Scalable Central-stage Buffered Clos-network Packet Switches with QoS

Feng Wang and Mounir Hamdi
Department of Computer Science
The Hong Kong University of Science and Technology
{fwang, hamdi}@cs.ust.hk

Abstract - In our previous work [1], we proposed a scalable packet switch architecture based on the Central-stage Buffered Clos-network (CBC). We analyzed the memory requirements for the CBC to emulate an output-queued (OQ) switch and left the corresponding scheduling algorithms unexplored. In this paper[1], we set out to find a practical algorithm to schedule packets in order for the CBC to emulate an OQ switch supporting Quality of Service (QoS). We observe that the CBC surprisingly extend the well-known Birkhoff-von Neumann input-queued switches [15], making it able to scale to large switches with many input/output ports. In particular, as far as we know, the most efficient scheduling algorithm for a Birkhoff-von Neumann switch has a time complexity of $O(N^{4.5})$, where N is the number of switch ports. We show in this paper that we can reduce it to $O(N^{2.25})$ by employing a multi-stage multi-layer switch implementation.

I. INTRODUCTION

With the constantly growing Internet traffic and the development of broadband access technologies, such as DSL, cable modem and gigabit Ethernet, the future broadband packet switches/routers should be able to support a *large* number of connection ports for at least the following two reasons [2]. a) The number of Internet access points is still rapidly increasing; and b) The development of optical transmission technologies makes huge number of communication channels available. We cannot afford to scale current single-stage crossbar based routers simply because the building costs and complexity of the switching hardware and scheduling algorithms usually depend on the *square* of the number of switch ports.

As an alternative approach, we try to construct a large switching system out of many smaller switches. However, we cannot simply interconnect these smaller switches in a straightforward manner. A simple hierarchical network is not efficient for building large switching systems. It will also create several stages of queuing delay and may result in unpredictable performance.

In our previous work [1], we proposed a scalable packet switch based on the *Central-stage Buffered Clos-network,* which we denoted as the CBC architecture. We figured out the memory requirements for the CBC to emulate an output-queued (OQ) switch, for both first-come-first-served OQ switches and OQ switches with a general queuing discipline, e.g. weighted fair queuing (WFQ) [9], for various QoS guarantees.

In the meantime, in industry Cisco has unveiled its next generation routing system CRS-1 [13] very recently, which is also based on a multi-stage multi-layer *Benes* architecture. But their technical specifications remain proprietary that we do not know much about the details of their design.

The basic intuition behind the Central-stage Buffered Clos-network (CBC) for packet switches is based on the ideas of single-stage buffering [8] and multi-stage multi-layer switching. Different from many implementations of combined-input-output-queued (CIOQ) switches, the CBC switch employs a single stage of buffering, which makes each memory efficiently shared among all inputs/outputs. Multi-stage multi-layer switching is a natural way to scale switches and may decompose complex scheduling algorithms into many smaller layers that will have less complexity, since they may deal with traffic only locally if properly designed.

The memory requirements for the CBC to emulate an OQ switch, as shown in [1], are very interesting and illuminating. In this paper, we propose a scheduling algorithm for the CBC to emulate an OQ switch based on the Birkhoff-von Neumann matrix decomposition method.

Although the Birkhoff-von Neumann matrix decomposition has been employed in the input-queued switches [15] and the load-balanced version [19], they are mainly used in the single layer crossbar switching. The most efficient matrix decomposition method available has a time complexity of $O(N^{4.5})$, which was suggested by Birkhoff [14] and Von Neumann [18], where N is the number of switch ports. The main contribution of this paper is that by employing a multi-stage multi-layer switching, we can distribute the traffic matrix decomposition computation into many smaller switches, thus reducing the scheduling complexity to $O(N^{2.25})$. Not all multi-layer switches can achieve this since the Birkhoff-von Neumann matrix decomposition relies on the *non-overbooking* conditions of switch inputs/outputs. For a multi-layer switch, the overall non-overbooking does not necessarily mean all the smaller switches inside also have the non-overbooking conditions. We show that in the CBC architecture, by carefully dispatching packets into the central stage buffers, all the smaller switches (layers) inside hold the non-overbooking conditions as well.

[1] This research is supported under RGC HKUST6200/02E.

