International Journal of
Soft Computing And Software Engineering (JSCSE)
e-ISSN: 2251-7545
Vol.2,No.4, 2012

DOI: 10.7321/jscse.v2.n4.2

Integration of Unicast and Multicast Scheduling in Input-Queued Packet Switches with High Scalability

[1 *] Yongbo Jiang, [2]Zhiliang Qiu, [3]Jian Zhang, [4]Jun Li
[1, 2,3]*State Key Lab of ISN, Xidian University, Xi'an, China*
[4] *State Key Lab of Science and Technology on Space Microwave Laboratory, Xi'an, China*
[1*]*ybjiang@stu.xidian.edu.cn,* [2]*zlqiu@mail.xidian.edu.cn,*[4]*lij41@cast504.com*

Abstract. **This paper focuses on the scalability problems for high-speed switches, and presents an integrated scheduling algorithm that supports unicast and multicast traffic efficiently in input-queued packet switches. Considering the tradeoff balancing complexity and performance, the proposed integrated algorithm performs without iteration, and reduces the scheduling overhead to O(*N*) with a two-phase (request-grant) sequential scheduling for unicast and multicast traffic. In addition, it can be implemented in a fully distributed way, which is more suitable for high-speed switches. Simulation results show that the proposed algorithm exhibits a good performance in terms of throughput and average delay, at different traffic compositions under various traffic patterns.**

Keywords: *integrated scheduling algorithm, IQ switches, multicast, scalability*

1. Introduction

The growing number of newly emerging applications on the Internet has created an increasing need for efficient multicast traffic support. As a result, with the continuous growth of bandwidth in fiber links, the need for switches/routers that are capable of switching unicast and multicast cells at very high speeds is urgent. To the best of our knowledge, the integrated scheduling algorithms presented are, in fact, a combination of earlier unicast and multicast algorithms unified in one integrated scheduler. The input queuing structure has also been a combination of unicast queuing structure and multicast queuing structure. The widely used unicast queuing structure is the virtual output queuing (VOQ) since it can avoid the head-of-line (HoL) blocking problem, and 100% throughput could be achieved using schedulers such as iSLIP [1] and DRRM [2]. In a VOQ-based $N \times N$ switch, N queues are maintained at each input port; each queue contains packets having the same destined output. As for multicast traffic, a multicast packet can have more than one destination, known as its fan-out set. Consequently, multicast queuing structure can vary from just one multicast (first in first out) FIFO queue per input to $2^N - 1$ queues per input, where N is the number of output ports of the switch, and considerable amount of solutions based on the architecture have therefore been proposed such as [3-8]. The performance of such queuing structure was analyzed in

DOI: 10.7321/jscse.v2.n4.2

[9-11]. Depending on the above input queuing structure, integrated scheduling algorithms have been proposed. They were mainly proposed for the input queued (IQ) crossbar-fabric-based switching architecture because of its scalability, low hardware requirements, and its intrinsic multicast capabilities. Most of these algorithms were based on input VOQ for unicast traffic and one FIFO queue for multicast traffic, such as ESLIP [12] and others [13]. However, the HoL blocking problem of multicast traffic limits the throughput achievable by switches. Other algorithms [14] used VOQ for unicast traffic and a small number, k ($1 < k \leq 2^N - 1$), of FIFO queues for multicast traffic to alleviate the multicast HoL blocking. Algorithm in [15] used VOQ queuing structure for unicast and multicast traffic separately to deal with the HoL blocking.

Compare to the main constraint of limited energy of sensors in designing wireless sensor networks protocols [16], with backbone networks, high-speed switches have very short time to perform scheduling as link speed grows dramatically, and as a result iterative design and high scheduling overhead with existing integrated scheduling algorithms become the bottleneck for integrated scheduler designs since scheduling overhead scales up very quickly as the link speed and switch size increase, and the need for simple and high performance switches which support unicast and multicast traffic simultaneously is urgent. For this reason, we propose a new non-iterative integrated scheduling algorithm named Unicast and Multicast Dual Round-Robin integrated algorithm (UMDRR) which performs with only one matching cycle by a sequential scheduling for unicast and multicast traffic in a time slot rather than traditional log(N) iteration times, and reduces the multicast scheduling overhead from O(kN) to O(N), which makes it implementable at high speeds. Simulation results show UMDRR achieves a good performance under various traffic patterns.

