On Multicast Scheduling and Routing in Multistage Clos Networks

Bin Tang

Department of Computer Science
Stony Brook University
Stony Brook, NY 11794
bintang@cs.sunysb.edu

Abstract

Multicast communication, which involves transmitting information from one node to multiple nodes, is a vital operation in both broadband integrated services digital networks (BISDN) and scalable parallel computers. Among different multicast switching networks, much work has been centered around the crossbar switches due to the simplicity of implementation. However, crossbars have the maximum number of crosspoints and implementation complexity among all switching fabrics. Multistage interconnect networks (MINs) such as Clos networks have attracted many attentions in recent years due to the non-blocking property as the crossbars, while at the same time possessing many advantages over crossbars. However, no multicast scheduling and routing algorithm have been applied to Clos network. This paper explores this aspect – we study some existing multicast scheduling and routing algorithms and apply them to Clos networks. Moreover, we propose two packet scheduling strategies to improve the system performance in terms of switch blocking probability and throughput, as shown by simulations.

1 Introduction

Multicast communication, which involves transmitting information from one node to multiple nodes, is a vital operation in both broadband integrated services digital networks (BISDN) and scalable parallel computers. It has become an important functionality of current switching network due to its many real-time applications such as video conference calls and video-on-demand services. On the other hand, Asynchronous Transfer Mode (ATM) is the most widely studies and implemented form of high speed network.

To implement fast multicast switching ATM networks, three important issues have to be solved. First, there are usually more than one packets destined for the same output at the same time and thus results in *output port contention*. This problem can be solved by placing queues at input ports, internal switching fabric, or output ports of the switches. Second, efficient scheduling algorithm is needed to coordinate conflicting packets with the same destination output ports. Third, efficient routing algorithm is needed so that the switching fabric can transmits the packets to their destined output ports following the routing control algorithm.

In this paper, we discuss the scheduling and routing aspects of the switching networks. How to switch multicast traffic from input ports to output ports have a direct effect on many network characteristics including bandwidth utilization, packet delay and Quality of Service (QoS) provisioning.

In literatures, various kinds of multicast switching systems have been studied. They include one-shot scheme [5], cell splitting scheme [4], residue concentration scheme [17], window-based scheme [18] etc. However, most of the research work of multicast scheduling and routing design are centered around crossbar switches. Clos-type networks have been extensively studied for both one-to-one communication and multicast communication in the literatures [6, 15, 23, 22, 12]. To the best of our knowledge, no multicast scheduling and routing algorithm has been implemented on Clos networks, which have many advantages over crossbars. In this paper, we will discuss the algorithm design and analysis in this regard. Figure 1 is the schematic of the multicasting switch we proposed.

The rest of the paper is organized as follows. In section 2, we give an overview of the related work. Section 3 discusses the architectures of the current ATM switching networks, emphasizing crossbar switches and Clos networks. Section 4 presents an existing multicast scheduling algorithm and discuss the reason why we choose it to apply to the Clos networks. Section 5 shows our proposed algorithms and the simulation results, along with analysis and discussion. Section 6 concludes the paper and proposes some open research problems.

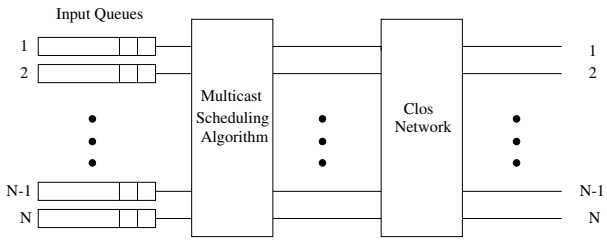


Figure 1: Schematic of the multicasting switch

2 Related Work

A growing part of traffic on the Internet is multicast, with users distributing a wide variety of audio and video materials. This dramatic change in the use of the Internet has been facilitated by the MBONE [16, 8, 7]. It is inevitable that the volume of multicast traffic will continue to grow. There are many work done to make high speed network switches be able to handle multicast [9, 11, 20, 14, 2, 23].

