$ to S_{xy}, the resulting matrix still maintains feasibility, and also satisfies the bandwidth constraint of In_i. Note that $D'_{ij} > 0$ in both cases. It is the amount of bandwidth that can be added to the allocated bandwidth

of flow F_{ij} without reducing the satisfaction degree of any flow with lower satisfaction degree than S_{ij}.

Similarly, there are two possible cases when Out_j is not a bottleneck port of F_{ij} in S.

Case 1: There exists another flow to Out_j with higher satisfaction degree, i.e., $\exists i' S_{i'j} > S_{ij}$. Let

$$D''_{xy} = \begin{cases} \frac{S_{i'j}-S_{ij}}{R_{i'j}+R_{ij}} R_{i'j} R_{ij}, & \text{if } x = i \text{ and } y = j \\ -\frac{S_{i'j}-S_{ij}}{R_{i'j}+R_{ij}} R_{i'j} R_{ij}, & \text{if } x = i' \text{ and } y = j \\ 0, & \text{otherwise} \end{cases}$$

Case 2: The bandwidth of Out_j is not fully allocated, i.e., $OB_j > \sum_p S_{pj} R_{pj}$. Let

$$D''_{xy} = \begin{cases} \min\{R_{ij}(1 - S_{ij}), OB_j - \sum_p S_{pj} R_{pj}\}, & \text{if } x = i \text{ and } y = j \\ 0, & \text{otherwise} \end{cases}$$

The satisfaction matrix formed by $S_{xy} + \frac{D''_{xy}}{R_{xy}}$ is feasible and satisfies the bandwidth constraint of Out_j, and D''_{ij} (> 0) is the amount of bandwidth that can be added to $R_{ij} S_{ij}$ without reducing the satisfaction degree of any flow with lower satisfaction degree.

Considering the constraints at both In_i and Out_j, define matrix D as follows

$$D_{xy} = \begin{cases} \min\{D'_{ij}, D''_{ij}\}, & \text{if } x = i \text{ and } y = j \\ -\min\{D'_{ij}, D''_{ij}\}, & \text{if } x = i \text{ and } y = j' \text{ and } D'_{ij'} \neq 0 \\ -\min\{D'_{ij}, D''_{ij}\}, & \text{if } x = i' \text{ and } y = j \text{ and } D''_{i'j} \neq 0 \\ 0, & \text{otherwise} \end{cases}$$

Create a new satisfaction matrix S' where $S'_{xy} = S_{xy} + \frac{D_{xy}}{R_{xy}}$. It is easy to verify that S' is still feasible. Since $D_{ij} > 0$, $S'_{ij} > S_{ij}$. According to the definition of D, we know that the only other entries in S' except S'_{ij} that may be different from the corresponding entries in S are $S'_{ij'}$ and $S'_{i'j}$. Because S is max-min fair and $S'_{ij} > S_{ij}$, by the definition of max-min fairness, either $S'_{ij'}$ and $S'_{i'j}$ must be smaller than its counterpart in S, i.e., $S'_{ij'} < S_{ij'} \vee S'_{i'j} < S_{i'j}$.

If $S'_{ij'} < S_{ij'}$ or $D_{ij'} < 0$, by the construction process, we know that $D'_{ij'} \neq 0$, which indicates that $S_{ij'} > S_{ij}$. Similarly, we can obtain that if $S'_{i'j} < S_{i'j}$, then $S_{i'j} > S_{ij}$. Thus, there do not exist i' and j' such that $S_{i'j'} \leq S_{ij}$ and $S_{i'j'} > S'_{i'j'}$, which contradicts the assumption that S is max-min fair. ■

Based on Theorem 2, we now present the max-min fair bandwidth allocation algorithm for unicast traffic. The basic idea of the algorithm is to find the bottleneck ports of unsatisfied flows in an iterative manner. After either each flow is satisfied or a bottleneck port is identified for it, the algorithm converges with a max-min fair satisfaction matrix.

