This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2011 proceedings.

Queue-Length Proportional and Max-Min Fair Bandwidth Allocation for Best Effort Flows

Tosmate Cheocherngngarn, Jean Andrian, Zhenyu Yang, and Deng Pan
Florida International University
Miami, FL

***Abstract*—A group of switch schedulers make packet scheduling decisions based on predefined bandwidth allocation for each flow. Allocating bandwidth for best effort flows is challenging due to lack of allocation criteria and fairness principles. In this paper, we propose sequential and parallel algorithms to allocate bandwidth for best effort flows in a switch, to achieve fairness and efficiency. The proposed algorithms use the queue length proportional allocation criterion, which allocates bandwidth to a best effort flow proportional to its queue length, giving more bandwidth to congested flows. In addition, the algorithms adopt the max-min fairness principle, which maximizes bandwidth utilization and maintains fairness among flows. We first formulate the problem based on the allocation criterion and fairness principle. Then, we present a sequential algorithm and prove that it achieves max-min fairness. To accelerate the allocation process, we propose a parallel version of the algorithm, which allows different input ports and output ports to conduct calculation in parallel, resulting in fast convergence. Finally, we present simulation data to demonstrate that the parallel algorithm is effective in reducing the convergence iterations.**

***Index Terms*—bandwidth allocation; max-min fairness; parallel processing**

I. Introduction

There exists an important group of switch scheduling algorithms that rely on pre-calculated bandwidth allocation to make scheduling decisions [1], [2]. With such algorithms, each traffic flow of the switch needs to be allocated a certain amount of bandwidth, which the scheduling algorithms will try to guarantee by arranging packet transmissions for different flows. The objective is to emulate the ideal Generalize Processor Sharing (GPS) model [3], where each flow has an logical independent transmission channel with the allocated bandwidth. The bandwidth allocation has to be feasible, which means that the total amount of allocated bandwidth at any input or output port cannot exceed its physically available bandwidth. In addition, it should be efficient, which means to fully utilize any potential transmission capacity and allocate bandwidth in a fair manner.

Switches handle two broad categories of traffic flows [4]: guaranteed performance flows with strong bandwidth and delay requirements, such as multimedia streaming traffic, and best effort flows requiring no strict delay, such as file sharing traffic. Since the former have higher priority than the latter, the switch first satisfies the bandwidth requests of the former, and uses the remaining available bandwidth for the latter. We will focus on the issue of how to efficiently and fairly allocate the leftover bandwidth to the best effort flows.

Bandwidth allocation for best effort flows faces two difficulties. In the first place, an criterion is necessary as the bandwidth allocation policy. For guaranteed performance flows, the allocation criteria are easy to find, which can be application requests or historical statistics. However, best effort flows have no such apparent information. In this paper, we propose the queue length proportional criterion, with the reasoning that if a flow has more packets waiting in the queue, then it needs more bandwidth. More importantly, a fairness principle is necessary to resolve the conflicts among flows. Each flow of the switch competes bandwidth with two different sets of flows, one departing from the same input port and one destined for the same output port. It is thus important to maintain fairness among flows. In this paper, we use max-min fairness for this purpose, which has long been a popular fairness principle for resource allocation [5], [6]. As suggested by the name, max-min achieves fairness by maximizing the minimum allocation. Hosaagrahara and Sethu proposed a bandwidth allocation algorithm based on the max-min fairness principle, which is called Fair Resource Allocation (FRA) [5]. However, FRA is specific to virtual output queueing switches, and is integrated with the Largest Credit First scheduling algorithm in [5].

It is challenging to calculate bandwidth allocation schemes following the above policies. As mentioned earlier, each flow is subject to two bandwidth constraints, at its input port and output port respectively. Since each input port has the bandwidth information of its own flows, it does not know how much bandwidth the corresponding output port will allocate to a specific flow, and the same happens to the output port. Thus, a naive bandwidth allocation scheme may be under-utilized or over-utilized. For the under-utilized case, the unused bandwidth should be allocated to make full use of available resource. For the over-utilized case, it is necessary to fairly scale down the claimed bandwidth of each flow to make the scheme feasible. In this sense, centralized global controllers are appropriate. However, centralized controllers are usually slow and cannot satisfy the time constraints of high speed switches and routers.

