The Central-stage Buffered Clos-network to emulate an OQ switch

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

Abstract – **In this paper[1], we propose a highly *scalable packet* switch that is based on a multi-stage multi-layer architecture made up of many modest size switches. This new architecture resembles the famous Clos-network studied in *circuit* switching systems except that it has distributed shared memories in the central stage. We call it Central-stage Buffered Clos-network (CBC). We first analyze the memory requirements for the CBC to emulate an output-queued (OQ) switch since OQ switches are generally regarded as having the optimal delay-throughput performance. Then we design an efficient packet-scheduling algorithm for the CBC to emulate an FCFS OQ switch. We show two distinguished features of this algorithm. First, it converges to the maximum matching *faster* than any other scheduling algorithms using the same paradigm. Secondly, the performance of the algorithm is *independent* of any arriving traffic pattern, which is not seen in other scheduling algorithms, such as iSLIP, DRRM and so on…**

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 [1]. 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 modest size switches. Building huge switches/routers out of smaller ones based on a multi-stage multi-layer idea has been studied extensively in the literature. In the area of *circuit* switching, researchers are much familiar with the scalable Clos-network that came into being more than 50 years ago [2]. When designing scalable packet switches, it is natural to use the same idea as that in the Clos-network. The problems here are *where and how to put buffers in the Clos-network* since we are dealing with *packets* which need to be stored and forwarded.

Along this idea, many implementations are proposed. However, they mainly use Memory-Space-Memory (MSM) architectures [1, 3], i.e. they just put memories in both the input and output sides. Although this strategy may adopt good algorithms and results from the existing single-stage crossbar-based switches, the memories are not efficiently shared no matter whether we put the majority of memories in the input side or in the output side. Memory sharing may seem trivial in small switches but is very important in a large switching system since memory is always the costly resource and bottleneck in routers. In order to build a highly *scalable* router and make memories fully *shared*, we base our architecture on the Clos-network and put memories only in the central stage. Single-stage buffering [4] may make 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.

When evaluating new switch architectures, the output-queued (OQ) switch model is always a desirable standard to be compared with, since it is normally regarded to have the optimal delay-throughput performance. In our previous work, we have analyzed memory requirements [5] and designed a scheduling algorithm [6] for the CBC to emulate a *general* OQ switch as well. However, the scheduling algorithm has a time complexity of $O(N^{2.25})$ and packets may experience a bounded delay. What are more, as many other architectures that buffer packets in the middle stage, packets may incur the *out-of-sequence* problem. In this paper, we try to make the emulation simple and practical. We design an efficient scheduling algorithm for the CBC to emulate a first-come-first-served (FCFS) OQ switch. We show two distinguished features of this algorithm. First, it converges to the maximum matching *faster* than any other scheduling algorithms using the same paradigm. Secondly, the performance of the algorithm is *independent* of any arriving traffic pattern. In addition, the CBC is free of the out-of-sequence problem since the whole system emulates an OQ packet switch.

The rest of this paper is organized as follows. In section II, we introduce the architecture of the Central-stage Buffered Clos-network (CBC). In section III, we present the RGRG scheduling algorithm for the CBC to emulate an FCFS-OQ switch. The performance analysis of the RGRG algorithm is carried out in section IV. Then in section V, we have a conclusion.

[1] This research is supported under RGC HKUST 6200/02E.

1-4244-0357-X/06/$20.00 ©2006 IEEE

II. THE CENTRAL-STAGE BUFFERED CLOS-NETWORK

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 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.

Departure time – Refers to the time a cell is scheduled to leave the switch system. It is assigned to the cell according to its arriving order and never changes in the future if all cells are scheduled in an FCFS manner. In prioritized service disciplines, cells may change their departure time due to arriving higher priority cells.

Output-queued (OQ) switch – A switch in which arriving packets are placed *immediately* in queues at the output, where they contend with packets destined to the same output waiting for their turn to depart. One characteristic of an OQ switch is that the buffer memory must be able to accept (write) N new cells and read one cell per time slot, where N is the number of ports. Hence, the memory must operate at $N+1$ times line rate.

OQ switch emulation – For a packet switching system, if the real departure time of packets through it is identical with that from the shadow OQ switch fed with the same incoming traffic, we say that this packet switching system *emulates* an OQ switch.