The pseudo code description of the algorithm is given in Table 1. The input parameters of the algorithm are the input bandwidth vector IB, output bandwidth vector OB and

Table 1. Max-min Fair Bandwidth Allocation Algorithm for Unicast Traffic

Input: IB, OB, R
Output: S (The initial value of each entry in S is 0.)

```
01) for each In_i, IR_i = Σ_j R_ij;
02) for each Out_j, OR_j = Σ_i R_ij;
03) while (∃i IR_i > 0) {
04)     select In_p such that ∀i IB_p/IR_p ≤ IB_i/IR_i;
05)     select Out_q such that ∀j OB_q/OR_q ≤ OB_j/OR_j;
06)     if (IB_p/IR_p ≤ OB_q/OR_q) {
07)         if (IB_p/IR_p ≤ 1) {
08)             for each F_pj, if(R_pj ≠ 0 && S_pj = 0) S_pj = IB_p/IR_p;
09)             IB_p = 0;
10)         }
11)         else {
12)             for each F_pj, if(R_pj ≠ 0 && S_pj = 0) S_pj = 1;
13)             IB_p = IB_p − IR_p;
14)         }
15)         IR_p = 0;
16)         for each Out_j, if (OR_j ≠ 0) {
17)             OB_j = OB_j − R_pj S_pj;
18)             OR_j = OR_j − R_pj;
19)         }
20)     }
21)     else {
22)         if (OB_q/OR_q ≤ 1) {
23)             for each F_iq, if(R_iq ≠ 0 && S_iq = 0) S_iq = OB_q/OR_q;
24)             OB_q = 0;
25)         }
26)         else {
27)             for each F_iq, if(R_iq ≠ 0 && S_iq = 0) S_iq = 1;
28)             OB_q = OB_q − OR_q;
29)         }
30)         OR_q = 0;
31)         for each In_i, if (IR_i ≠ 0) {
32)             IB_i = IB_i − R_iq S_iq;
33)             IR_i = IR_i − R_iq;
34)         }
35)     }
36) }
```

request matrix R, and it generates a max-min satisfaction matrix S with respect to IB, OB and R. Before running the algorithm, each entry in S is initialized to 0. We define the bandwidth share of a port to be the ratio of its total available bandwidth to the total requested bandwidth, i.e., the amount of bandwidth that can be allocated to each unit of the requested bandwidth.

In steps 1 and 2 of the algorithm, two vectors IR and OR are initialized with the total requested bandwidth at each input port and output port, respectively. During the execution of the algorithm, when all the flows of an input port or output port have been assigned satisfaction degrees, the corresponding entry in IR or OR will be cleared to zero, which means that there is no more bandwidth request. Step 3 starts the iteration loop, and the algorithm converges when no input port has any bandwidth request. Since any flow must be associated with an input port, when there is no bandwidth request at any input port, all the flows have been assigned satisfaction degrees. Steps 4 and 5 find the input port and output port with the smallest bandwidth share respectively. Steps 6 to 20 handle the case when In_p has the smallest bandwidth share among all ports. Step 7 checks whether the bandwidth share of In_p is less than or equal to one. If it is, step 8 assigns the bandwidth share as the satisfaction degree for all flows of In_p that have not been assigned satisfaction degrees yet, and step 9 sets the leftover bandwidth of In_p to zero. If the bandwidth share of In_p is greater than one, in order for the resulting satisfaction matrix to be feasible, step 12 assigns the satisfaction degree of any un-allocated flow to be the maximum value 1, and step 15 sets the leftover bandwidth of In_p accordingly. Now all the flows of In_p have been assigned satisfaction degrees, and therefore step 15 clears the bandwidth request of In_p to zero. On the other hand, steps 16 to 19 update the available bandwidth and requested bandwidth of the remaining output ports by removing the flows of In_p from consideration. Similarly, steps 21 to 35 handle the case when Out_q has the smallest bandwidth share.

We give an example to illustrate the operation of the algorithm. Consider a 3×3 switch, and each input port and output port have a unit of available bandwidth. The request matrix R is given as follows. Before running the algorithm, vectors IR and OR are initialized. Then the iteration loop begins. In iteration 1, Out_1 has the smallest bandwidth share and all flows to Out_1 are assigned a satisfaction degree $\frac{OB_1}{OR_1} = \frac{2}{3}$. Set OB_1 and OR_1 to zero, and update the entries in IB and IR accordingly. In iteration 2, In_1 has the smallest bandwidth share and all the remaining flows from In_1 are assigned a satisfaction degree $\frac{4}{5}$. In iteration 3, all the remaining flows from In_2 are satisfied, and the algorithm converges.