For each multicasting cell, the set of its destination output ports is called $fanout$. The *cardinality* of a fanout refers to the number of the destination output ports in the set. Based on how to satisfy the fanout, there are two multicasting service disciplines. One is *full multicast* or *one-shot multicast*, in which all copies of the same packet must be switched at the same time slot. This can easily result in *head of line (HOL)* blocking [13]. The other discipline is called *partial multicast* or *fanout splitting* discipline. In this case, copies generated by the same input packet may gain access to output ports over any number of slots until all of the copies are transmitted. This discipline is work conserving and has better performance than full multicast, which is simpler to implement. Again, there is a performance-implementation tradeoff between these two disciplines.

The main obstacle of multicast scheduling algorithms is output port contention, which happens when there are more than one cells destined to the same output port at the same time. So it is important to schedule the cells before they get switched by the switching fabric. This idea of prescheduling is the main trend of the current multicasting algorithm design.

It has been observed that many multicasting algorithms have been implemented in hardware. For example, Chao et al. proposed two of the most recent multicasting switches — *Multicast output buffered ATM switch* (MOBAS) and Abacus switch [2, 3]. In both cases, fairly complicated architectures and components are involved to accommodate the functionality of multicasting. We observe that while it can be tempting to throw hardware to improve switch performance, that strategy maybe counter-productive in practice, since more complicated solutions can be very difficult to implement at high speed. Furthermore, specific hardware does not reflect and help developing general scheduling algorithms. Based on above considerations, we only discuss the multicasting algorithms which are independent of the architectures underneath.

Chen et al. [5] proposed the cyclic priority (CP) Reservation scheduling algorithm, which utilize a token to indicate the priority of the input ports and the availability of the output ports. However, the technique of CP reservation is somewhat inefficient due to its determinacy, which means there is no adaptability to particular incoming traffic. A neural-network-based contention resolution mechanism was proposed in the same paper to improve the performance of above CP reservation algorithm. It basically constructs an energy function in which the most stable state of the neural network corresponds to the contention-free traffic combination and maximized throughput. Although the NN scheme provides certain improvement, it seems that the delay-performance advantage is not great enough to overrule the simplicity of implementation of the CC scheme. McKeown et al. [17] proposed a series of new algorithms — *concentration*, *TATRA* and *weight based algorithm* to schedule the multicasting cells for input ports. Because they are easy to implement and produce good simulation results, they have drawn many attentions since their publication. Several proposals based on them are published [21, 10]. We also target one of the algorithms — TATRA.

3 Overview of ATM Switching Networks

Multicasting is required in implementing multimedia applications like audio and video conferencing. These real-time applications require a constant flow of data because large jitter can have a negative impact on the quality of the video and audio. New technologies like Asynchronous Transfer Mode (ATM) are likely to have low jitter due to the use of optical fibers as the transport media. However, the real advantage of ATM is its fixed-sized packet — cell. Cell is very helpful to build fast and highly scalable switches for the following reasons:

- it is easier to build hardware to process packets when their sizes are already known.
- many switching elements (SEs) can do the same thing in parallel when all packets are of the same length. This parallelism greatly improves the scalability of switch designs.

Throughout the survey we will use "cell" and "packet" interchangeably since most of the work done falls into fixed-length packet multicast switching.

In this paper, we use the following three metrics to evaluate the performance of a switching network:

- *Throughput*, which is the average number of packets transmitted by the switch per time slot, or it can be normalized by the number of packets arrived at the switch in unit time.
- *Cell delay*, which is the time lapse from the point the packet enters the switch to the point it leaves.
- *Blocking probability*, which is the ratio between the number of packets blocked in the switch and the number of all the packets arriving at the input ports.

Besides these three metrics, fairness, implementation complexity and scalability should also be taken into consideration when comparing different architectures and scheduling algorithms.