B. The Central-stage Buffered Clos-network for packet switching

We proposed a new model (CBC) made of multi-stage multi-layer smaller packet switches. As shown in figure 1, this 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. A cell traverses the CBC like this. It is first switched in the first switching phase, reaching and staying in one of the central memories. When it is time for the cell to go, the cell is switched out from the memory in the second switching phase.

As in 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 CMs is $m\times k$ 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 $N = n\times k$.

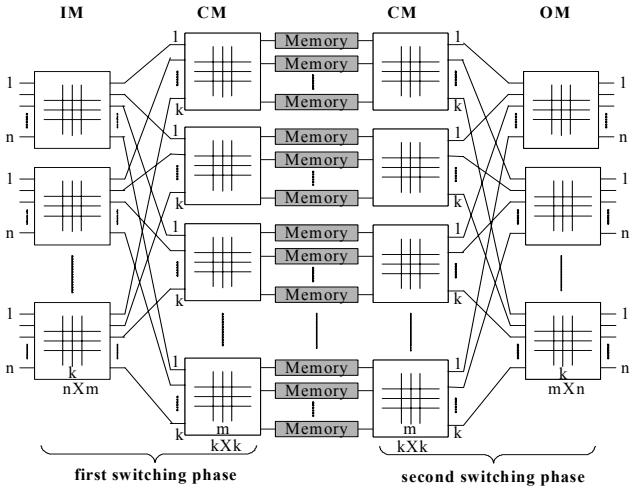

Fig. 1: Architecture of the Central-stage Buffered Clos-network

The reason why we use two copies of the CM is that we want to make the memories fully *shared* among all the input/output ports. Otherwise, if we use only one copy of the CM, e.g. missing all the CMs in the first switching phase, then we cannot make every memory accessible by all the input ports, which will make inefficient use of all the memories.

In the following OQ switch emulation, we assume no speedup in the CBC, which indicates that in one time slot one link can only transfer one cell and one memory can only support one *read and write*. Therefore, it is natural to define the following *compatibility*.

For an incoming cell A and a memory B, A's departure time being t and B belonging to a central module M, we say memory B is compatible with cell A, if the following conditions hold.

1. B does not contain cells whose departure time is t.
2. If there are cells bearing the departure time t in other memories linking module M, those cells do not request the same OM as which A requests.

This definition of *compatibility* is important for the CBC's second switching phase to work properly. The first condition guarantees at most one read to the memory B at time t. The second condition guarantees at most one cell that will subscribe the link between the CM and OM at time t. We can see that if all cells are switched into the memories from the first switching phase under this compatibility constraint, it is trivial for them to be switched out of the second switching phase, since the compatibility property removes all contentions among the cells waiting in the central stage memories.

C. Memory Requirements for the CBC to emulate an FCFS OQ switch

In our previous work [5], we have derived the basic memory requirements for the CBC to emulate an FCFS-OQ

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.

switch. The result is shown by the following theorem and corollary.

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 [5].

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 [2] ($m \geq 2n-1$) as a black box and make a *combined-input-output-queued* (CIOQ) switch out of it, then according to [8] 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 RGRG Algorithm for CBC to Emulate an FCFS-OQ Switch

In this paper, we propose a randomized algorithm in the first switching phase to switch incoming cells into the central memories under the compatibility constraint. Using the results we derive in the last section, we assume $m = 4n$ in the following description of the algorithm and its analysis. Thus, we have $4nk = 4N$ memories in total in between the CMs.

We represent the first switching phase by a bipartite graph as shown in figure 2. An edge between an input and a memory indicates that the memory is *compatible* for that incoming cell. Circles group inputs/memories into the CBC modules. A *match* in this graph corresponds to a switch configuration in one time slot. Conventional match definition dictates that one vertex be linked by at most one edge. The Clos-structure imposes yet another constraint to the final match since there is *one and only one* link between an IM and a CM. Concerning fig. 2, two edges linking the same pair of circles are not allowed in the final match.