Input:

$$R = \begin{Bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & 0 & 0 \end{Bmatrix}$$
$$IB = \{1\ 1\ 1\}$$
$$OB = \{1\ 1\ 1\}$$
$$IR = \{\frac{4}{3}\ \frac{7}{6}\ \frac{1}{2}\}$$
$$OR = \{\frac{3}{2}\ \frac{5}{6}\ \frac{2}{3}\}$$

Iteration 1:

$$S = \begin{Bmatrix} \frac{2}{3} & 0 & 0 \\ \frac{2}{3} & 0 & 0 \\ \frac{2}{3} & 0 & 0 \end{Bmatrix}$$
$$IB = \{\frac{2}{3}\ \frac{2}{3}\ \frac{2}{3}\}$$
$$OB = \{0\ 1\ 1\}$$
$$IR = \{\frac{5}{6}\ \frac{2}{3}\ 0\}$$
$$OR = \{0\ \frac{5}{6}\ \frac{2}{3}\}$$

Iteration 2:

$$S = \begin{Bmatrix} \frac{2}{3} & \frac{4}{5} & \frac{4}{5} \\ \frac{2}{3} & 0 & 0 \\ \frac{2}{3} & 0 & 0 \end{Bmatrix}$$
$$IB = \{0\ \frac{2}{3}\ \frac{2}{3}\}$$
$$OB = \{0\ \frac{3}{5}\ \frac{11}{15}\}$$
$$IR = \{0\ \frac{2}{3}\ 0\}$$
$$OR = \{0\ \frac{1}{3}\ \frac{1}{3}\}$$

Iteration 3:

$$S = \begin{Bmatrix} \frac{2}{3} & \frac{4}{5} & \frac{4}{5} \\ \frac{2}{3} & 1 & 1 \\ \frac{2}{3} & 0 & 0 \end{Bmatrix}$$
$$IB = \{0\ 0\ \frac{2}{3}\}$$
$$OB = \{0\ \frac{4}{15}\ \frac{7}{15}\}$$
$$IR = \{0\ 0\ 0\}$$
$$OR = \{0\ 0\ 0\}$$

Theorem 3 *The max-min fair bandwidth allocation algorithm for unicast traffic achieves max-min fairness.*

Proof: First, it can be observed that the flows that are assigned satisfaction degrees in later iterations have larger values. This is because in each iteration, the port with smallest bandwidth share is selected, and the bandwidth share is assigned as the satisfaction degree to all the flows in this iteration. Thus, for any flow that is assigned a satisfaction degree in step 8 or step 23, its satisfaction degree is larger than or equal to the satisfaction degree of any flow that is assigned a satisfaction degree before the current iteration. Also note that any unsatisfied flow can only be assigned a satisfaction degree either in step 8 or step 23, and that the flows that are assigned satisfaction degrees in step 12 or step 27 are satisfied flows. Furthermore, we can see from step 9 or step 24 that the available bandwidth of In_p or Out_q is fully allocated. Therefore, In_p or Out_q must be the bottleneck port of any unsatisfied flow that is assigned a satisfaction degree in step 8 or step 23. Based on Theorem 2, we know that S is max-min fair. ■

Due to the one-to-one correspondence relationship, after S is obtained, its corresponding max-min fair allocation matrix A can be easily obtained as well, which gives the allocated bandwidth of the logical guaranteed performance flow F_{ij}.

By now we have completed phase one of bandwidth allocation. All guaranteed performance flows have been allocated bandwidth based on the max-min fairness principle. However, at this time, some input ports and output ports may still have leftover bandwidth, which can be used to transmit best effort traffic. Fortunately, the allocation of

leftover bandwidth to best effort flows can be done using the same algorithm but with different input parameters, which we denote as $\overline{IB}$, $\overline{OB}$ and $\overline{R}$, respectively. $\overline{IB}$ and $\overline{OB}$ should be set as the values of IB and OB when the algorithm for phase one finishes, i.e., the unused bandwidth of all input ports and output ports. The initialization of $\overline{R}$ can be quite flexible. For example, $\overline{R}_{ij}$ can be set to either IB_i or OB_j, which means that the logical best effort flow requests the whole bandwidth of its input port or output port. With $\overline{IB}$, $\overline{OB}$ and $\overline{R}$, the algorithm then generates the max-min satisfaction matrix for best effort traffic.

Now we analyze the time and space complexities of the algorithm. Since each iteration of the algorithm assigns satisfaction degrees to the flows of an input port or output port, it converges with $M + N$ iterations in the worst case. Note that any "for each ..." operation can be done in parallel. Thus, in each iteration, the most time consuming operation is to compare the bandwidth share of all ports to find the smallest one. The comparison can be done in parallel and has time complexity $O(\log(M + N))$. Thus, the time complexity of the algorithm is $O((M + N)\log(M + N))$. Considering that the algorithm needs to be executed only once each time after the requested bandwidth of the flows changes, the time complexity is acceptable. As to the space complexity, except the input parameters, the only extra variables used in the algorithm are vectors IR and OR. Thus, the algorithm has space complexity of $O(M + N)$, which is moderate.