The rest of this paper is organized as following: In section II, we introduce the Central-stage Buffered Clos-network for packet switches and show the memory requirements for the CBC to emulate an FCFS-OQ switch. In section III, we first analyze the memory requirements for the CBC to emulate an OQ switch with a general queuing discipline and then propose a scheduling algorithm. In section IV, we identify the main advantages of the scheduling algorithm and talk about some practical considerations. Then we will have a conclusion.

I. THE CENTRAL-STAGE BUFFERED CLOS-NETWORK FOR PACKET SWITCHES

A. Some definitions

Before proceeding into the introduction of the CBC architecture, we would like to make clear some definitions used in our presentation. We adopt the fixed-length packet concept and call the packets or segmented packets *'cells'* afterwards. This is common practice in high performance routers [7].

Time Slot - Refers to the time taken to transmit or receive a fixed length cell at the line rate *R*. We assume that in every time slot, there is at most one cell arriving at each input port and at most one cell departing from each output port.

Output Queued (OQ) switch - A switch in which arriving cells are placed immediately in queues at the output, where they contend with cells destined for the same output waiting for their turns to depart. The departure order may be simple *First-Come-First-Served* (FCFS), in which case we call it an FCFS-OQ switch. Other service disciplines, such as WFQ [9], GPS [10], and DRR [11] are widely used to provide QoS guarantees. One characteristic of an OQ switch is that the buffer memories must be able to accept (write) *N* cells and read one cell per time slot, where *N* is the number of ports. Hence, the memory must operate at *N* + 1 times the line rate.

PIFO queues - A *Push-In-First-Out* queue ordering is defined according to the following rules. (1) Arriving cells are placed at (or, pushed in) an arbitrary location in the queue; (2) The relative order of cells in the queue does not change once cells are in the queue, i.e., cells in the queue cannot switch places; and (3) Cells may be selected to depart from the queue only from the head. PIFO queues are quite general and can be used to implement various QoS scheduling disciplines such as WFQ, GPS and strict priorities.

B. The Central-stage Buffered Clos-network

We proposed a new model (CBC) for multi-stage multi-layer packet switches. As shown in figure 1, the architecture resembles the traditional Clos-network *except* that we split the central-stage switches into two identical copies, with a memory linking each output/input pair. As shown in the figure, we call the left part to the memories the first switching phase and the right part the second switching phase.

Like the Clos-network, we also use parameters (n,m,k) to describe the symmetric CBC architecture. There are *k* input modules (IM), each of which is an $n \times m$ switch. There are two copies of *m* central modules (CM), each of which is a $k \times k$ switch. The number of independent memories between the two copies of CM is mk in total. There are *k* output modules (OM), each of which is an $m \times n$ switch. Each pair of IM and CM (CM and OM) is connected by *one and only one* link. There are totally *N* input ports and *N* output ports for the whole CBC architecture, where $N = n \times k$.

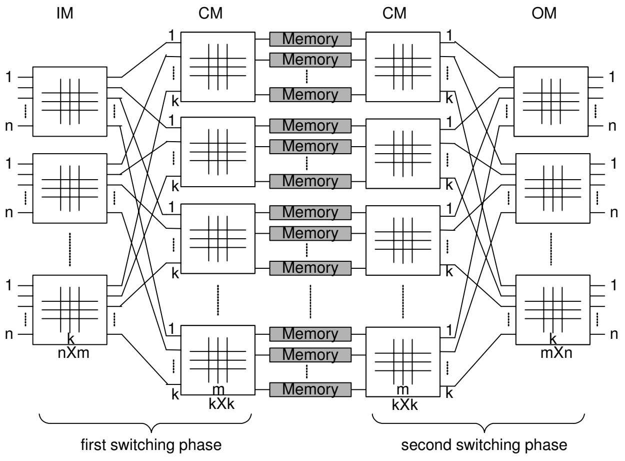

Figure 1: Basic architecture of the Central-stage Buffered Clos-network

The reason why we use two copies of the central stage is that we want to make the memories to be fully shared among all the input and output ports. Otherwise, if we only use one copy of the central stage, e.g. missing all the CM in the first switching phase, we cannot make one memory be accessed by all the input ports, which makes inefficient use of all the memories.