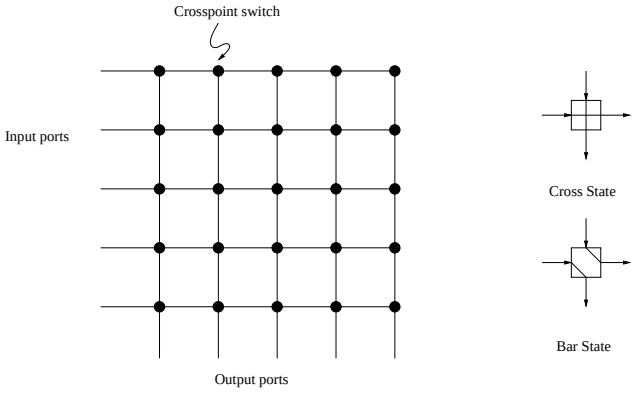


Figure 2: Schematic of a 4 x 4 Crossbar network

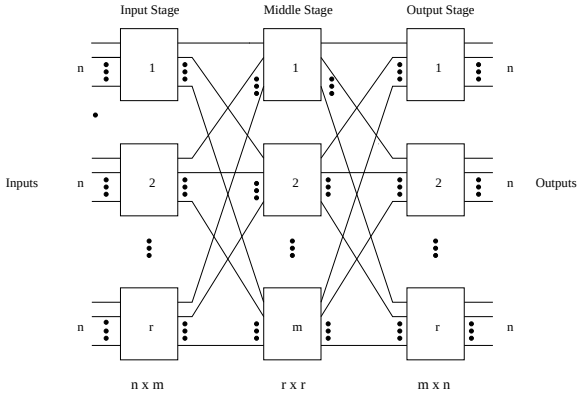


Figure 3: A general schematic of a v(m,n,r) network

3.1 Crossbar Switches

In general, multicast ATM switches can be build from three basics fabrics: crossbar network with Knockout switch as its representative, self-routing network with Banyan network as its representative, and Clos network. The following three subsections will cover them one by one.

Crossbar switches are conceptually very simple — every input port is connected to every output port and forms a grid of switching elements (SEs). An $N \times N$ crossbar switch has N^2 crosspoints. Each crosspoint has two possible states: cross and bar. A connection between input port i and output port j is established by setting the (i, j)th crosspoint switch to the bar state and other crosspoints along the route remain the cross state. A 4×4 crossbar switch is shown in Figure 2.

There are several advantages of crossbars. First, they are simple structures and easy to implement. Second, they have natural multicast property. Third, they are intrinsically non-blocking (i.e. a path is always available to connect an idle input port to an idle output port). As the result, the early unicast and multicast switches adopt crossbars as the switching fabric. Knockout switch is such an example. However, high hardware complexity(N^2) makes crossbars only be suitable for fast switches with moderate size.

3.2 Multistage Interconnection Networks (MINs)

MINs connect input ports to output ports through a number of switch stages, where each switch could be a crossbar network or a MIN. The aim of MINs is to avoid the high hardware complexity of crossbar networks and achieve the nonblocking capability at the same time. Two typical MINs: self-routing switches and Clos networks are to be examined in detail:

3.2.1 Self-Routing Switches

The general principle behind self-routing fabrics is that each packet has enough information in its header, so each switch stage can make the routing decision based on that. This can be done by having the input port add an extra header to the packet before it arrives at the fabric, and then having the output port remove this header before the packet leaves the switching network. This header is called a *self-routing header*.

The advantages of self-routing switches are:

1. There is no central control mechanism needed for routing cells.
2. Several cells on different paths can be processed simultaneously.
3. The modular and recursive structure makes it possible to build large-scale switches.

3.2.2 Clos Networks

Clos networks are one kind of MINs with only three stages [6]. A general schematic of a $v(m, n, r)$ Clos network is shown in Figure 3. A three-stage Clos network with N input ports and N output ports has r switch modules of size $n \times m$ in input stage, m switch modules of size $r \times r$ in middle stage, and r switch modules of size $m \times n$ in output stage.

Clos networks provide more than one path from one input port to one output port and thus there are less internal conflicts inside Clos network. Clos network provides more reliability than crossbars based on the same reason.

Hardware Complexity. In general, the network cost of such multistage networks is measured by the number of crosspoints in the networks. An $a \times b$ switch module has ab crosspoints. The total number of crosspoints of $v(m, m, r)$ network equals

$$rnm + mr^2 + rmn = m(2nr + r^2) = m(2N + r^2)$$

For fixed N and r, the network cost is proportional to the number of middle stage switches m. The interest of

research is to reduce the number of middle switches to yield lower cost, while not causing blocking in the network. Yang et al. [22] present results that lead to the currently best known explicit constructions of nonblocking broadcast switching Clos networks. It employs a routing control strategy and shows for multicasting networks to be nonblocking, there is an improvement of the minimum number of middle switches from $O(nr)$ to $O(nlogr/loglogr)$. We will examine it in more detail later.