3 Max-min Fair Bandwidth Allocation Algorithms for Multicast Traffic

In Section 2, we studied how to fairly allocate bandwidth for unicast traffic in packet switches. In this section, we extend the discussion to the more general scenarios where multicast traffic exists. The bandwidth allocation problem for multicast can also be solved in two phases, in which phase one allocates bandwidth to guaranteed performance multicast flows and phase two deals with best effort flows.

Multicast is the data transmission from one source to multiple destinations, and is especially of interest for Internet multimedia applications, such as teleconference, distance learning and video on demand. The simplest way to process a multicast packet is to create multiple copies of the packet and send each copy as an independent unicast packet to one of the destinations. However, some switches, such as crossbar switches, have built-in multicast support to simultaneously send one packet to multiple output ports, and smartly scheduling multicast traffic on these switches can greatly save network bandwidth and reduce multicast latency. In input buffers, a multicast packet is usually saved as a single copy in order to save space. A pointer based queueing scheme, similar to that in [26], can be used to efficiently organize multicast packets in the buffer. The fair bandwidth allocation problem that we will discuss does not rely on any specific hardware architecture, and is generally applicable to all packet switches with multicast traffic.

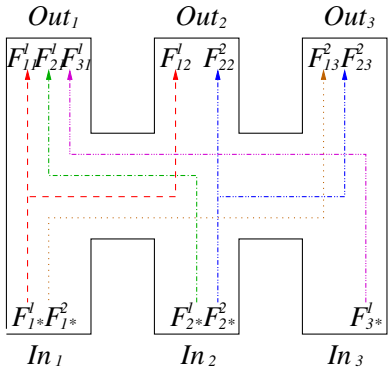

Figure 2. An example of multicast flows.

Multicast traffic has some significant differences from unicast traffic. With unicast traffic, any flow has only one destination. As analyzed in Section 2, unicast flows of the same input-output pair can be viewed as a single logical flow. However, this simplification approach does not work for multicast traffic. Since a multicast flow may be destined to several different output ports, its bandwidth allocation is constrained by the available bandwidth of all these output ports. Thus, when analyzing multicast traffic, we can combine only the flows of the same input-multicast destination pair, that is, the flows leaving from the same input port and destined to the same set of output ports. As a result, with multicast traffic, there can be as many as $2^N - 1$ different logical flows at each input port, comparing to N with unicast traffic.

Thus, the notation representing a flow needs to be revised to reflect the above observation. We assign each logical flow from the same input port a unique number k, and identify the flow with this number. For example, the k^{th} flow from input port In_i is now denoted by F^k_{i*}. We define a branch of a multicast flow to be the packet transmission sequence from the source input port to one of the destination output ports, and represent the branch of F^k_{i*} to Out_j as F^k_{ij}. For example, in Figure 2, the multicast flow F^1_{1*} has two branches: F^1_{11} to Out_1 and F^1_{12} to Out_2.

As discussed above, with multicast traffic, there are more than one logical flows for an input-output pair, and the bandwidth request or allocation matrix is no longer a $M \times N$ matrix, but can have as many rows as the number of different flows. Each row of the request or allocation matrix represents the requested or allocated bandwidth of a specific flow at all the output ports. If there are K_i different logical flows at In_i, we can use the $(\sum_{p=1}^{i-1} K_p + k)^{th}$ row to represent flow F^k_{i*}.

Use R^k_{ij} to denote the request of F^k_{ij}, or the requested bandwidth of flow F^k_{i*} at Out_j, and A^k_{ij} to denote the allocation of F^k_{i*}, or the allocated bandwidth of flow F^k_{i*} at Out_j. Define S^k_{ij} to be the ratio of A^k_{ij} to R^k_{ij}, and call it

the satisfaction degree of F_{ij}^k or F_{i*}^k at Out_j, i.e.,

$$S_{ij}^k = \frac{A_{ij}^k}{R_{ij}^k}$$

The matrix S formed by S_{ij}^k is called the satisfaction matrix. When Out_j is not a destination of F_{i*}^k, all R_{ij}^k, A_{ij}^k and S_{ij}^k are set to zero.

We define the following function to describe the fanout property of a multicast flow:

$$f(F_{i*}^k) = \{Out_j | R_{ij}^k > 0\}$$

In other words, $f(F_{ij}^k)$ is the set of output ports that flow F_{i*}^k are destined to. For the example in Figure 2, $f(F_{1*}^1) = \{Out_1, Out_2\}$.