The rest of the paper is structured as follows. In Section 2, we describe the system architecture and the proposed integrated scheduling algorithm with scheduling overhead analysis. In Section 3, we evaluate the performance of the proposed scheme by simulation. Finally, we conclude the paper in Section 4.

2. System architecture and the algorithm

The proposed integrated scheduling algorithm is targeted at $N \times N$ input-queued switches. We first describe the system architecture of the proposed algorithm and then elaborate on the details of the algorithm.

2.1. System architecture

The $N \times N$ switch system architecture of the proposed integrated scheduling algorithm is shown in Figure 1. We fix our attention on synchronous slotted switch architecture. The incoming variable-sized packets are segmented into fixed-sized packets before entering input queues and segments are put back

International Journal of
Soft Computing And Software Engineering (JSCSE)
e-ISSN: 2251-7545
Vol.2,No.4, 2012
DOI: 10.7321/jscse.v2.n4.2

together before departing from output ports. Packets transmitted by the switching fabric are assumed have equal length. The fixed-sized packet is also called cell. In this paper, we consider only the fan-out splitting discipline that cells may be delivered to output ports over several cell times. The switch fabric is a bufferless crossbar with a speedup of one, i.e., at each time slot, each input can transmit no more than one cell and each output can receive no more than one cell. However, it is obviously possible that multiple copies of the same HOL multicast cell are forwarded to different output ports during the same time slot. Both unicast traffic scheduler and multicast traffic scheduler are provided and coordinated by an integration controller. They are distributed at each input and output port. Another critical component is the traffic and resource management module (not shown in Figure 1). It monitors the traffic composition in terms of the amount of unicast and multicast traffic and provides this information to the integration controller.

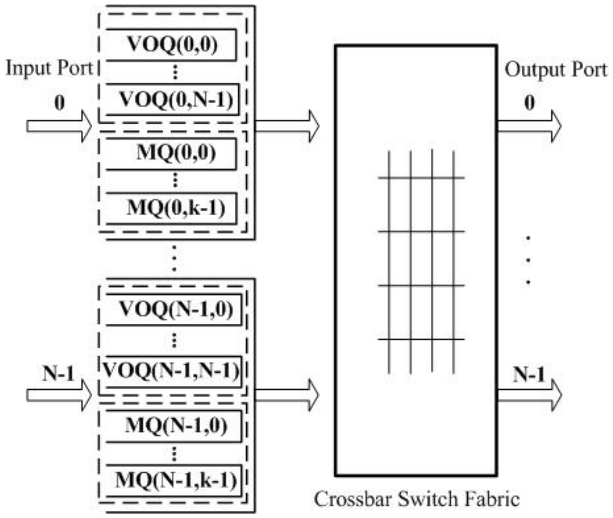

Figure 1. Input-queued switch architecture for integrated scheduling

As illustrated in Figure 1, two sets of queues are organized separately at each input port. For unicast traffic, VOQ technique is deployed and N VOQs are maintained at each input; for multicast traffic, a small number of FIFO queues are allocated at each input port. Unicast packets are assigned to the proper queues according to their destinations, while multicast flows are partitioned into the k queues according to a modulo multicast cells assignment described in [17].

We first define the terms that are used throughout the paper. Let λ be the average arrival rates, equal to the input load, μ be the output load, and the unicast and multicast output loads be denoted as μ_u and μ_m, then the following relations hold:

$$\mu=\mu_u+\mu_m \ =\lambda P_u+\lambda E[F]P_m \quad (1)$$

where P_u (P_m) represents the probability that an arrival packet is a unicast (multicast) cell, and $E[F]$ denotes

International Journal of
Soft Computing And Software Engineering (JSCSE)
e-ISSN: 2251-7545
Vol.2,No.4, 2012
DOI: 10.7321/jscse.v2.n4.2

the average number of destinations of multicast cells. The total length of unicast traffic queues and multicast traffic queues are denoted by L_u and L_m respectively, and then the total length of mixed traffic queues L is derived as

$$L = L_u + L_m = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} L_{i,j}^{u} + \sum_{i=0}^{N-1}\sum_{j=0}^{k-1} L_{i,j}^{m} E[F] \quad (2)$$

where $L_{i,j}^{u}$ ($L_{i,j}^{m}$) represents the length of the jth unicast (multicast) queue allocated at input i.