In this paper, we propose bandwidth allocation algorithms for best effort flows, which are independent of switch structures and scheduling algorithms. We first formulate the problem based on the queue length proportional criterion and max-min fair principle. We then present a sequential algorithm and prove that it achieves max-min fairness. To accelerate the

allocation process, we further propose a parallel version of the algorithm, which enables different input and output ports to conduct calculation in parallel, and thus achieves fast convergence. Finally, we present simulation data to demonstrate that the parallel algorithm is effective in reducing the convergence time.

The rest of the paper is organized as follows. In Section II, we discuss the background and related works. In Section III, we propose the sequential and parallel bandwidth allocation algorithms. In Section IV, we present simulation data. In Section V, we conclude the paper.

II. Background and Related Works

In this section, we provide a brief overview of switch structures and corresponding scheduling algorithms based on pre-defined bandwidth allocation.

Switches buffer packets at three possible locations: output ports, input ports, and crosspoints, and can be consequently divided into several categories. Output queued (OQ) switches have buffers only at output ports. 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. Thus, OQ switches need large speedup to achieve optimal performance, and are not practical [7]. On the other hand, since all the packets are already in output buffers, OQ switches can run various fair queueing algorithms, such as WFQ [3] and DRR [8], to provide different levels of performance guarantees. The fair queueing algorithm schedules packets to ensure the allocated bandwidth of each flow as in the ideal GPS [3] fluid model.

Input queued (IQ) switches have buffers at input ports, and eliminate speedup requirements. Input buffers are usually organized as multiple virtual output queues (VOQ) [9], with a logical separate queue for flows to a different destination, to avoid the Head of Line (HOL) blocking. Scheduling algorithms based on allocated bandwidth for IQ switches try to emulate the corresponding fair queueing algorithms for OQ switches with iterative matching. For example, iFS [10] and iDRR [11] emulate WFQ [3] and DRR [8], respectively. In addition, WPIM [12] improves PIM [13] with bandwidth enforcement and provides probabilistic bandwidth guarantees. However, those algorithms cannot duplicate the exact packet departure time to achieve perfect emulation.

Combined input-crosspoint queued (CICQ) switches and combined input-output queued (CIOQ) switches are special IQ switches with additional buffers at output ports and crosspoints, respectively. Such switches are shown to be able to perfectly emulate certain OQ switches with small speedup. Thus, various scheduling algorithms [1], [14]–[17] have been proposed to duplicate the packet departure time of existing fair queueing algorithms for OQ switches, and provide desired performance guarantees.

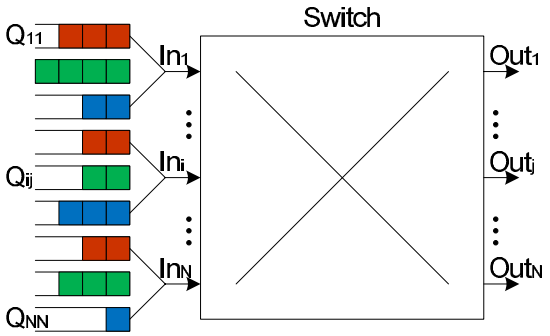

Fig. 1. Switch structure.

III. Queue-Length Proportional and Max-Min Fair Bandwidth Allocation

In this section, we formulate the switch bandwidth allocation problem, present the solution algorithms, and prove that they achieve the design goals.

A. Problem Formulation

We consider an $N \times N$ switch as shown in Figure 1, without assuming any specific switching fabrics to make the analysis general. Use In_i (Out_j) to denote the i^{th} input (j^{th} output) port, and $IB_i(t)$ ($OB_j(t)$) to denote its leftover bandwidth at time t, after satisfying requests of guaranteed-performance flows. Our algorithms work in a cycle mode, i.e. allocating bandwidth at the beginning of each new cycle. Thus we consider only the statues of all the variables at the same time, and omit the time parameter t in the variables for easy reading. We use the input queue length as the bandwidth allocation criterion, to allocate bandwidth to more congested flows. We do not consider output queues and crosspoint queues, because the former stores packets already transmitted to output ports, and the latter have limited and small capacities.

Denote the best-effort flow from In_i to Out_j as F_{ij}, and use Q_{ij} to represent its input queue length at time t. Use R_{ij} to denote the allocated bandwidth of F_{ij} at time t. Define the ratio between R_{ij} and Q_{ij} to be the bandwidth share S_{ij}, i.e.

$$S_{ij} = \frac{R_{ij}}{Q_{ij}} \tag{1}$$

which represents the bandwidth allocated to each unit of the queue length. If F_{ij} has no buffered packets at t, i.e. $Q_{ij} = 0$, set R_{ij} and S_{ij} to zero as well. Define the bandwidth share matrix S to be the $N \times N$ matrix formed by all S_{ij}, which determines the bandwidth allocation scheme.