4 TATRA Multicast Scheduling Algorithms

For each input cell destined to multiple output ports, each of the copy is defined as an *output cell*. When output contention occurs, the set of all output cells that lose contention is called *residue*. McKeown et al. propose the selection of where to place the residue uniquely defines the scheduling algorithm. In order to achieve high throughput, the residue should be concentrated into as few inputs as possible so that it is more likely new cells can be forwarded as HOL cells and be switched earlier. The detailed description of the algorithm is in [17].

5 Experiment Simulation

Crossbars have such advantages as nonblocking and modular and easy to implement. However, it also has a very serious drawback. The implementation complexity of an N-port crossbar switch increases with N^2, making crossbars impractical for systems with a very large number of ports. It is the case that nowadays, the majority of high-performance switches and routers have only a relatively small number of ports (usually between 8 and 32). However, in the future, when switches and routers with large aggregate bandwidth become reality, the crossbar switches will no longer be suitable.

The Clos network structure demonstrates that strictly nonblocking multistage switching networks could be designed at a considerable savings in switching costs as compared to other alternatives [6]. We also believe Clos network will come to play a significant role for the switching technology due to its scalability.

To the best of our knowledge, most of the multicast scheduling algorithms are based on crossbars. We hereby apply above TATRA algorithm to Clos network and investigate the performance of TATRA under the condition of weak Clos networks, which means the number of middle stages of Clos network is small. We will then present two local strategies to improve the blocking probability and throughput of the Clos network. We will make use of the linear routing control algorithm proposed by Yang et al. [22].

5.1 A Routing Control Algorithm in Clos Networks

Figure 3 is a schematic of Clos network. The most interesting property of the Clos network is its path diversity. Since the network has exactly one link between every two switch modules in its consecutive stages, for a Clos network with m middle switches, there are m routes from any input a to any output b, one through each middle stage switch. We also assume that every switch in the network has multicast capability, that is, each idle input link of a switch can be simultaneously connected to any subset of idle output links of the switch.

Since output stage switches in a $v(m,n,r)$ network have multicast capability, a multicast connection can be described in terms of connections between an input port and its corresponding output stage switches. The number of output stage switches in a multicast connection is referred to as the *fanout* of the multicast connection. Let O denote the set of all output stage switches. Based on the structure of the $v(m,n,r)$ network, we have $O = \{1, 2, ..., r\}$.

First of all, we need the following definitions to characterize the $v(m,n,r)$ network:

To characterize a multicast connection, we define the *connection request* I_i for each input port $i \in \{1, \ldots, rn\}$ as the subset of the switch modules in the output stage to which i is to be connected in the connection. It is obvious that $I_i \subseteq O = \{1, 2, \ldots, r\}$.

To characterize the connection state between the input stage and middle stage in a $v(m,n,r)$ network, for any input port $i \in \{1, \ldots, rn\}$, we refer to the set of middle switches with currently unused links to the input switch associated with input port i as the *available middle switches*.

To characterize the state of the m switch modules in the middle stage of a three-stage switching network, let $M_j \subseteq O = \{1, \ldots, r\}$, $j = \{1, 2, ..., m\}$ denote the subset of the switch modules in the output stage to which the middle stage switch j is providing connection paths from the input ports. We will refer to the sets M_j as the *destination sets* of the middle switches.

The *network controller* of a Clos-type network executes a network routing control algorithm to establish connection path between input and output ports. Since the network has exactly one link between every two switch modules in its consecutive stages, finding connection path between input and output ports is equivalent to finding the appropriate middle switch.

As discussed in [22], a routing control strategy plays an important role in reducing the nonuniformity of multicast connections and, in turn, reducing the blocking probability of the $v(m,n,r)$ multicast network. [24] proposed seven different routing control strategies, among which the *smallest relative cardinality strategy* leads to the lowest blocking probability for a $v(m,n,r)$ network with a much smaller m than the nonblocking condition.

We are going to utilize this strategy in our algorithm and here we describe it again:

Smallest Relative Cardinality Strategy: Choose a middle switch whose destination set has the smallest cardinality with respect to the connection request (that is, first intersect the connection request with the destination sets and then choose the smallest cardinality).