2.2. Integrated scheduling algorithm

By employing an existing unicast scheduling algorithm [2] and a new multicast scheduling algorithm, we propose a sequential integrated scheduling algorithm that supports unicast and multicast traffic efficiently. Unicast scheduling and multicast scheduling are coordinated together with a specific priority in a time slot. The scheduling procedure works as follows.

Both unicast traffic scheduler and multicast traffic scheduler are distributed at each input and output port. Each input scheduler maintains three priority pointers: a unicast pointer, a multicast primary pointer and a multicast secondary pointer. Primary pointers are designed to provide fairness among k multicast queues at each input, while secondary pointers are used to alleviate the HoL blocking and thus guarantee high performance. Each output scheduler maintains two priority pointers: a unicast pointer and a multicast pointer. All output multicast pointers point to the same input, and increase by one at each multicast time slot. This pointer update rule [4] is fundamental to guarantee that the scheduler can run in a fully distributed way. We denote the input preferred by all output multicast pointers as primary input, and the others as secondary inputs. A detailed description of the integrated algorithm follows, including three phases:

Phase 1: At the beginning of each time slot, determine the scheduling priority with the following probabilities:

$$\Pr(\text{multicast slot}) = L_m / L,\ \Pr(\text{unicast slot}) = 1 - L_m / L.$$

A time slot identified to schedule unicast (multicast) traffic first is called a unicast (multicast) slot.

Phase 2: Serve the prioritized traffic. This process includes the following two steps:

Step 1: Request. In a unicast slot, each input sends an output unicast request corresponding to the first nonempty VOQ in a fixed round-robin order, starting from the current position of the unicast pointer. The unicast pointer of the input scheduler is incremented to one location beyond the selected output if, and only if, the request is granted in step 2. In a multicast slot, the primary input (each secondary input) sends multicast requests to all destined output ports corresponding to the first nonempty multicast queue in a fixed round-robin order, starting from the current position of the multicast primary pointer (secondary pointer). The primary pointer (each secondary pointer) of the primary input (each secondary input) is incremented to one location beyond the selected queue.

DOI: 10.7321/jscse.v2.n4.2

Step 2: Grant. In a unicast slot, if an output receives one or more requests, it chooses the one that appears next in a fixed round-robin schedule starting from the current position of the unicast pointer. The output notifies each requesting input whether or not its request was granted. The unicast pointer of the output scheduler is incremented to one location beyond the granted input. In a multicast slot, if an output receives one or more requests, it chooses the one that appears next in a fixed round-robin schedule starting from the current position of the multicast pointer. The output notifies each requesting input whether or not its request was granted. The multicast pointer of the output scheduler is incremented by one.

Phase 3: Serve the nonprioritized traffic with the remaining resources. This process includes the following two steps:

Step 1: Request. In a unicast slot, each unmatched input sends multicast requests to all destined output ports corresponding to the first nonempty multicast queue in a fixed round-robin order, starting from the current position of the multicast secondary pointer. In a multicast slot, each unmatched input sends an output unicast request corresponding to the first nonempty VOQ in a fixed round-robin order, starting from the current position of the unicast pointer.

Step 2: Grant. In a unicast (multicast) slot, if an unmatched output receives one or more requests, it chooses the one that appears next in a fixed round-robin schedule starting from the current position of the multicast (unicast) pointer. The output notifies each requesting input whether or not its request was granted.

2.3. Scheduling overhead analysis

The major problem with existing iterative scheduling algorithms is that the scheduling overhead scales up very quickly as the link speed and switch size increase, which limits the scalability in high-speed switches having very short time to perform scheduling. This study overcomes the limitations and proposes a new integrated scheduling algorithm with reduced communication overhead.

We first define scheduling overhead as the information exchanged at an input port in one matching cycle. As we can see from Table 1, in contrast to existing three-phase (request-grant-accept) integrated scheduling algorithms with log(N) iteration times, UMDRR performs with only one matching cycle; moreover, it has one less operational step (request-grant), and less information exchange between inputs and outputs. Specifically, UMDRR reduces the unicast scheduling overhead from O(N) to O(log$N$) by selecting one of the N VOQs for requesting, and multicast scheduling overhead from O(kN) to O(N) by selecting one of the k multicast HoL cells for requesting, where k ($1 < k \leq 2^N - 1$) represents the number of multicast queues maintained at each input.