We now define *feasibility* for bandwidth allocation. A bandwidth allocation scheme is feasible if there is no oversubscription at any input port or output port, i.e.

$$\forall i \sum_j R_{ij} \le IB_i, \forall j \sum_i R_{ij} \le OB_j \tag{2}$$

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 next define *fairness* based on the max-min fairness principle. A bandwidth allocation scheme is max-min fair if it is feasible and there is no way to increase the allocated bandwidth of any flow without reducing the allocated bandwidth of another flow with a lower bandwidth share value. Formally, a feasible bandwidth share matrix S is max-min fair, if for any feasible bandwidth share matrix S' the following condition holds

$$S'_{ij} > S_{ij} \rightarrow \exists i' \exists j' \left(S_{i'j'} \leq S_{ij} \wedge S_{i'j'} > S'_{i'j'} \right) \quad (3)$$

As can be seen, the objective of max-min fairness is twofold: increasing the bandwidth share of each flow as much as possible to fully utilize available bandwidth, and maximizing the minimum bandwidth share of all the flows to achieve fairness.

Theorem 1: A max-min fair bandwidth allocation scheme is unique.

Proof: By contradiction, assume that two bandwidth allocation matrices S and S' are both max-min fair, and $S \neq S'$. Without loss of generality, assume that S_{ij} is the smallest entry among all the ones in S that are different from their counterparts in S', i.e.

$$S_{ij} \neq S'_{ij} \wedge \forall i' \forall j' \left(S_{i'j'} \neq S'_{i'j'} \rightarrow S_{i'j'} \geq S_{ij} \right) \quad (4)$$

We look at two possible cases regarding the relationship between S_{ij} and S'_{ij}.

Case 1: $S_{ij} < S'_{ij}$. Because S is max-min fair and S' is feasible, by the definition there exist i' and j' such that $S_{i'j'} \leq S_{ij}$ and $S_{i'j'} > S'_{i'j'}$. Define $x = i'$ and $y = j'$, and we have $S_{ij} \geq S_{xy}$ and $S_{xy} > S'_{xy}$.

Case 2: $S_{ij} > S'_{ij}$. Define $x = i$ and $y = j$, and we have $S_{ij} \geq S_{xy}$ and $S_{xy} > S'_{xy}$.

Noting that in both cases $S_{xy} > S'_{xy}$, because S' is max-min fair and S is feasible, there exist x' and y' such that $S'_{x'y'} \leq S'_{xy}$ and $S'_{x'y'} > S_{x'y'}$, and therefore $S'_{xy} > S_{x'y'}$. Since $S_{x'y'} \neq S'_{x'y'}$ and S_{ij} is the smallest different entry in S, we have $S_{x'y'} \geq S_{ij}$. Combined with the previous inequality $S'_{xy} > S_{x'y'}$, we obtain $S'_{xy} > S_{ij}$, which is a contradiction with $S_{ij} > S'_{xy}$ obtained in the above two cases. ■

Next, we give the definition of bottleneck ports, which will be the base to calculate a max-min fair bandwidth allocation scheme. Given a bandwidth share matrix, a port is the bottleneck port of a flow if the flow has the highest bandwidth share 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

$$\forall j' S_{ij} \geq S_{ij'} \wedge \sum_x S_{ix} Q_{ix} = IB_i \quad (5)$$

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

$$\forall i' S_{ij} \geq S_{i'j} \wedge \sum_x S_{xj} Q_{xj} = OB_j \quad (6)$$

The following theorem shows how to calculate max-min fair bandwidth allocation.

Theorem 2: A feasible satisfaction scheme is max-min fair if and only if each flow has a bottleneck port in it.

Proof: Assume that S is a feasible bandwidth share matrix and each flow has a bottleneck port in S. Suppose S' is also feasible and $S'_{ij} > S_{ij}$. Then we know that $S_{ij} < S'_{ij}$. Since each 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 $\sum_j S_{ij} Q_{ij} = IB_i$. On the other hand, since S' is feasible, we have $\sum_j S'_{ij} Q_{ij} \leq IB_i$ and thus $\sum_x S'_{ix} Q_{ix} \leq \sum_x S_{ix} Q_{ix}$. Because $S'_{ij} > S_{ij}$, there must exist j' such that $S_{ij'} > S'_{ij'}$, otherwise we can obtain the contradiction that $\sum_x S'_{ix} Q_{ix} > \sum_x S_{ix} Q_{ix}$. 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.