Note that although a multicast flow have multiple branches destined to different output ports, it does not make sense for these branches to have different requested or allocated bandwidth. In practice, all the branches of a multicast flow have packets arriving at the same rate. If a branch has more bandwidth than this rate, the excessive portion cannot be utilized. On the other hand, if the bandwidth of a branch is smaller than this rate, there must be some packets that cannot be transmitted and are jammed in the buffer. In the rest of the discussion, we make the assumption that a multicast flow always requests and is allocated the same amount of bandwidth at all of its destination outputs. As a consequence, all the branches of a multicast flow have the same satisfaction degree as well due to the one-to-one correspondence between the allocation and the satisfaction. More formally,

$$Out_j \in f(F_{i*}^k) \wedge Out_{j'} \in f(F_{i*}^k) \rightarrow R_{ij}^k = R_{ij'}^k \wedge S_{ij}^k = S_{ij'}^k$$

The second significant difference between multicast traffic and unicast traffic is that bandwidth allocation for multicast traffic has different feasibility criterion. For a multicast flow F_{i*}^k, it is allocated A_{ij}^k (where $Out_j \in f(F_{i*}^k)$) bandwidth at each of its output ports, and the total bandwidth that flow F_{i*}^k needs at the output side for all branches is $|f(F_{i*}^k)|A_{ij}^k$. On the contrary, at the input side, there is only a single copy of traffic, and it needs A_{ij}^k bandwidth. Thus, the feasibility of bandwidth allocation for multicast traffic should be adjusted.

IB and OB still denote the input bandwidth vector and output bandwidth vector, respectively. We say that an allocation matrix A for multicast traffic or its corresponding satisfaction matrix S is feasible with respect to IB, OB and R if no branch is allocated more bandwidth than its requested bandwidth, i.e.,

$$\forall i \forall j \forall k A_{ij}^k \leq R_{ij}^k$$

and there is no oversubscription at any input, i.e.,

$$\forall i \sum_{j,k} \frac{A_{ij}^k}{|f(F_{i*}^k)|} \leq IB_i$$

and there is no oversubscription at any output, i.e.,

$$\forall j \sum_{i,k} A_{ij}^k \leq OB_j$$

Next, we define max-min fairness of bandwidth allocation for multicast traffic. A bandwidth allocation matrix A for multicast traffic is max-min fair with respect to IB, OB and R, if it is feasible, and it is impossible to increase the allocated bandwidth of any flow without reducing the allocated bandwidth of a flow with a lower satisfaction degree. When A is max-min fair, we also say that its corresponding satisfaction matrix S is max-min fair. Formally, if satisfaction matrix S is max-min fair, for any feasible matrix S', the following condition holds

$$S'^k_{ij} > S_{ij}^k \rightarrow \exists i' \exists j' \exists k' S_{i'j'}^{k'} \leq S_{ij}^k \wedge S_{i'j'}^{k'} > S'^{k'}_{i'j'}$$

Theorem 4 *The max-min fair bandwidth allocation matrix for multicast traffic is unique.*

The proof is similar to that of Theorem 1 and is omitted.

Given a satisfaction matrix for multicast traffic, a port is a bottleneck port of a multicast flow if the flow has the highest satisfaction degree among all the flows traversing the port and the bandwidth of the port is fully allocated. Formally, In_i is a bottleneck port of flow F_{i*}^k if

$$\exists j \left(R_{ij}^k > 0 \wedge \forall j' \forall k' S_{ij}^k \geq S_{ij'}^{k'} \right) \wedge \sum_{q,r} \frac{R_{iq}^r S_{iq}^r}{|f(F_{i*}^r)|} = IB_i$$

and Out_j is a bottleneck port of flow F_{i*}^k if

$$\left(R_{ij}^k > 0 \wedge \forall i' \forall k' S_{ij}^k \geq S_{i'j}^{k'} \right) \wedge \sum_{p,r} R_{pj}^r S_{pj}^r = OB_j$$

It should be noted that a multicast flow may have more than one bottleneck output ports.

Theorem 5 *A feasible satisfaction matrix for multicast traffic is max-min fair if and only if each unsatisfied flow has at least one bottleneck port.*

The proof of the theorem is similar to that of Theorem 2, and is omitted.

Based on Theorem 5, we present the max-min fair bandwidth allocation algorithm for multicast traffic, whose pseudo code description is given in Table 2.