International Journal of
Soft Computing And Software Engineering (JSCSE)
e-ISSN: 2251-7545
Vol.2,No.4, 2012
DOI: 10.7321/jscse.v2.n4.2

Table 1. Scheduling overhead comparisons

Algorithms	Conflict resolutions		Control messages (bits)			Convergence iteration times
	Input	Output	Request	Grant	Accept	
ESLIP	round-robin	round-robin	$2N$	N	$\log(N)$	$\log(N)$
*f*SCIA	max_service	max_weight	$N+kN(2+\log N)$	N	$\log(N)$	$\log(N)$
UMDRR	round-robin	round-robin	$\log(N)+N$	$1+N$	-	-

3. Performance evaluation and analysis

In this section, we show some simulation results derived from OPNET Modeler [18]. To evaluate the performance of the proposed integrated scheduling algorithm, we consider several different traffic conditions and compare the algorithm with ESLIP [12] and *f*SCIA [14] for a 16×16 switch. The ESLIP algorithm is chosen for comparison because it is practical, and is being deployed on commercial switching products, while *f*SCIA using a number of queues for multicast traffic as well, and exhibits a good performance. The simulated switch is assumed to have sufficient buffers at the input. We consider the mixture of unicast and multicast traffic in this study, and algorithms perform with a single iteration for a fair comparison.

3.1. Traffic model

Two traffic scenarios are used to evaluate system performance. For Bernoulli (uncorrelated) arrival, in each time slot, the probability that a new packet arrives is independent of any other time slot; for Bursty (correlated) arrival, cells are generated by a two-state Markov process which has busy and idle states. The process stays in each state for a random number of timeslots following a geometrical distribution with expectancy of E[B] and E[I]. Bursty traffic is provided to each input with a mean burst size of 16 cells. It is assumed that the destination port of a unicast cell is generated according to a uniform and nonuniform distribution, while that the fan-out set of a multicast cell is chosen uniformly at random among all the 2^N-1 possible fan-out sets. The nonuniform unicast traffic is defined by using an unbalanced probability, ω ($0\le\omega\le1$). For input port i and output port j, the traffic load, $\mu_{i,j}$, is followed by the following relation:

$$\mu_{i,j}=\begin{cases}\mu_u\left(\omega+\dfrac{1-\omega}{N}\right)+\dfrac{\mu_m}{N}, & \text{if } i=j,\\ \mu_u\dfrac{1-\omega}{N}+\dfrac{\mu_m}{N}, & \text{otherwise.}\end{cases} \tag{3}$$

International Journal of
Soft Computing And Software Engineering (JSCSE)
e-ISSN: 2251-7545
Vol.2,No.4, 2012
DOI: 10.7321/jscse.v2.n4.2

Note that when $\omega = 0$, the load is uniform over all outputs and when $\omega = 1$, the unicast traffic load is totally unbalanced.

3.2. Performance under uniform traffic

Figure 2 and Figure 3 show the average delays against the output load for ESLIP, *f*SCIA and UMDRR under uniform traffic. With given traffic composition (P_m=0.1), we can observe that as output load increase, UMDRR is very effective in reducing the average delay, and performs reasonably well. It has lower latency comparing to ESLIP and *f*SCIA with a single iteration. We also show the simulation results of ESLIP and *f*SCIA with log(N) iteration times for a reference. We can see that at the expense of high complexity, ESLIP and *f*SCIA with 4 iterations achieve lower cell delays, however, the difference is even not significant at high load for ESLIP. Note that the reason why ESLIP and *f*SCIA with one iteration perform not very well is that they experience an inefficiency matching where some of the grants can be wasted because of input contention, and as a result some outputs can be idle for the scheduling decision in the timeslot.

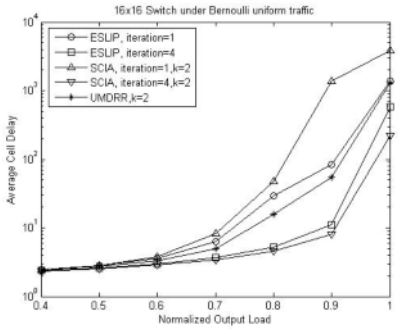

Figure 2. Delay as a function of output load with uniform Bernoulli traffic.