[24] shows a generic algorithm for routing in a $v(m, n, r)$ multicast network as follows:

1. If there are no available middle switches for the current connection request, then exit without making the connection; otherwise go to Step 2.
2. Choose a nonfull middle switch (i.e., a middle switch with at least one idle output link) among the available middle switches for the connection request according to smallest relative cardinality strategy. If no such middle switch exists, then exit without making the connection.
3. Realize as large as possible portion of the connection request in the middle switch chosen in Step 2.
4. Update the connection request by discarding the portion that is satisfied by the middle switch chosen in Step 2.
5. If the connection request is nonempty, go to Step 1.

Above routing control algorithm is effective in reducing the blocking probability of the multicast network. It can provide a factor of two to three performance improvement over random routing.

5.2 Simulations and Analysis

We now study TATRA algorithm by showing its limitation on weak Clos networks and our improvement upon it. We will take a look at the traffic models first.

5.2.1 Traffic Models

In order to compare our strategies with TATRA, we adopt the traffic models in [17]. Both the incoming traffic and the scheduling decision operate in a time-slotted manner. Specific details of the traffic models are as following:

- Bernoulli Traffic

  For each input port, the probability that there is a cell arriving in each time slot is identical and independent of any other time slot. This probability represents the offered load or the arrival rate of each input port of the switch.

- Bursty Traffic

  In the bursty traffic model, cells are generated by Markov chain which has two states — *busy* and *idle*. The chain remains in these two states for a geometrically distributed number of cell times, with expected duration $E[B]$ and $E[I]$, respectively. During the *busy* state, cells destined for the same output ports arrive continuously in consecutive time slots. We set $E[B]$ to be 16 cell times. The packet arrival rate is given by: $p = E[B]/(E[B] + E[I])$.

Our simulated switch is assumed to have infinite buffers at the inputs. The simulation runs for typically 1 million cell times unless the switch becomes unstable (i.e. the cell delays are so huge that the switch is unable to sustain the offered load).

5.2.2 Local Strategies and Simulations

For comparison, we have implemented TATRA algorithm with both crossbar and Clos network as switching fabric. We simulate on 8×8 switches in both cases ($v(m, 4, 2)$ for Clos network). In the case of Clos networks, we use the routing control algorithm explained before. It shows when the number of middle stages in Clos network is large enough ($v(6, 4, 2)$ in this case), it will be nonblocking as crossbars. Figure 4 shows they have exactly the same average cell delay as a function of offered load, under both Bernoulli and bursty traffic. However, with the same traffic load, the cell delay under Bernoulli traffic is much smaller than under bursty traffic. This is can be explained that the burstiness of traffic will cause more "traffic jam" and thus more cell delay.

We are more interested in the simulation results under weak (blocking) Clos networks. TATRA algorithm was originally developed for crossbars, which are nonblocking. When this algorithm is applied to blocking Clos networks, cell loss occurs. Our goal is to let the cells go through the blocking Clos networks without incurring any cell loss. We modify the TATRA algorithm to discharge the bottom-most row and present two local strategies under weak Clos networks:

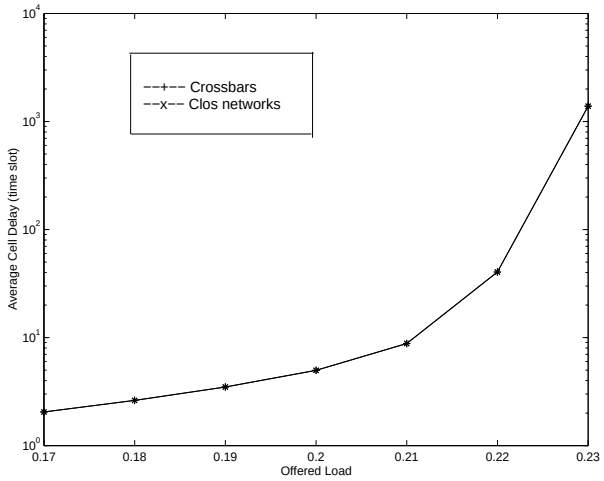
- Small Fanout First Strategy (SFFS):

  Before discharging the multicast connection requests at bottom-most row, we rearrange them in the ascending order according to the cardinality of the fanout. Then we call the routing control algorithm to realize as many multicast connection requests as we can.