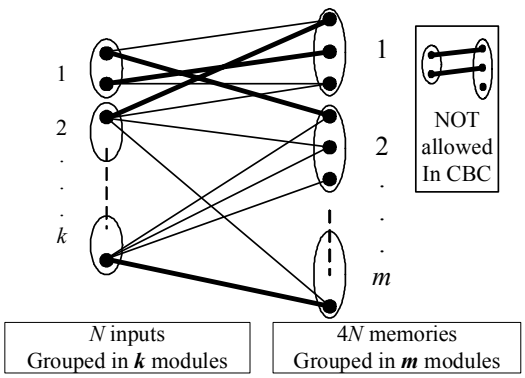

Fig. 2: Bipartite graph representation in the first switching phase

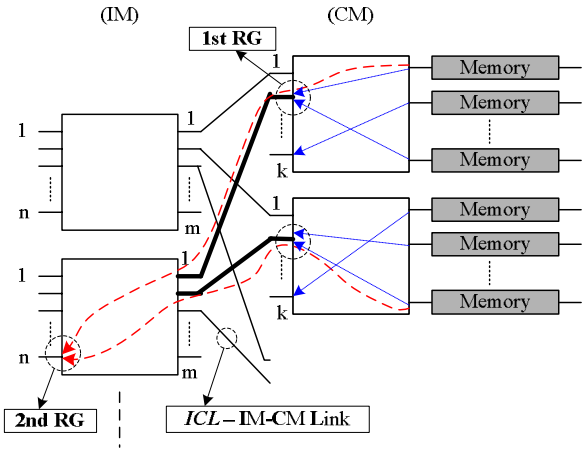

Fig. 3: RGRG matching algorithm

Therefore, we define a CBC match of the bipartite graph shown in figure 2. Every set of edges is called a *CBC Match* if it conforms to the following two conditions:

1. No two edges in the set share a vertex.
2. There is at most one edge between two circled sets of inputs/memories.

For example, the set of thick edges in figure 2 forms a CBC match.

Scheduling in the first switching phase can be abstracted into finding a CBC Match for all the cells arriving every time slot. It seems a little harder to find a maximum *CBC match* since an additional constraint is imposed on the graph. However, we shall show that the CBC can employ a very efficient and simple scheduling algorithm.

The proposed scheduling algorithm is based on the similar *'Request-Grant-Accept'* paradigm of the prevailing crossbar schedulers, such as iSLIP [9], DRRM [10] and so on… but it is proven to converge to the maximum matching *faster* than any of them does. Another distinguished feature of this algorithm is that its performance is *independent* of the arriving traffic patterns.

The algorithm runs in rounds. We call it RGRG algorithm since every round involves two phases of '*Request-Grant*'. As shown in figure 3, the basic idea is to let memories go to match cells. We define links between the IM and CM as *ICL*. The first RG steps are for the memories contending for the use of *ICL*. The second RG steps are for the memories contending for compatible cells via the obtained *ICL*. We define a *basic round operation* as follows:

R1 step: Every unmatched memory *randomly* selects one unmatched *ICL* in its linked CM and sends a request to it.

G1 step: After an *ICL* receives all requests from memories, it chooses *randomly* one of them and grants that request.

R2 step: After a memory receives a grant from the *ICL* it requests, it uses that *ICL* to send a request *randomly* to a compatible cell in the IM linking to that *ICL*. If there is no compatible cell in that IM, the memory does nothing.

G2 step: After a cell receives requests from memories, it chooses *randomly* one of them and grants that memory.

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.

All memories, *ICL*s and cells operate in parallel. After memories receive grants from cells in G2 step, a set of matches are formed, and then the compatibility information is updated. Another round is performed if there are cells left unmatched.

We devote the following section to finding how many rounds are needed for the RGRG algorithm to find to a maximum *CBC match*.

IV. ANALYSIS OF THE RGRG ALGORITHM

Lemma 1: If there are r unmatched cells in an IM, there are at least $N+kr+n+1$ memories that are compatible with each of the r unmatched cells.

Lemma 1 is very easy to verity. We skip the proof due to the length limit of the paper.

Lemma 2: In G1 step of the RGRG algorithm, the probability for a memory M to be selected by the *ICL* it requests is $1-e^{-1}$. It is *independent* of the traffic patterns and number k, the size of central module (CM).