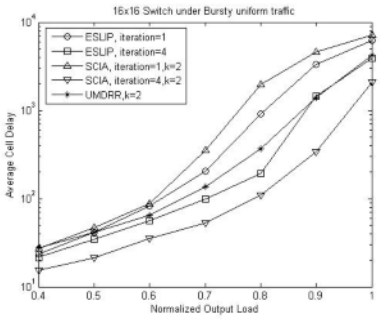

Figure 3. Delay as a function of output load with uniform Bursty traffic.

International Journal of
Soft Computing And Software Engineering (JSCSE)
e-ISSN: 2251-7545
Vol.2,No.4, 2012
DOI: 10.7321/jscse.v2.n4.2

Figure 4 and Figure 5 illustrate the throughput as a function of multicast fraction, and compare the maximum achievable throughput of UMDRR with two other integrated algorithms for different iteration times under uniform traffic. It is shown that irrespective of the arrival traffic pattern, UMDRR always achieves higher throughput than ESLIP and *f*SCIA with a single iteration , and also exhibits better throughput performance than ESLIP with 4 iterations when the percentage of multicast traffic beyond 0.5, which demonstrates that UMDRR can allocate the switch resources to unicast and multicast traffic more equitably, and can alleviate the multicast HoL blocking more efficiently than ESLIP. On the other hand, owing to the multicast HoL blocking alleviation and high complexity (both high time complexity and high scheduling overhead), *f*SCIA with 4 iterations can achieve a set of maximal matches between input and output ports and therefore provides a good throughput performance.

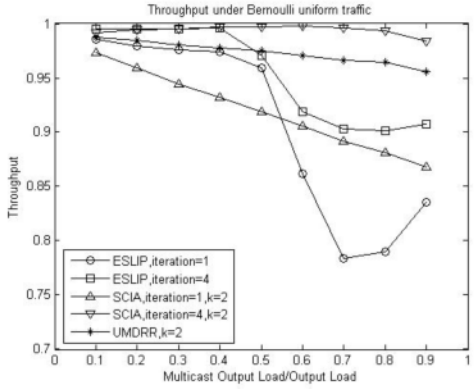

Figure 4. Throughput as a function of multicast fraction with uniform Bernoulli traffic.

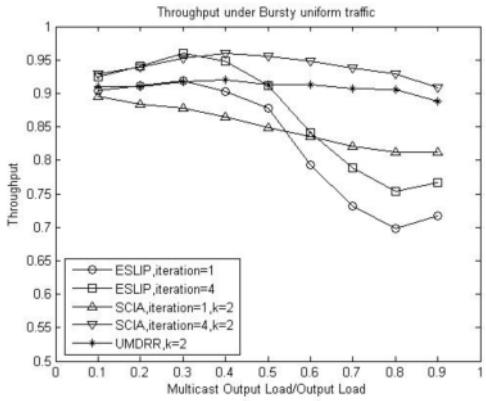

Figure 5. Throughput as a function of multicast fraction with uniform Bursty traffic.

International Journal of
Soft Computing And Software Engineering (JSCSE)
e-ISSN: 2251-7545
Vol.2,No.4, 2012
DOI: 10.7321/jscse.v2.n4.2

3.3. Performance under nonuniform traffic

Figure 6 and Figure 7 demonstrate the average delays as a function of output load for ESLIP, *f*SCIA and UMDRR under nonuniform traffic(ω = 0). With given traffic composition (P_m=0.1), we can see that the delay performance of the three schemes is affected by the nonuniform traffic severely as the output load grows, and UMDRR still has smaller cell delays than ESLIP and *f*SCIA with one iteration.

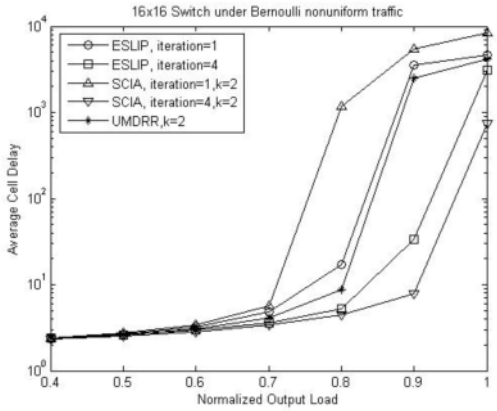

Figure 6. Delay as a function of output load with nonuniform Bernoulli traffic.