C. Emulating an FCFS-OQ switch

We state the memory requirements for the CBC to emulate an FCFS-OQ switch in the following theorem. In an FCFS-OQ switch, the departure time is assigned to every cell according to their arriving time and never changes in future.

Theorem 1: The number of the central modules $m \geq (2n-1)(2-1/k)$ suffices to guarantee every incoming cell a compatible memory to be written in without input contention, thus making the CBC capable of emulating an FCFS-OQ switch without internal speedup [1].

Corollary: The CBC can emulate an FCFS-OQ switch with the number of central stage modules $m \geq 2n-1$, i.e. a traditional Clos-network, using a speedup of $2-1/k$.

This corollary deserves more attention here. If we regard the strictly non-blocking Clos-network [3] ($m \geq 2n-1$) as a black box and make a *combined-input-output-queued* (CIOQ) switch out of it, then according to [6] we can see that, for emulating an FCFS-OQ switch, the minimum speedup needed is $2-1/N$, i.e. $2-1/nk$, which is slightly *larger* than what we get here, which is $2-1/k$. We achieve this number by using a multi-stage multi-layer switch architecture and single-stage buffering strategy, which is very different from CIOQ's two stages of buffering and one stage of switching.

III. The CBC architecture to provide QoS

In order for the CBC to emulate an OQ switch supporting Quality of Service (QoS), we use the generalization of the weighted-fair-queuing (WFQ) discipline known as a PIFO queue.

First, we point out that it is impossible for the CBC to emulate a *precise* PIFO queue switch without internal memory speedup. We note that when a cell comes at a PIFO queue, it cannot figure out which memories are *compatible* since it may change its departure time afterwards due to the PIFO discipline. For example, a cell C comes with departure time t initially; so, it will conflict with all other cells bearing the departure time t since the memory cannot support two reads in time t. Because C's departure time may become $t+1$, $t+2$… in the future, it may also conflict with all other cells bearing the departure time $t+1$, $t+2$ and so on. The departure conflict can be *infinite*. That is to say, we do not have a deterministic scheme for the CBC to emulate strict PIFO-OQ packet switches.

Using the same compromise as that in [8], we relax the QoS requirements slightly. We just guarantee to switch out cells within no more than a constant delay according to their departure times in the PIFO queue. That is to say, if a cell C should depart from the shadow strict PIFO-OQ switch at time t, C is guaranteed to depart from the CBC in no later than time $t+T$, where T is a constant determined by the configuration of the CBC architecture.

With this compromise, we are able to design an elegant scheduling algorithm that has a low time complexity. We introduce the basic idea of our scheme here. The first switching phase is responsible for dispatching arriving cells into the central memories. In the second switching phase, we wait for N time slots and then lock all the cells required to be out within the N time slots in the PIFO queue. We guarantee to switch all of them out in the next N time slots. In the meanwhile we have locked another batch of cells for the next N time slots to switch. Our scheme works in a pipeline manner. Readers will find that some cells may be delayed, and we will prove the delay is bounded.

We shall modify the CBC structure slightly. We add small buffers in the inputs of all the OMs, allowing the cells that are switched out of the CM to be buffered temporarily in the OMs, as shown in figure 2. Each buffer runs at the line speed and holds up to N cells. A cell traverses the CBC like this. First, it is assigned a departure time according to the PIFO discipline and written into one of the central memories via the first switching phase (meanwhile, all the cells already in the central memories will change their departure times if affected). Then, it will be switched out of the CM of the second switching phase, arriving at the buffers of the OM. Finally, it will be switched out of the OM, thus out of the CBC.

A. Scheduling in the first switching phase

Just like emulating an FCFS-OQ switch, to provide QoS support, the first switching phase of the CBC should carefully put cells into the central buffers according to the PIFO queuing discipline. We first describe three rules for the first switching phase to observe when dispatching cells into the central memories. We will prove later that, with these three rules observed by the first switching phase, the second switching phase of the CBC is capable of switching out all the cells in the central memories and every cell's delay is bounded compared with that of a shadow PIFO-OQ switch.