Proof: Focus on one CM in the G1 step. Assume there are s memories unmatched, then no matter what the traffic pattern is, there are also s *ICL*s remaining unmatched. It is well know that when casting s balls into s boxes randomly, the probability for one ball to be chosen by a box is $1-e^{-1}$. This result holds as long as the number of balls equal to the number of boxes. In the RGRG matching process, the number of unmatched memories is always equal to the unused *ICL*s linking the CM, and this fact is *independent* of the traffic pattern and the number k. Therefore, the probability of the memory M to be selected by the *ICL* it requests is always $1-e^{-1}$, regardless of the traffic pattern and number k.

■

Lemma 3: If there are r unmatched cells in an IM, then the probability for a cell to remain unmatched after a RGRG *basic round operation* is less than $e^{-(1-e^{-1})(1+\frac{n}{r}+\frac{n+1}{kr})}$.

Proof: Consider a cell A and one of its compatible memories B. A and B are linked by one and only one *ICL* which we call C. The event that B will survive to send a request to A in R2 step only comes after these three facts:

a) B has sent a request to the *ICL* C in R1 step.
b) The *ICL* C has chosen B in G1 step.
c) B has sent a request to A via C in R2 step.

The probability for a) is at least $1/k$ since there are at most k unmatched *ICL*s in one CM. The probability for b) is $1-e^{-1}$, according to lemma 2. The probability for c) is $1/r$, since there are r unmatched cells. Therefore, the probability for B to survive to send a request to cell A is at least $(1/k)(1-e^{-1})(1/r)=(1-e^{-1})/kr$. As long as cell A receives at least one request from all its compatible memories whose number is larger than $N+kr+n+1$ according to lemma 1, A will be matched for sure.

Packet A will remain unmatched only if none of its compatible memories survives to send it a request. The probability for this event is less than

$$\left(1-\frac{1-e^{-1}}{kr}\right)^{N+kr+n+1}$$

$$=\left(1-\frac{1}{kr/(1-e^{-1})}\right)^{kr/(1-e^{-1})\times\frac{(1-e^{-1})(N+kr+n+1)}{kr}}$$

$$\to e^{-(1-e^{-1})(1+\frac{n}{r}+\frac{n+1}{kr})}$$

■

We can notice that as r becomes smaller in the matching process, this number converges *super exponentially* to 0. This provides a basic intuition that the RGRG algorithm converges quickly to a maximum CBC match.

Theorem 2: The RGRG algorithm converges to the maximum CBC match in $O(\log N)$ time in the worst case, and in $O(\log^* N)$ [2] time in average time. The convergence is independent of traffic patterns.

Proof: According to lemma 3, we know that the number of unmatched cells decreases at least in a *geometric* ($e^{-(1-e^{-1})}$) speed, which will obviously result in a $O(\log N)$ time to converge to 0. For the averaging $O(\log^* N)$ time analysis, please refer to the appendix.

■

This theorem tells that the converging speed of the RGRG is much *faster* than iSLIP and the DRRM whose converging time is $O(\log N)$ in average and $O(N)$ in worst case. Lemma 2 is very essential to provide the *traffic-independence* property of the RGRG algorithm.

V. CONCLUSIONS

Recently, Cisco unveiled its next generation routing system that can scale up to 92 terabits per second (Tbps), powering the first OC-768c/STM-256c IP interface and supporting up to 1152 40-Gbps line-card slots. According to its technical reports, the switching fabric is a cell switch based on buffered three-stage *Benes* architecture. Since it is a commercial product, it did not provide any analysis on the delay and throughput performance.

Our work is motivated particularly by the advent of this new router. We propose a scalable switch based on the Central-stage Buffered Clos-network (CBC) architecture. We analyze the memory requirements and design a randomized scheduling algorithm called RGRG algorithm for the CBC to emulate an FCFS-OQ switch.

---

[2] $\log^* N \equiv \min\{k \mid \underbrace{\log\log\ldots\log}_{k} N < 1\}$, which is less than 5 in real world, where N is normally less than 2^{65536}.

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.

Although the '*Request-Grant*' paradigm is not new in designing packet scheduling algorithms, it is still surprising to see that two phases of RG can converge even *faster*. In fact, it should be regarded as one of the inherent properties of the CBC architecture. We also reveal that two phases of RG applied on the CBC make the scheduling performance *independent* of traffic patterns, which is rather different from other similar algorithms, such as iSLIP, DRRM and so on… Intuitively, the first RG phase shapes the traffic and makes it distribute uniformly into the fully shared central memories.