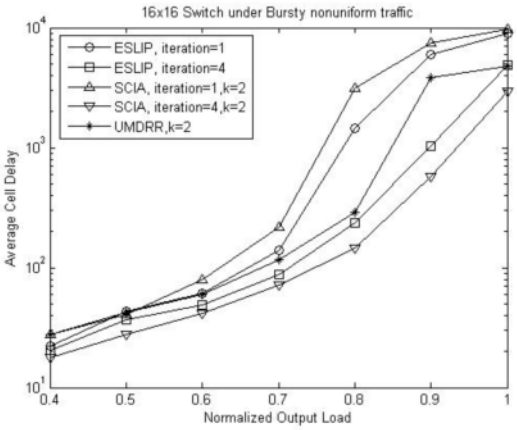

Figure 7. Delay as a function of output load with nonuniform Bursty traffic

3.4. Performance improvement by increasing *k*

The performance of delay and throughput for the proposed integrated algorithm can be increased efficiently through increasing the number of multicast queues. It is of great importance that the multicast scheduling overhead of the proposed algorithm remains O(*N*) when *k* grows, which is different from the

International Journal of
Soft Computing And Software Engineering (JSCSE)
e-ISSN: 2251-7545
Vol.2,No.4, 2012
DOI: 10.7321/jscse.v2.n4.2

other existing algorithms with O(*kN*).

Figure 8 and Figure 9 present the improvement of delay and throughput performance introduced by increasing the number of multicast queues for Bernoulli traffic and Bursty traffic, respectively. For the given traffic composition(P_m=0.1), we can observe that with a novel update rule for multicast requesting pointers at each input, the delay and throughput of UMDRR improve efficiently as *k* increases. The intuition behind this is that as *k* increases, the update rule of the requesting pointer allows more new cells to participate in scheduling during the next time slot, and as a result reduces the output contention and consequently alleviates the HOL blocking problem. From Figure 9 we can see that the improvement of the throughput is not significant when the multicast traffic fraction is small, and as the proportion of multicast traffic grows, the improvement is obvious. We can also observe that a high throughput can be achieved when *k* grows to 8, which corresponds to the conclusion in [9] that a small number of multicast queues (less than 10) are enough to obtain a high switch performance.

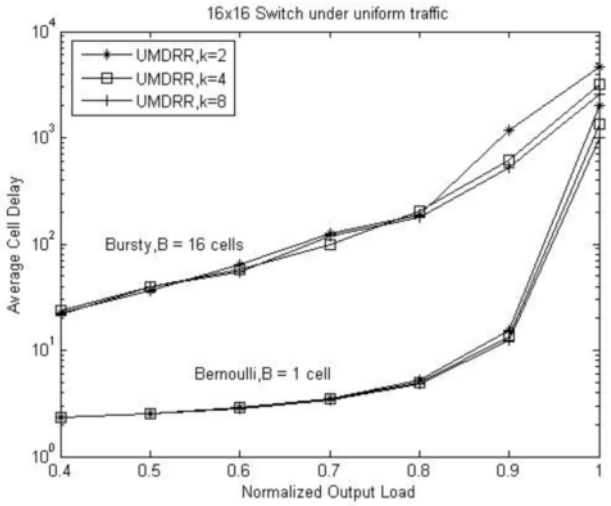

Figure 8. Delay as a function of output load with increasing *k* under uniform traffic

4. Conclusion

In this paper, we present a scalable, fully distributed, fair, and simple integrated scheduling algorithm that supports unicast and multicast traffic simultaneously. From a practical point of view, the proposed algorithm reduces the multicast scheduling overhead from traditional O(*kN*) to O(*N*) , whereas provides a good performance in terms of delay and throughput. Simulation results show that the algorithm is more suitable for large capacity, high-speed switches/routers that have very short time to perform scheduling under various traffic patterns. In addition, several issues are not discussed in this paper including the improvement of switching performance with pure unbalanced unicast traffic and the analytical analysis of

DOI: 10.7321/jscse.v2.n4.2

the proposed integrated scheduling algorithm, and it will be discussed in our further study.