■

B. Sequential Bandwidth Allocation Algorithm

We are now ready to present the bandwidth allocation algorithm. The main idea is to find the bottleneck ports for all the flows in an iteration manner, after which a max-min fair bandwidth share scheme is obtained by Theorem 2.

We define some notations before describing the algorithm. Initialize the bandwidth share of each flow to zero, i.e. $S_{ij} = 0$. Define the remaining bandwidth of a port In_i (Out_j) at the beginning of the n^{th} iteration to be the available bandwidth that has not been allocated, and denote it as $B_{i*}(n)$ ($B_{*j}(n)$), i.e.

$$B_{i*}(n) = IB_i - \sum_{S_{ix} \neq 0} S_{ix} Q_{ix} \quad (7)$$

$$B_{*j}(n) = OB_j - \sum_{S_{xj} \neq 0} S_{xj} Q_{xj} \quad (8)$$

Define the remaining queue length of a port In_i (Out_j) at the beginning of the n^{th} iteration to be the total queue length of the flows that have not been assigned bandwidth share values, and denote it as $Q_{i*}(n)$ ($Q_{*j}(n)$), i.e.

$$Q_{i*}(n) = \sum_{S_{ix} \neq 0} Q_{ix} \quad (9)$$

$$Q_{*j}(n) = \sum_{S_{xj} \neq 0} Q_{xj} \quad (10)$$

Define the bandwidth share of a port In_i (Out_j) at the beginning of the n^{th} iteration to be the ratio of the remaining bandwidth and remaining queue length, and denote it as $S_{i*}(n)$ ($S_{*j}(n)$), i.e.

$$S_{i*}(n) = \frac{B_{i*}(n)}{Q_{i*}(n)} \quad (11)$$

$$S_{*j}(n) = \frac{B_{*j}(n)}{Q_{*j}(n)} \quad (12)$$

In each iteration, the algorithm first finds the port with the smallest bandwidth share, and assigns the bandwidth share of the port to its flows without bandwidth share values. As will be formally shown later, the port is the bottleneck port of all

such flows. Processing the ports one by one guarantees that eventually each flow will have a bottleneck port.

In detail, each iteration consists of the following three steps.

1) *Calculation:* Calculate the bandwidth share of each remaining port.
2) *Comparison and Assignment:* Select the port with the smallest bandwidth share, and assign the value as the bandwidth share of all the remaining flows of the port.
3) *Update:* Remove the above selected port and the flows assigned bandwidth share values. Update the remaining bandwidth and queue length for each of the rest ports.

In the following, we show that the proposed algorithm achieves max-min fairness.

Lemma 1: The bandwidth share of a port does not decrease between iterations.

Proof: Without loss of generality, assuming that the port is an input port In_i, we show that $S_{i*}(n) \geq S_{i*}(n+1)$. The proof for an output port is similar.

First, assume that a different input port $In_{i'}$ instead of In_i is selected in the n^{th} iteration with the smallest bandwidth share. Because $In_{i'}$ and In_i have no common flows, the remaining bandwidth and queue length of In_i do not change, and thus

$$S_{i*}(n+1) = \frac{B_{i*}(n+1)}{Q_{i*}(n+1)} = \frac{B_{i*}(n)}{Q_{i*}(n)} = S_{i*}(n) \qquad (13)$$

Next, assume that an output port Out_j is selected in the n^{th} iteration with the smallest bandwidth share. Note that $S_{*j}(n) \leq S_{i*}(n)$ and that F_{ij} will be assigned the bandwidth share value of $S_{*j}(n)$, and we have

$$\begin{aligned} S_{i*}(n+1) &= \frac{B_{i*}(n+1)}{Q_{i*}(n+1)} \\ &= \frac{B_{i*}(n) - S_{*j}(n)Q_{ij}}{Q_{i*}(n) - Q_{ij}} \\ &\geq \frac{B_{i*}(n) - S_{i*}(n)Q_{ij}}{Q_{i*}(n) - Q_{ij}} \\ &= \frac{B_{i*}(n)}{Q_{i*}(n)} \\ &= S_{i*}(n) \end{aligned} \qquad (14)$$

■

Theorem 3: The bandwidth allocation algorithm achieves max-min fairness.

Proof: The key is to see that if a port assigns bandwidth share for a flow, then it is the bottleneck port of the flow.