The most difficulty to implement the CBC is to maintain the compatibility information. One simple solution may employ a bit matrix to keep updating the compatibility information. However, as the CBC scales, the matrix might become too large to manage. We will address this problem in our future work.

Compared with commercially available crossbar based single-stage switches, we believe that the CBC architecture with this randomized RGRG scheduling algorithm is a promising, scalable and simple design for high performance switches (routers) in future.

REFERENCES

[1] H. J. Chao, K. Deng, and Z. Jing, "A Petabit Photonic Packet Switch (P3S)," presented at IEEE Infocom, 2003.

[2] C. Clos, "A study of Non-Blocking Switching Networks," *Bell Systems Technical Journal*, pp. 406-24, 1953.

[3] 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, pp. 44-53, 1997.

[4] S. Iyer, R. Zhang, and N. McKeown, "Routers with a single stage of buffering," presented at ACM SIGCOMM, 2002.

[5] F. Wang and M. Hamdi, "Analysis on the Central-stage Buffered Clos-network for packet switching," presented at IEEE International Conference on Communications, Korea, 2005.

[6] F. Wang and M. Hamdi, "Scalable Central-stage Buffered Clos-network Packet Switches with QoS," presented at IEEE Workshop on High Performance Switching and Routing, Poland, 2006.

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

[8] 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, pp. 1030-1039, 1999.

[9] N. Mckeown, P. Varaiya, and J. Warland, "The iSLIP Scheduling Algorithm for Input-Queued Switch," *IEEE Transaction on Networks*, 1999.

[10] H. J. Chao and J. S. Park, "Centralized contention resolution schemes for a large-capacity optical ATM switch," presented at IEEE ATM workshop, 1998.

APPENDIX

Proof of Theorem 2:

From Lemma 3, we know that after one RGRG round the number of unmatched cells $r' = re^{-(1-e^{-1})(1+\frac{n}{r}+\frac{n+1}{kr})}$, where r is the number of unmatched cells in the last round and $r = N$ in the first round. It is obvious to see that the sequence of r is bounded by a geometric sequence whose convergence ratio is $e^{-(1-e^{-1})}$. It is easy to find that r will decrease to less than a fixed arbitrarily small number ε in $O(\log N)$ time.

However, we know that the result in lemma 3 comes from the worst case that all the unmatched cells gather in the same input module. To calculate the average convergence time, we assume that r cells are distributed in all k IM uniformly. We shall note here that this uniformity is *independent* of the traffic patterns. It only relies on the RGRG matching process.

If all the r unmatched cells are distributed in k IM, the formula in lemma 3 will change to

$$\left(1-\frac{1-e^{-1}}{k\times(r/k)}\right)^{N+kr+n+1}$$

$$=\left(1-\frac{1}{r/(1-e^{-1})}\right)^{r/(1-e^{-1})\times\frac{(1-e^{-1})(N+kr+n+1)}{r}}$$

$$\to e^{-(1-e^{-1})(\frac{N}{r}+k+\frac{n+1}{r})}$$

So, the recursive formula for r becomes

$$r' = re^{-(1-e^{-1})(\frac{N}{r}+k+\frac{n+1}{r})} < re^{-(1-e^{-1})\frac{N}{r}}$$

Substitute r recursively and let $r_0 = N$, we can easily get

$$r_t < \frac{N}{\underbrace{e^{(1-e^{-1})e^{(1-e^{-1})}\ldots e^{(1-e^{-1})}}}_{t}}$$

For any fixed arbitrary small ε, solve the function

$$\frac{N}{\underbrace{e^{(1-e^{-1})e^{(1-e^{-1})}\ldots e^{(1-e^{-1})}}}_{t}} < \varepsilon$$

$$\Rightarrow \frac{N}{\varepsilon} < \underbrace{e^{(1-e^{-1})e^{(1-e^{-1})}\ldots e^{(1-e^{-1})}}}_{t}$$

$$\Rightarrow \underbrace{\log_{e^{(1-e^{-1})}}\log_{e^{(1-e^{-1})}}\ldots\log_{e^{(1-e^{-1})}}\frac{N}{\varepsilon}}_{t} < 1$$

The minimum of t is $\log^* N$, which is the time for r to converge to a fixed arbitrarily small number.

1-4244-0357-X/06/$20.00 ©2006 IEEE

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.