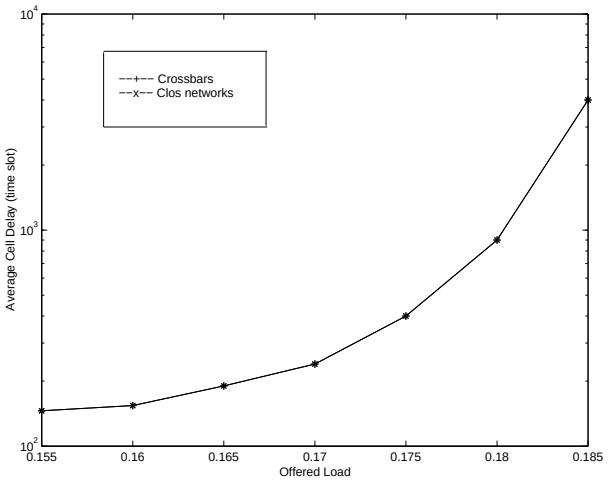
  Recall blocking probability is the ratio between the number of cells blocked in the switch and the number of all the cells arriving at the switch. We compare blocking probabilities of different size of Clos networks ($v(m, 4, 4)$ and $v(m, 8, 4)$) under Bernoulli traffic (Figure 5). Under SFFS, the blocking probabilities are decreased in all the cases.

  Figure 5 also indicates by increasing the middle stage number of the Clos networks, blocking probabilities are decreased dramatically, which can be seen when $v(4, 4, 4)$ is changed to $v(5, 4, 4)$ in Figure 5(a) and $v(8, 8, 4)$ is changed to $v(9, 8, 4)$ in Figure 5(b).

  The comparison under bursty traffic also shows the same trend as Bernoulli traffic (Figure 6).

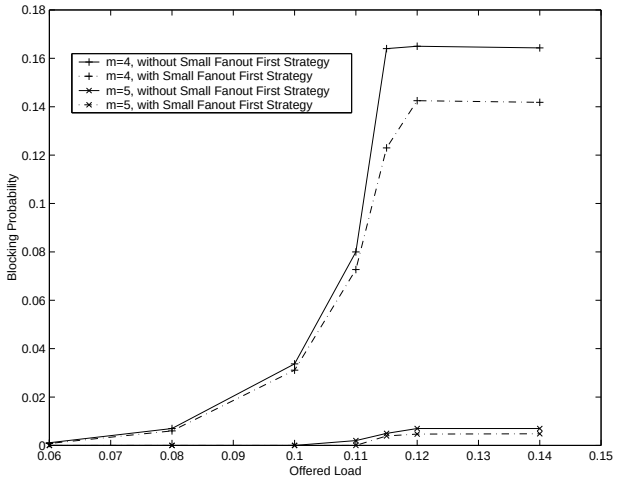


(a) *Bernoulli*

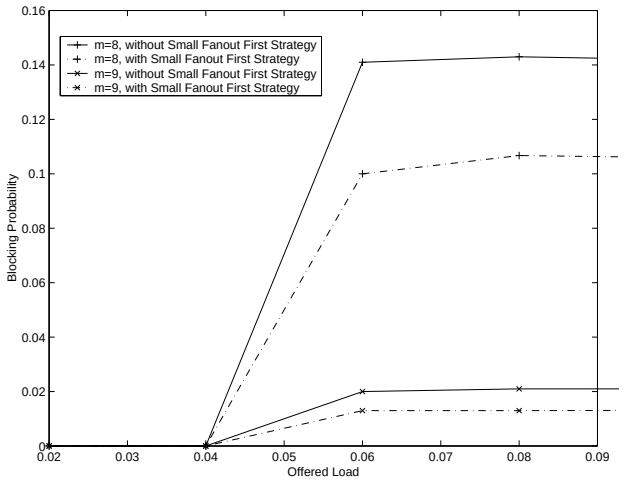


(b) *Bursty*

Figure 4: Graph of average cell delay (in number of cell times) as a function of offered load