Without loss of generality, assume that F_{ij} is assigned bandwidth share by In_i in the n^{th} iteration. Consider another flow $F_{ij'}$ of In_i. If $F_{ij'}$ is assigned bandwidth share by $Out_{j'}$ in an earlier iteration m, based on Lemma 1 we have $S_{ij'} = S_{*j}(m) \leq S_{i*}(m) \leq S_{i*}(n) = S_{ij}$. Otherwise, if $F_{ij'}$ is assigned bandwidth share by In_i in the same iteration, we know $S_{ij'} = S_{i*}(n) = S_{ij}$. Therefore, F_{ij} has the largest bandwidth share among all flows of In_i. In addition, since In_i is selected in the n^{th} iteration, all its remaining bandwidth is fully allocated, i.e. $B_{i*}(n) = S_{i*}(n)Q_{i*}(n)$. Based on Theorem 2, we know that S is max-min fair. ■

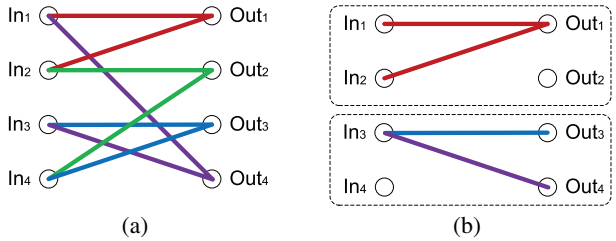

Fig. 2. Parallel processing for independent port sets. (a) Ports correlated. (b) Ports divided into independent sets.

The time complexity of the algorithm is $O(N \log N)$, because the algorithm runs $O(N)$ iterations and the sorting operation in each iteration takes $O(\log N)$. As can be seen, the algorithm finds bottleneck ports sequentially, and requires $O(N)$ iterations in both the best and worst cases. Large-size switches thus need long convergence time, which creates obstacles for high speed processing.

C. Parallel Bandwidth Allocation

To accelerate the bandwidth allocation process, we propose a parallel version of the algorithm. The design is based on the observation that an input (output) port only needs to be compared with the output (input) ports which it has a flow heading to (coming from). After some iterations, an input (output) output has flows only to (from) a small number of output (input) ports. It is thus possible to find multiple bottleneck ports in a single iteration by parallel comparison. For example, in Figure 2(a), the ports are correlated with each other. However, in Figure 2(b), it is easy to see that two port sets $\{In_1, In_2, Out_1\}$ and $\{In_3, Out_3, Out_4\}$ are independent, and that bandwidth allocation can be conducted in parallel in the two sets.

Similarly, each entry of the bandwidth share matrix S is initialized to zero. The parallel algorithm also works in iterations. One iteration of the algorithm consists of the following three steps, each of which can be conducted by different input and output ports in parallel.

1) *Calculation and Distribution:* An input (output) port In_i (Out_j) calculates its bandwidth share, and sends the result to every output (input) port that it has a flow heading to (coming from).
2) *Comparison and Assignment:* An input (output) port In_i (Out_j) compares its own bandwidth share with that of every output (input) port received in the first step. If its bandwidth share is the smallest, the value is assigned as the bandwidth share for all its remaining flows.
3) *Notification and Update:* An input (output) port In_i (Out_j) notifies every output (input) port its bandwidth share, if it has the smallest bandwidth share in the second step. The output (input) port will then know that the flow F_{ij} has been assigned a bandwidth share, and updates its remaining bandwidth and queue length. Flows already assigned with bandwidth share are removed.

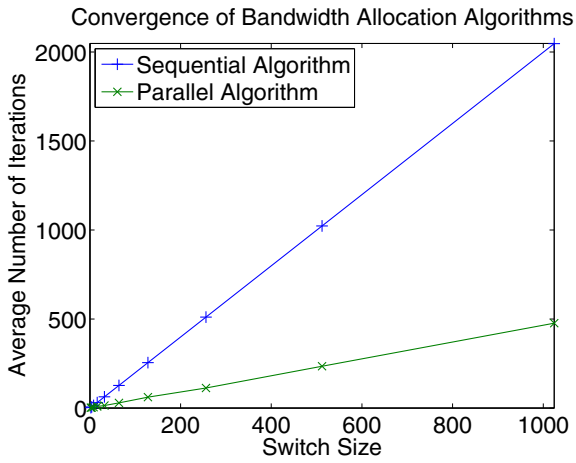

Fig. 3. Convergence iteration numbers of sequential and parallel bandwidth allocation algorithms.

We show that the parallel algorithm also achieves max-min fairness.