Here are the rules for the first switching phase to observe to put an arriving cell C with a departure time t into a memory:

1. The memory is not about to be written by another arriving cell in the same time slot;
2. The CM into which C is about to be written is not occupied by another arriving cell that is from the same IM as C in the same time slot; and
3. The memory has no cells destined for the same output as C bearing the departure time within the time interval $(t-N,t)$ and $(t,t+N)$.

Now we further explain the three rules. The first rule is obvious since the memory does not support two simultaneous writes. The second rule is due to the structure property of the Clos-network. Since there is one and only one link between one IM and one CM, the link can only be used for one cell to pass in one time slot. The third rule is not so obvious, and we will explain later that it is very essential for the second switching phase to switch all the cells in memories out in time.

B. Scheduling in the second switching phase

We prove that the locked batch of cells can be switched out of the CM in N time slots and every OM can switch out one batch of cells in N time slots. Then we design an algorithm for the second switching phase to schedule cells based on the Birkhoff-von Neumann matrix decomposition.

Edge coloring problem

We point out one property of the central memories first. That is, one memory can contain at most one cell which is destined for a specific output port of the CBC and has the departure time within any arbitrary time slot interval of length N. To make it clear, the property is formulated in the following:

For any given output port p and time τ, for one memory,

# {cell: cell is destined for p and bearing departure time $t\in[\tau,\tau+N)$ } ≤ 1.

This is because of the third rule stated above by the first switching phase. Even if cells may change their departure times due to later arrivals, this property still holds.

Now, consider the CM of the second switching phase of the CBC, as shown in the left dashed rectangle in Figure 2. Its inputs are linked with memories and its outputs are linked with the OMs of the CBC. We represent the scheduling problem using a bipartite graph with 2k vertices, as shown in Figure 3. In every N time slots, we will get a request graph. An edge connecting input i to output j represents a cell that needs to be transferred from memory i to output j. From the property of the memories stated above, we can see that in every consecutive N time slots each memory can contain at most one cell destined for one output port of the CBC, so each memory can have at most N cells to send in this interval of N time slots. That is to say the maximum degree of the input i is N. Consider the degree of the output j now. Every edge of pair (i, j) stands for that memory i contains cells destined for the OM connected by j. Every output module has at most n output ports and every memory contains at most one cell for one output port, so there are at most n edges between every pair (i, j). Therefore, the number of edges connecting output j is no more than nk, i.e. N.

Scheduling in each CM can be transformed into an edge-coloring problem in the request graph, in which the maximum degree of vertices is N. From [12], we can prove that the request graph is N-colorable, which can be transformed into a series of switch configurations to schedule the batch of cells out of the CM in N time slots.

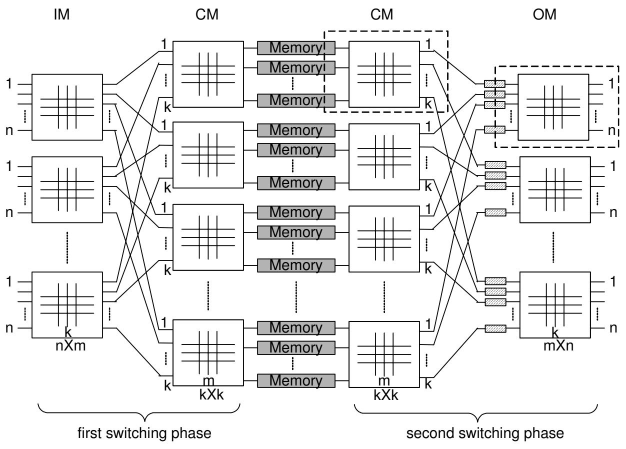

Fig. 2: Analysis on one of the central modules (CM).

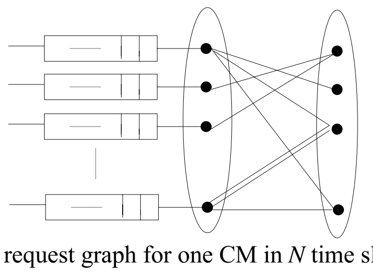

Fig. 3: A request graph for one CM in N time slots.