(a) $v(m, 4, 4)$



(b) $v(m, 8, 4)$

Figure 5: Blocking probabilities of Clos network under Bernoulli traffic

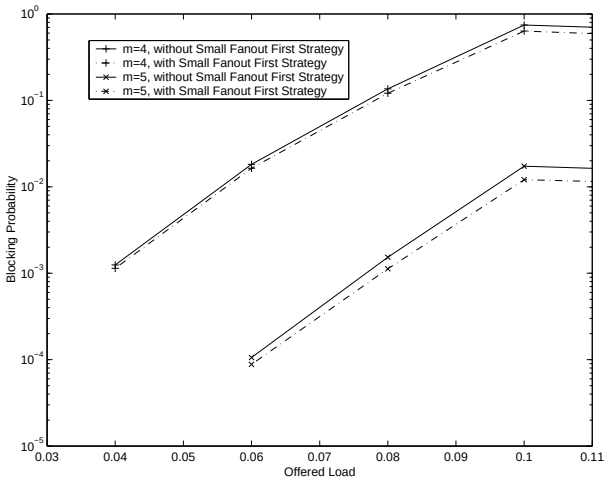


Figure 6: Blocking probabilities of 16 x 16 Clos network under Bursty traffic

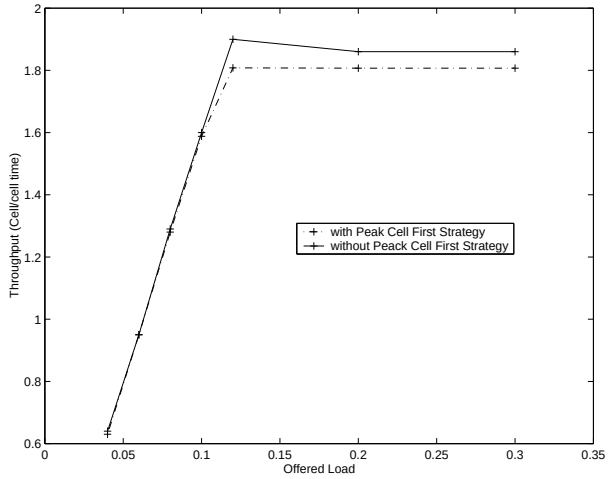


Figure 7: Throughput of v(4,4,4) Clos network

- Peak Cell First Strategy (PCFS):

  Before discharging the multicast connection requests at bottom-most row, we rearrange them in the order so that multicast connection requests with peak cells come first. Then we call the routing control algorithm to realize as many multicast connection requests as we can.

  We apply PCFS strategy to $v(4, 4, 4)$ Clos network. The simulation of throughput as a function of offered load is given in Figure 7. At above saturated load, the PCFS strategy can improve the throughput performance.

6 Conclusion and Future Considerations

Current multicast scheduling algorithms are developed on crossbars, not on MINs such as Clos networks, even though they hold great promise in the future switching networks. We therefore apply some existing algorithm to Clos networks and propose some scheduling strategies to improve the system performance. It shows our strategies can improve both the throughput and blocking probabilities. Our intermediate next step is to develop an algorithm making weak Clos network nonblocking (or almost nonblocking) without affecting throughput and cell delays seriously.

The above two local strategies are limited. They only consider rearranging the bottom-most row of the connection request. It is also assumed the blocked cells will be discarded. However, since there are infinite input queues in our case, it is not reasonable to discard the blocked cells. In the future, we will come up with a robust algorithm to improve the blocking probability under the condition of weak Clos network, while at the same time, the throughput and cell delay will not deteriorate too much. The basic idea will be *recirculation*, which means if the local strategies cannot help, the blocked cells have to be kept in the input queues and rescheduled in the next time slot. Of course, this will cause more cell delay.

In our future research, we will also consider:

- Multiple queues per input port

  Even though the VOQ implementation in multicast is impractical due to the exponential growth of the number of queues for each input, it is still feasible to use multiple queues per input for multicast scheduling. [10] has addressed this issue. However these algorithms are all heuristic simulations without any theoretical justification. Further analytical model need to be developed for better justification.

- Randomization Algorithms

  Radomization algorithms have been applied to the design of input-queued switch schedulers [19, 1]. However, as far as we know, randomization algorithms have not yet been applied to multicast switching networks.