Similar to the unicast scenarios, the algorithm finds the port with the smallest bandwidth share in each iteration and assigns satisfaction degrees to all the flows of the port, and the algorithm converges when no input port has any bandwidth request. However, due to the introduction of multicast traffic, the requested bandwidth of a branch at its input port is divided by its fanout size. When the flows of an input port has been assigned satisfaction degrees, all the branches of these flows should be removed from the remaining output ports. Similarly, when the flows of an output port has been

Table 2. Max-min Fair Bandwidth Allocation Algorithm for Multicast Traffic

Input: IB, OB, R
Output: S (The initial value of each entry in S is 0.)

01) for each In_i, $IR_i = \sum_{j,k} \frac{R_{ij}^k}{|f(F_{i*}^k)|}$;
02) for each Out_j, $OR_j = \sum_{i,k} R_{ij}^k$;
03) while ($\exists i IR[i] > 0$) {
04) select In_p such that $\forall i \frac{IB_p}{IR_p} \leq \frac{IB_i}{IR_i}$;
05) select Out_q such that $\forall j \frac{OB_q}{OR_q} \leq \frac{OB_j}{OR_j}$;
06) if ($\frac{IB_p}{IR_p} \leq \frac{OB_q}{OR_q}$) {
07) if ($\frac{IB_p}{IR_p} \leq 1$) {
08) for each F_{pj}^k, if ($R_{pj}^k \neq 0$ && $S_{pj}^k = 0$) $S_{pj}^k = \frac{IB_p}{IR_p}$;
09) $IB_p = 0$;
10) }
11) else {
12) for each F_{pj}^k, if ($R_{pj}^k \neq 0$ && $S_{pj}^k = 0$) $S_{pj}^k = 1$;
13) $IB_p = IB_p - IR_p$;
14) }
15) $IR_p = 0$;
16) for each Out_j, if ($OR_j \neq 0$) {
17) $OB_j = OB_j - \sum_k R_{pj}^k S_{pj}^k$;
18) $OR_j = OR_j - \sum_k R_{pj}^k$;
19) }
20) }
21) else {
22) if ($\frac{OB_q}{OR_q} \leq 1$) {
23) for each F_{iq}^k, if ($R_{iq}^k \neq 0$ && $S_{iq}^k = 0$) $S_{iq}^k = \frac{OB_q}{OR_q}$;
24) $OB_q = 0$;
25) }
26) else {
27) for each F_{iq}^k, if ($R_{iq}^k \neq 0$ && $S_{iq}^k = 0$) $S_{iq}^k = 1$;
28) $OB_q = OB_q - OR_q$;
29) }
30) $OR_q = 0$;
31) for each In_i, if ($IR_j \neq 0$) {
32) $IB_i = IB_i - \sum_k R_{iq}^k S_{iq}^k$;
33) $IR_j = IR_j - \sum_k R_{iq}^k$;
34) }
35) for all F_{iq}^k and each Out_j,
36) if ($j \neq q$ && $Out_j \in f(F_{i*}^k)$ && $OR_j \neq 0$) {
37) $OB_j = OB_j - R_{iq}^k S_{iq}^k$;
38) $OR_j = OR_j - R_{iq}^k$;
39) }
40) }
41) }

assigned satisfaction degrees, it is necessary to remove all other branches of the flows from consideration, as well as to update the available bandwidth and requested bandwidth of the remaining input ports.

To help understand the operation of the algorithm, we give an example in the following. We still consider a 3×3 switch. Each input port and output port have a unit of available bandwidth. There are five logical guaranteed performance flows as illustrated in Figure 2. R gives the bandwidth request of each flow. First, the bandwidth request at each input port and output port is summed up to initialized IR and OR. It should be noted that although OR is still the same as that in the unicast example, no entry in IR is larger than 1 due to the existence of multicast flows. In iteration 1, Out_1 has the smallest bandwidth share, and all the branches to Out_1 are assigned a satisfaction degree $\frac{2}{3}$. Note that because F_{11}^1 has been assigned a satisfaction degree, all of the branches of F_{1*}^1 should be assigned the same satisfaction degree. Thus, F_{12}^1 is assigned the satisfaction degree $\frac{2}{3}$. Vectors IB, OB, IR and OR are then updated accordingly. In iteration 2, Out_3 has the smallest bandwidth share, which is larger than 1. Therefore, all the flows to Out_3 are assigned a satisfaction degree 1. Again, since F_{2*}^2 has a branch to Out_3, all its branches should be assigned a satisfaction degree 1, and thus $S_{22}^2 = 1$. After two iterations, all the flows have been assigned satisfaction degrees, and the algorithm converges.