Similarly, we can analyze the scheduling of the OM, which is shown in the right dashed rectangle in figure 2. From the analysis above, we can see that in every N time slots there are at most N cells queued in one input of the OM. Therefore, the degree of the input i of the request graph for the OM is no more than N. The degree of the output j of the request graph is also no more than N since there are no more than N cells having to be switched out of one output port of the CBC in every N time slots. Using the same graph coloring theorem and from [12] we can see that there exists an algorithm scheduling the batch of cells in N time slots.

Birkhoff-von Neumann matrix decomposition

Our scheduling algorithm in the second switching phase is based on the Birkhoff-von Neumann matrix decomposition. This method has been employed in the Birkhoff-von Neumann input queued crossbar switches by C. S. Chang et al. [15]. However, we will see the differences later.

To explain the idea, let the request graph in figure 3 be represented by a traffic matrix

$$R = \begin{pmatrix} r_{11} & \cdots & r_{1k} \\ \vdots & \ddots & \vdots \\ r_{k1} & \cdots & r_{kk} \end{pmatrix},$$

with r_{ij} stands for r_{ij} cells requested to be switched out from memory i to output j. According to the analysis above, we can know the following two conditions hold:

$$\sum_{i=1}^{k} r_{ij} \le N, \forall j \text{............}(1)$$

$$\sum_{j=1}^{k} r_{ij} \le N, \forall i \text{............}(2)$$

where all the r_{ij} are integers.

Using Birkhoff-von Neumann matrix decomposition method below, we know that there exists a set of positive integers ϕ_i and permutation matrices P_i, $i = 1, 2, ..., K$ for some $K \le N$ that satisfies

$$R \le \sum_{i=1}^{K} \phi_i P_i \text{, and } \sum_{i=1}^{K} \phi_i = N .$$

Once we obtain such decomposition, we can simply schedule the connection pattern P_i proportional to its weight ϕ_i, $i = 1, 2, ..., K$.

Different from others' definition, we call in this paper a matrix $R = (r_{ij})$ that satisfies the conditions (1) and (2) to be *doubly substochastic*. If, furthermore, both inequalities in (1) and (2) are equalities, then the matrix $R = (r_{ij})$ is called *doubly stochastic*. The decomposition method consists of two steps: (i) it first finds a doubly stochastic matrix R' that is no smaller than the original request matrix R, and (ii) it then finds decomposition for the doubly stochastic matrix R'.

The first step is based on the following result by von Neumann.

Proposition 1: (von Neumann [18]) If a matrix $R = (r_{ij})$ is doubly substochastic, then there exists a doubly stochastic matrix $R' = (r'_{ij})$ such that $r_{ij} \le r'_{ij}, \forall i, j$. ■

The second step is based on Birkhoff's result on doubly stochastic matrices.

Proposition 2: (Birkhoff [14]) For a doubly stochastic matrix R', there exists a set of positive integers ϕ_i and permutation matrices P_i, $i = 1,2,...,K$ for some $K \le N$ such that

$$R' = \sum_{i=1}^{K} \phi_i P_i \text{, and } \sum_{i=1}^{K} \phi_i = N \text{ . } \blacksquare$$

C. Scheduling algorithm for the CBC to emulate an OQ switch with QoS guarantees

As we have established the basic Birkhoff-von Neumann matrix decomposition method for the second switching phase to schedule batches of cells, we now summarize the compelete scheduling algorithm for the CBC.

For the first switching phase of the CBC, every arriving cell is dispatched into the central buffers according to the three rules stated above.

For the second switching phase, every CM uses the Birkhoff-von Neumann matrix decomposition method applied to the request traffic matrix that is a $k \times k$ doubly substochastic matrix. We should do a little more work for every OM in the second switching phase since request traffic matrices for them are $m \times n$ dimensional. We first add $m - n$ (assuming $m > n$) column vectors with all elements being 0 to the request traffic matrix, making it a $m \times m$ dimensional doubly substochastic matrix. Then we apply the Birkhoff-von Neumann matrix decomposition method to it to obtain a sequence of connection patters P_i.

The following theorem states the memory requirements and delay bounds for the CBC to emulate an OQ switch supporting QoS.

Theorem 2: The CBC can emulate a PIFO-OQ switch within a relative delay of $3N - 1$ time slots, if $m \ge 4n - 1$, with no internal speedup.

Proof: First, we prove that each cell will be switched out within a relative delay of $3N - 1$ time slots.