Theorem 4: The parallel bandwidth allocation algorithm achieves max-min fairness.

Proof: It is easy to see that Lemma 1 still applies to the parallel algorithm. Thus, with the same reasoning as in the proof of Theorem 3, we know that if a port assigns its bandwidth share to a flow, then it is the bottleneck port of the flow. Since each flow has a bottleneck port, by Theorem 2, the bandwidth allocation scheme is max-min fair. ■

IV. Simulation Results

We now present simulation results to demonstrate the effectiveness of the parallel bandwidth allocation algorithm. In the simulations, we consider switch sizes of 2^n with n from 1 to 10. We assign random values between 0 and 10000 as the queue lengths for the flows. For a specific switch size, we conduct 20 simulation runs for the sequential algorithm and parallel algorithm each, and calculate the average number of convergence iterations. Figure 3 shows the simulation results. As can be seen, although the convergence iteration numbers of both algorithms grow approximately linearly with the switch size, the result of the parallel algorithm increases much slower than that of the sequential algorithm. In detail, the average convergence iteration number of the sequential algorithm is about twice of the switch size, which is consistent with the analysis. The reason is that a switch of size N has N input ports and N output ports, and each iteration of the sequential algorithms finds one bottleneck port. On the other hand, due to parallel processing at each port, the average convergence iteration number of the parallel algorithms is about half of the switch size. We can thus make the conclusion that the parallel algorithm is effective in reducing the running time.

V. Conclusions

In this paper, we have studied bandwidth allocation for best effort flows in a switch. We propose the queue-length proportional allocation criterion, the max-min fairness principle, and bandwidth allocation algorithms that are independent of switch structures and scheduling algorithms. First, we formulate the problem, and define feasibility and fairness for bandwidth allocation. Then, we present the first version of the algorithm, which calculates the allocation bandwidth in a sequential manner. Furthermore, to accelerate the algorithm convergence, we propose a parallel version of the algorithm, by allowing different input ports and output ports to conduct calculation in parallel. We prove that both the sequential and parallel algorithms achieve the initial design objectives. Finally, we present simulation data to demonstrate that the parallel algorithm is effective in reducing the convergence iterations. Since multicast traffic is also an important component of the Internet, our future work includes extending the parallel bandwidth allocation algorithm to switches with multicast flows.

References

[1] J. Turner, "Strong performance guarantees for asynchronous crossbar schedulers," *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, pp. 1017–1028, Aug. 2009.

[2] S. min He, S. tao Sun, H. tao Guan, Q. Zheng, Y. jian Zhao, and W. Gao, "On guaranteed smooth switching for buffered crossbar switches," *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 718–731, Jun. 2008.

[3] A. Parekh and R. 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, pp. 344–357, Jun. 1993.

[4] D. Pan and Y. Yang, "Credit based fair scheduling for packet switched networks," in *IEEE INFOCOM*, Miami, FL, Mar. 2005.

[5] M. Hosaagrahara and H. Sethu, "Max-min fairness in input-queued switches," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 4, pp. 462–475, Apr. 2008.

[6] D. Pan and Y. Yang, "Max-min fair bandwidth allocation algorithms for packet switches," in *IEEE IPDPS*, Long Beach, CA, Mar. 2007.

[7] ——, "Localized independent packet scheduling for buffered crossbar switches," *IEEE Transactions on Computers*, vol. 58, no. 2, pp. 260–274, Feb. 2009.

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

[9] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input queued switch," *IEEE Transactions on Communications*, vol. 47, no. 8, pp. 1260–1267, Aug. 1999.

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

[11] X. Zhang and L. Bhuyan, "Deficit round-robin scheduling for input-queued switches," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 4, pp. 584–594, May 2003.

[12] D. Stiliadis and A. Varma, "Providing bandwidth guarantees in an input-buffered crossbar switch," in *IEEE INFOCOM '95*, Boston, MA, Apr. 1995.

[13] 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.

[14] S. Chuang, S. Iyer, and N. McKeown, "Practical algorithms for performance guarantees in buffered crossbars," in *IEEE INFOCOM*, Miami, FL, Mar. 1996.

[15] 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.

[16] D. Pan and Y. Yang, "Providing flow based performance guarantees for buffered crossbar switches," in *IEEE IPDPS*, Miami, FL, Apr. 2008.

[17] L. Mhamdi and M. Hamdi, "Output queued switch emulation by a one-cell-internally buffered crossbar switch," in *IEEE GLOBECOM*, San Francisco, CA, Dec. 2003.