- Clos network as the switching fabric

  For most of the multicast algorithms discussed above, they have one point in common: they are all centered around the selection policy to overcome output port contention. In the case of blocking Clos network, we are more concerned about the internal blocking. Related algorithms need to be designed to cope with internal blocking.

7 Acknowledgments

I am very grateful to Professor Yuanyuan Yang for her suggestions and guidances when I was working on this project.

References

[1] Scheduling algorithms for input-queued switches: Randomized techniques and experimental evaluation, 2000.

[2] H. J. Chao and B. S. Choe. Design and analysis of a large scale multicast output buffered atm switch. *IEEE/ACM Transactions on Networking*, 3(2), 1995.

[3] H. J. Chao, B. S. Choe, J. S. Park, and N. Uzun. Design and implementation of abacus switch: a scalable multicast atm switch. *IEEE Journal on Selected Areas in Communications*, 15(5):830–843, 1997.

[4] X. Chen and J. F. Hayes. Call scheduling in multicasting packet swiching. In *Proc. IEEE ICC*, pages 895–899, 1992.

[5] X. Chen, J. F. Hayes, and M. K. Mehmet-Ali. Performance comparison of two input access methods for a multicast switch. *IEEE Transactions on Communications*, 42(5), 1994.

[6] C. Clos. A study of non-blocking switching networks. *Bell Syst. Tech. J.*, 28:406–424, 1953.

[7] S. E. Deering and D. R. Cherition. Multicast routing in datagram internetworks and extended lans. *ACM Transactions on Computer Systems*, 8(2), pages = 85-110, year = 1990,).

[8] H. Eriksson. Mbone: the multicast backbone. *Communications of the ACM*, 37.

[9] M. H. Guo and R. S. Chang. Multicast atm switches: Survey and performance evaluation. *ACM Computer Communication Review*, 28:98–131, 1998.

[10] S. Gupta and A. Aziz. Multicast scheduling for switches with multiple input-queues. In *Proc. of Hot Interconnects X*, 2002.

[11] A. Huang. Starlite: a wideband digital switch, 1984.

[12] F. K. Hwang and A. Jajszczyk. On nonblocking multiconnection networks. *IEEE Transactions on Communications*, 34:1038–1041, 1986.

[13] M. Karol and M. Hluchyj. Queueing in high performance packet switching. *IEEE Journal on Selected Areas in Communications*, 6(9):1587–97, 1988.

[14] T. T. Lee. Nonblocking copy networks for multicast packet switching. *IEEE Journal on Selected Areas in Communications*, pages 1455–1467, 1988.

[15] G. M. Masson and B. W. Jordan. Generalized multistage connection networks. *Networks*, 2:191–209, 1972.

[16] V. Paxson. Growth trends in wide-area tcp connections. 8(4), pages = 8-17, year = 1994,).

[17] B. Prabhakar, N. McKeown, and R. Ahuja. Multicast scheduling for input-queued switches. *IEEE Journal on Selected Areas in Communications*, 15(5):855–66, 1997.

[18] K. J. Schultz and P. G. Gulak. Multicast contention resolution with single-cycle windowing using content addressable fifo's. *IEEE/ACM Transactions on Networking*, 4:731–742, 1996.

[19] D. Shah, P. Giaccone, and B. Prabhakar. Efficient randomized algorithms for input-queued switch scheduling. *IEEE Micro*, 22(1):19–25, 2000.

[20] J. S. Turner. Design of a broadcast switching network. In *Proc. IEEE Infocom*, pages 667–675, 1986.

[21] Luca Veltri. Maximum throughput in multicast queued packet switches. In *Proc. of ICC*, volume 7, pages 2033–2037, 2001.

[22] Y. Yang and G. M. Masson. Nonblocking broadcast switching networks. *IEEE Transactions on Computers*, 40(9):1005–1015, 1991.

[23] Y. Yang and G. M. Masson. Broadcast ring sandwich networks. *IEEE Transactions on Computers*, 44:1169–1180, 1995.

[24] Y. Yang and J. Wang. On blocking probability of multicast networks. *IEEE Transactions on Communications*, 46(7):957–968, 1998.