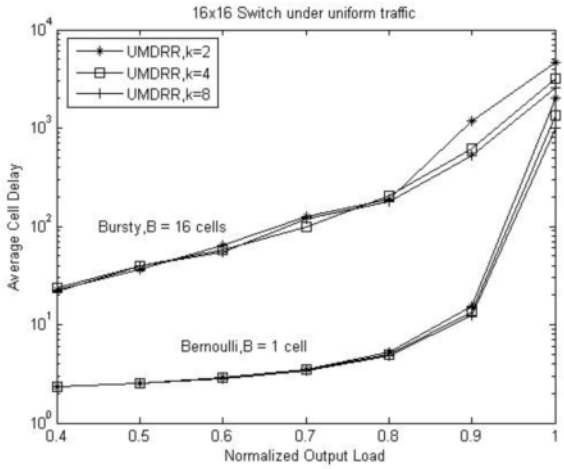

Figure 9. Throughput as a function of output load with increasing k under uniform traffic

Acknowledgments

This research was supported by the Program for Changjiang Scholars and Innovative Research Team in University (IRT0852) and National High Technology Research and Development Program of China (2008AA01A332, 2009AA01A335).

References

[1] McKeown, N., "The iSLIP scheduling algorithm for input-queued switches", *IEEE/ACM Trans. Netw.*, vol. 7, no. 2, pp. 188-201, April 1999.

[2] Li, Y., Panwar, S., and Chao, H. J., "On the performance of a dual round-robin switch", In Proc. IEEE INFOCOM, pp. 1688-1697, 2001.

[3] Prabhakar, B., McKeown, N. and Ahuja, R., "Multicast scheduling for input-queued switches", *IEEE J. Sel. Areas Commun.*, vol. 15, no. 5, pp. 855-866, June 1997.

[4] Bianco, A. and Scicchitano, A., "Multicast support in multi-chip centralized schedulers in Input Queued switches", *Computer Networks*, vol. 53, no.7, pp. 1040-1049, May 2009.

[5] Yu, H., "A Novel Round-Robin Based Multicast Scheduling Algorithm for 100 Gigabit Ethernet Switches", In Proc. IEEE INFOCOM, pp. 1-2, March,2010.

DOI: 10.7321/jscse.v2.n4.2

[6] Yu, H., Ruepp, S. and Berger, M.S., "Enhanced first-in-first-out-based round-robin multicast scheduling algorithm for input-queued switches", *IET Commun.*, vol. 5, no. 8, pp. 1163-1171, May 2011.

[7] Gupta, S. and Aziz, A., "Multicast scheduling for switches with multiple input-queues", In Proc. 10th Symposium on High Performance Interconnects, pp. 28-33, 2002.

[8] Bianco, A., Giaccone, P., Leonardi, E., Neri, F. and Piglione, C., "On the number of input queues to efficiently support multicast traffic in input queued switches", In Proc. Workshop on High Performance Switching and Routing, pp. 111-116, 2003.

[9] Song, M., Zhu, W., Francini, A. and Alam, M., "Performance analysis of large multicast switches with multicast virtual output queues", *Computer Communications*, vol. 28, no. 2, pp. 189-198, February 2005.

[10] Song, M. and Zhu, W., "Throughput analysis for multicast switches with multiple input queues", *IEEE Commun. Lett.*, vol. 8, no. 7, pp. 479-481, July 2004.

[11] Zhu, W., and Song, M., "Performance analysis of large multicast packet switches with multiple input queues and gathered traffic", *Computer Communications*, vol. 33, no. 7, pp. 803-815, May 2010.

[12] McKeown, N., "A fast switched backplane for a gigabit switched router", *Business Comm. Rev.*, vol. 27, no.12, pp. 1-17, 1997.

[13] Andrews, M., Khanna, S., and Kumaran, K., "Integrated scheduling of unicast and multicast traffic in an input-queued switch", In Proc. IEEE INFOCOM, pp. 1144-1151, March, 1999.

[14] Zhu, W., and Song, M., "Integration of unicast and multicast scheduling in input- queued packet switches", *Computer Networks*, vol. 50, no.5, pp. 667-687, April 2006.

[15] Chin, K. , "A new integrated unicast/multicast scheduler for input-queued Switches", In Proc. 8th Australasian Symposium on Parallel and Distributed Computing, pp. 13-20, 2010.

[16] Ladan D., Hossein S., and Mohammadreza S., "Simulated Annealing algorithm for Data Aggregation Trees in Wireless Sensor Networks", International Journal of Soft Computing and Software Engineering, vol. 1, no.1, December 2011.

[17] Mhamdi, L., "On the integration of unicast and multicast cell scheduling in buffered crossbar switches", *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no.6, pp. 818-830, June 2009.

[18] OPNET Modeler, Available at:http://www.opnet.com/solutions/networr_rd/modeler.html.