Consider a newly arrived cell C and its possible earliest departure time is just now; we denote it as time slot 1. The worst case is that the CBC is now switching a full batch of cells, which will cost at most N time slots, and the cell C has to be lagged into the next batch of cells. So C will be switched out sometime between time slots $[N + 1, 2N]$. Hence, the maximum relative delay cell C can incur is $2N - 1$ time slots before arriving at the OM. In addition, cell C may stay in the buffer of OM for at most another N time slots before it is switched out of the CBC. So, the total maximum relative delay of cell C compared with that in the shadow PIFO-OQ switch will add up to $3N - 1$ time slots.

Secondly, we will calculate the number of memories needed here. Consider a cell *C* arriving at the input module and is about to be written into one of the central-stage memories. According to the first rule, there may be up to $N - 1$ memories that *C* cannot be written to. The second rule may make up to $(n-1)k$ memories incompatible with *C* since there may be up to $n - 1$ cells arriving at the same IM as *C* and each of them may occupy a whole bunch of *k* memories of one CM. The third rule may make up to $2(N - 1)$ memories incompatible with *C*.

Sum them up and use the pigeonhole principle, then we need at least

$$(N-1) + (n-1)k + 2(N-1) + 1$$
$$= 4N - k - 2, (N = nk)$$

memories. Therefore, $4N - k$ memories are sufficient for the CBC to emulate a PIFO-OQ switch within a constant delay. In terms of the number of central modules:

$$m \ge (4N - k)/k = 4n - 1 \text{ . } \blacksquare$$

We note that for emulating an OQ switch with QoS support, we need to nearly double the central modules of a traditional strictly non-blocking Clos-network where $m \ge 2n - 1$.

IV. Remarks and Practical Considerations

The main contribution of this paper is that we build a highly scalable packet switch with QoS support. Although we use the same method as the well known Birkhoff-von Neumann input queued switch in the main switching phase, we achieved a lower scheduling complexity which makes the CBC scalable. In particular, the scheduling algorithm for the Birkhoff-von Neumann input queued switch has a time complexity of $O(N^{4.5})$ [15], where N is the number of switch ports. In our multi-stage multi-layer CBC architecture, the complexities for the second switching phase are $O(k^{4.5})$ and $O(m^{4.5})$ respectively for the CM and OM scheduling. In practice, we usually set the parameters $n = k$ in the CBC. Thus $k = \sqrt{N}$ and $m \ge 4\sqrt{N} - 1$ for the CBC to emulate a PIFO OQ switch. So, the time complexity of scheduling in the CBC is $O(\sqrt{N}^{4.5}) = O(N^{2.25})$.

What is more, we distribute the computation of the Birkhoff-von Neumann matrix decomposition in a single layer switch into many smaller switches (layers), which are CM and OM in the second switching phase. We achieve this by using a multi-layer of switching. We shall remind here that not all multi-layer switches can distribute the matrix decomposition into small switches, since the decomposition needs the *'non-overbooking'* conditions (inequations of (1) and (2)) and the overall non-overbooking does not necessarily mean all the modest size switches inside have non-overbooking conditions as well if cells are not dispatched into memories properly.

Another difference of the CBC from the Birkhoff-von Neumann input queued switch is that the matrices here are real traffic matrices, while in the Birkhoff-von Neumann input-queued switch the matrix is an estimated rate matrix that relies on some estimation methods. We do exact packet scheduling by using real traffic matrices.

Compared with the load-balanced Birkhoff-von Neumann switch, the CBC can be regarded as doing a load balancing function in the first switching phase. After load balancing in the first switching phase of the CBC, cells are guaranteed to switch out within a constant delay. However, cells can only be guaranteed to switch out of the Load-balanced Birkhoff-von Neumann switch with a high probability under some benign traffic conditions. We achieve hard QoS guarantees at the cost of three hard rules observed by the first switching phase. While in load-balanced Birkhoff-von Neumann switches, they do not need real schedulers in the first stage.