Input:	Iteration 1:	Iteration 2:
$R = \begin{Bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{3} \\ \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{2} & 0 & 0 \end{Bmatrix}$	$S = \begin{Bmatrix} \frac{2}{3} & \frac{2}{3} & 0 \\ 0 & 0 & 0 \\ \frac{2}{3} & 0 & 0 \\ 0 & 0 & 0 \\ \frac{2}{3} & 0 & 0 \end{Bmatrix}$	$S = \begin{Bmatrix} \frac{2}{3} & \frac{2}{3} & 0 \\ 0 & 0 & 1 \\ \frac{2}{3} & 0 & 0 \\ 0 & 1 & 1 \\ \frac{2}{3} & 0 & 0 \end{Bmatrix}$
$IB = \{1\ 1\ 1\}$	$IB = \{\frac{2}{3}\ \frac{2}{3}\ \frac{2}{3}\}$	$IB = \{\frac{1}{3}\ \frac{1}{3}\ \frac{1}{2}\}$
$OB = \{1\ 1\ 1\}$	$OB = \{0\ \frac{2}{3}\ 1\}$	$OB = \{0\ \frac{1}{3}\ \frac{1}{3}\}$
$IR = \{\frac{5}{6}\ \frac{5}{6}\ \frac{1}{2}\}$	$IR = \{\frac{1}{3}\ \frac{1}{3}\ 0\}$	$IR = \{0\ 0\ 0\}$
$OR = \{\frac{3}{2}\ \frac{5}{6}\ \frac{2}{3}\}$	$OR = \{0\ \frac{1}{3}\ \frac{2}{3}\}$	$OR = \{0\ 0\ 0\}$

Theorem 6 *The max-min fair bandwidth allocation algorithm for multicast traffic achieves max-min fairness.*

The proof is based on Theorem 5 and is similar to that of Theorem 3. The basic idea is as follows. Any unsatisfied flow can only be assigned a satisfaction degree either in step 8 or step 23, and accordingly either In_p or Out_q is a bottleneck port of the flow. Since all unsatisfied flows have bottleneck ports, the resulting satisfaction matrix is max-min fair. The detailed proof is omitted.

As we have seen, due to the introduction of multicast flows, the max-min fair bandwidth allocation algorithm for multicast traffic is much more complex than the algorithm for unicast traffic. Suppose that the maximum number of logical flows at any input port is K. The algorithm still converges in $O(M+N)$ iterations, and the complexity of each iteration is $\max\{O(\log(M+N)), O(K)\}$. Thus, the time

complexity of the algorithm is $\max\{O((M+N)\log(M+N)), O((M+N)K)\}$. As to the space complexity, the algorithms stills needs the vectors IR and OR to store the total requested bandwidth of all input ports and output ports. Thus, the space complexity remains to be $O(M+N)$.

4 Conclusions

The capability to support QoS has become an important consideration for the design of modern switches, and is usually provided by various fair scheduling algorithms. All fair scheduling algorithms require a feasible and efficient bandwidth allocation scheme as the scheduling basis. However, each input port or output port usually has only the bandwidth information of the flows traversing itself. Thus, it is essential to carefully design a globally efficient bandwidth allocation scheme. In this paper, we have considered how to fairly allocate bandwidth in packet switches. We formulated the bandwidth allocation problem, and gave the definitions of feasibility and max-min fairness. As the first step, we analyzed the simpler unicast scenarios, and presented the fair bandwidth allocation algorithm for unicast traffic, which is proved to achieve max-min fairness. We then extended the discussion to the more general multicast scenarios and presented the fair bandwidth allocation algorithm for multicast traffic. We proved that the algorithm for multicast traffic also achieves max-min fairness. In addition, we analyzed the complexity of the algorithms and presented examples to illustrate the operation of the algorithms.

References

[1] Y. Bernet et. al., "A frame work for differentiated services," *Internet Draft*, draft-ietf-diffserv-framework-01.txt, Nov. 1998.

[2] C. Metz, "IP routers: new tool for gigabit networking," *IEEE Internet Computing*, vol. 2, no. 6, pp. 14-18, Nov. 1998.

[3] S. Keshav and R. Sharma, "Issues and Trends in Router Design," *IEEE Communications Magazine*, vol. 36, no. 5, pp. 144-151, May 1998.

[4] A. Parekh and R. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single node case," *IEEE/ACM Trans. Networking*, vol. 1, no. 3, pp. 344-357, Jun. 1993.

[5] A. Demers, S. Keshav and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *ACM SIGCOMM '89*, vol. 19, no. 4, pp. 3-12, Austin, TX, Sep. 1989.

[6] H. Zhang, "WF2Q: worst-case fair weighted fair queueing," *IEEE INFOCOM '96*, pp. 120-128, San Francisco, CA, Mar. 1996.

[7] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round robin," *IEEE/ACM Trans. Networking*, vol. 4, no. 3, pp. 375-385, Jun. 1996.

[8] D. Pan and Y. Yang, "Credit based fair scheduling for packet switched networks," *IEEE INFOCOM '05*, pp. 843-854, Miami, FL, March 2005.

[9] J. Xu and R. Lipton, "On fundamental tradeoffs between delay bounds and computational complexity in packet scheduling algorithms", *ACM SIGCOMM'02*, Pittsburgh, PA, Aug. 2002.

[10] N. McKeown, A. Mekkittikul, V. Anantharam and J. Walrand, "Achieving 100% throughput in an input queued switch," *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1260-1267, 1999.

[11] M. J. Karol, M. J. Hluchyj and S. P. Morgan, "Input versus output queueing on a space-division packet switch," *IEEE Trans. Commun.*, vol. 35, pp. 1347-1356, 1987.

[12] S.-Q. Li, "Performance of a nonblocking space-division packet switch with correlated input traffic," *IEEE Trans. Commun.*, vol. 40, no. 1, pp. 97-108, Jan. 1992.

[13] J. Hopcroft and R. Karp, "An $N^{5/2}$ algorithm for maximum matching in bipartite graphs," *SIAM Journal of Computing*, vol. 2, no. 4, pp. 225-231, Dec. 1973.

[14] R. Tarjan, "Data structures and network algorithms," *CBMS-NSF Regional Conference Series in Applied Mathematics*, Dec. 1983.

[15] N. Ni and L. Bhuyan, "Fair scheduling for input buffered switches," *Cluster Computing*, vol. 6, no. 2, pp. 105-114, Hingham, MA, Apr. 2003.

[16] X. Zhang and L. Bhuyan, "Deficit round-robin scheduling for input-queued switches," *IEEE Journal on Selected Areas in Communications*, no. 4, pp. 584-594, May 2003.

[17] D. Stiliadis and A. Varma, "Providing bandwidth guarantees in an input-buffered crossbar switch," *IEEE INFOCOM '95*, pp. 960-968, Apr. 1995.

[18] T. Anderson, S. Owicki. J. Saxe and C. Thacker, "High-speed switch scheduling for local-area networks," *ACM Trans. Comput. Syst.*, vol. 11, no. 4, pp. 319-352, Nov. 1993.

[19] S. Chuang, S. Iyer and N. McKeown, "Practical algorithms for performance guarantees in buffered crossbars," *IEEE INFOCOM '05*, Miami, FL, March 2005.

[20] B. Magill, C. Rohrs and R. Stevenson, "Output-queued switch emulation by fabrics with limited memory," *IEEE Journal on Selected Areas in Communications*, vol 21, no. 4, pp. 606-615, May 2003.

[21] J. Turner, "Strong performance guarantees for asynchronous crossbar schedulers," IEEE INFOCOM '06, Barcelona, Spain, Apr. 2006.

[22] Y. Hou, H. Tzeng and S. Panwar, "A generalized max-min rate allocation policy and its distributed implementation using the ABR flow control mechanism," *IEEE INFOCOM '98*, pp. 1366-1375, Apr. 1998.

[23] A. Malla, M. El-Kadi, S. Olariu and P. Todorova, "A fair resource allocation protocol for multimedia wireless networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 14, no. 1, pp. 63-71, Jan. 2003.

[24] Y. Zhou and H. Sethu, "On achieving fairness in the joint allocation of processing and bandwidth resources: principles and algorithms," *IEEE/ACM Trans. Networking*, vol. 13, no. 5, pp. 1054 - 1067, Oct. 2005.

[25] M. Hosaagrahara and H. Sethu, "Max-min fairness in input-queued switches," *ACM SIGCOMM Student Poster Session*, August 2005, Philadelphia, PA, USA.

[26] D. Pan and Y. Yang, "FIFO based multicast scheduling algorithm for VOQ packet switches," *IEEE Trans. Computers*, vol. 54, no. 10, pp. 1283-1297, October 2005.