When N is so large that the Birkhoff-von Neumann matrix decomposition cannot be completed in one time slot, we can make the decompositions pipelined. For example, if the time needed for the second switching phase to perform matrices decompositions in the CM and the OM is 2τ time slots, we make the decomposition processes and scheduling processes pipeline and the delay bound for every cell now will become $3N-1+2\tau$. The advantage of this pipelining is that the delay is bounded, and does not accumulate as well.

We said little about the details of the first switching phase. Since the task of the first switching phase is to find a *compatible* memory for every arriving cell under the three rules. Few bits of communication suffice for the memories and arriving cells to determine their compatibilities and the implementation details are out of the scope of this paper. We only focus on the advantages of the multi-stage multi-layer CBC architecture and the scheduling algorithms for the second switching phase in this paper.

V. CONCLUSIONS

In this paper, we first propose a scalable packet switch based on the Central-stage Buffered Clos-network. We analyze the memory requirements for the CBC to emulate an FCFS-OQ switch and an OQ switch with QoS support. We then design a scheduling algorithm based on the Birkhoff-von Neumann matrix decomposition method.

Our scheduling algorithm for the CBC extends the well-known Birkhoff-von Neumann input-queued switch into multi-stage multi-layer switches, thus making it highly scalable. Although there are still some obstacles unaddressed in this paper, the ability of decomposing a whole large switching system into many modest size switches which reserve similar traffic features makes the CBC a desirable solution for the next generation high performance switches/routers.

REFERENCES

[1] F. Wang and M. Hamdi, "Analysis on the Central-stage Buffered Clos-network for packet switching", in proceedings of *IEEE ICC*'05, May 2005.

[2] H. J. Chao, "Next generation routers," invited paper, *IEEE Proceeding*, vol. 90, no. 9, pp. 1518-1558, Sep. 2002.

[3] C. Clos, "A study of Non-Blocking Switching Networks," *Bell Systems Technical Journal*, pp. 406-424, March 1953.

[4] F. M. Chiussi, J. G. Kneuer and V. P. Kumar, "Low-cost scalable switching solutions for broadband networking: the ATLANTA architecture and chipset," *IEEE Communication Magazine*, vol. 35, no. 3, pp. 44-53, December 1997.

[5] H. J. Chao, K. Deng and Z. Jing, "A Petabit Photonic Packet Switch (P^3S)," in proceedings of *IEEE Infocom*'03, March 2003.

[6] S. T. Chuang, A. Goel, N. McKeown and B. Prabhakar, "Matching output queuing with a combined input output queued switch," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 6, pp 1030-1039, June 1999.

[7] S. Iyer, A. Awadallah, and N. McKeown, "Analysis of a packet switch with memories running slower than the line rate," in proceedings of *IEEE Infocom* 2000.

[8] S. Iyer, R. Zhang, N. McKeown, "Routers with a single stage of buffering," in proceedings of *SIGCOMM*'02.

[9] Demers, S. Keshav, S. Shenker, "Analysis and simulation of a fair queuing algorithm," *J. of Internetworking: Research and Experience*, pp. 3-26, 1990.

[10] 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, June 1993.

[11] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round robin," in proceedings of *SIGCOMM'95.*

[12] D. Konig, Uber Graphen und ihre Anwedung auf Determinatentheorie und Mengenlehre, *Math. Ann.* 77 (1916), pp. 453-465 (in German).

[13] http://www.cisco.com

[14] G. Birkhoff, "Tres observaciones sobre el algebra lineal," *Univ. Nac. Tucuman Rev. Ser. A*, Vol. 5, pp. 147-151, 1946.

[15] C. S. Chang, W. Chen and H. Huang, "Birkhoff-von Neumann input buffered crossbar switches", in proceedings of *IEEE Infocom* 2000.

[16] L. Mirsky, "Transversal Theory", *New York: Academic Press*, 1971.

[17] C. Berge, "The theory of graphs and its applications", *New York: Wiely,* 1962.

[18] J. von Neumann, "A certain zero-sum two-person game equivalent to the optimal assignment problem", *Contributions to the theory of games,* Vol. 2, pp 5-12, Princeton University Press, Princeton, New Jersey, 1953.

[19] C.S. Chang, D.S. Lee and C.M. Lien, "Load balanced Birkhoff-von Neumann switches, Part II: multi-stage buffering," *Computer Comm.*, Vol. 25, pp. 623-634, 2002.