This full text paper was peer reviewed by subject matter experts for publication in the MULTICOMM 2006 proceedings.

# BMC: A Two-stage Switch Architecture for High Performance Multimedia Communication

Yang Xu[1], Bin Liu[2], Beibei Wu[3], Wei Li[4]
Department of Computer Science and Technology
Tsinghua University
Beijing 100084, P. R. China

{xy01[1], wbb02[3], li-wei03[4]}@mails.tsinghua.edu.cn, liub@tsinghua.edu.cn[2]

***Abstract*—Output Queueing (OQ) emulated switch is desirable in real-time applications because it can guarantee the lowest average cell delay. The challenging issue in OQ emulated two-stage switch is the extreme complexity of the scheduling operations between the first stage switch and the central shared memories. To overcome this problem, a new switching architecture called Banyan-Memory-Crossbar (BMC) is proposed in this paper. In BMC, the first stage switch is a banyan-like interconnection network, which is used to route the incoming cells from input ports to central memories without occurring arrival and departure conflicts. Several different implementations of banyan network are given in this paper, which are named as RANDBN, F$^2$BN, and SF$^2$BN respectively. Compared with previous methods in OQ emulated two-stage switch, they don't need centralized scheduler, causing the communication and computation overhead reduced dramatically. Theoretical analysis and simulations show that the cell loss rates in RANDBN and SF$^2$BN are 21.31%, while the cell loss rate of F$^2$BN is less than 1%. Delay guarantee capability and such a slim cell loss rate make F$^2$BN practical in real-time applications.**

***Index Terms*-- Multimedia Communication, Two stage switches, High-speed, Banyan, Scheduling***

## I. INTRODUCTION

Switch/Router is the infrastructural equipments of multimedia communication, where cell delay and throughput/drop ratio are two important performance measurements. (For real-time applications, cell delay is more important).

Output Queueing (OQ)[1] switch is known as the ideal switch model because it can guarantee the maximum throughput and the lowest average cell delay (the delay of each cell can also be predicted and easily controlled[2][3]), thus is desirable for high performance multimedia communication. However OQ is extremely hard to implement under high line speed or large port number environments, the reason lies in both the switch fabric and the output queues need to run at $N$ times the line rate, where $N$ is the number of ports.

Therefore several more scalable architectures, which less speedup, were proposed to emulate the behavior of OQ. One of such architectures is so called Combined Input and Output Queueing (CIOQ) [4][5][6]. It has been proven that a CIOQ switch with a speedup of 4 or 2 can exactly emulate an OQ switch by employing specially designed scheduling algorithms, such as MUCFA, CCF, and JPM, but none of these presented algorithms is practical in real routers/switches for their high time complexity.

Besides CIOQ switches, the Switch-Memory-Switch (SMS) architecture provides another way towards emulating OQ[7][8][9][10][11]. In SMS switches, two crossbar fabrics are used in the first and second stage switching respectively, and between them located some shared memories. The main problem in SMS switch is the scheduling in the first stage, where two kinds of conflicts must be avoided: arrival conflict and departure conflict[9][11]. RiPSS, a request-grant-accept based scheduling algorithm, was proposed together with its pipeline version PRiPSS in [8] and [9]. RiPSS needs $(2N+\varepsilon)$ independent memories (where $\varepsilon$ is a very small positive number), and can complete the matching with $O(\log^* N)$ rounds of iterations w.h.p in $N$. Despite the good performance, RiPSS/PRiPSS requires a bipartite graph without departure conflict set up in each time slot, which has the time complexity of $O(N)$ [11]. So RiPSS/PRiPSS is unpractical.

The objective of this paper is to design a new switch architecture optimized for real-time applications, which should have the following characteristics:

- ✓ Guaranteed cell delay performance with acceptable cell loss rate. For multimedia applications, especially real-time applications (such as VoIP and video conferencing), cell delay is the most crucial measurements, while slim cell loss is acceptable.
- ✓ The scheduling in the architecture should be simple enough to be implemented in very high-speed environment.

The rest of this paper is organized as follows. In section II, we introduce a new two-stage switch architecture named Banyan-Memory-Crossbar (BMC), which uses a banyan network to route the incoming cells into shared memories without occurring arrival and departure conflicts, so that emulates a FIFO-based OQ architecture. Section III presents several such banyan networks, whose performance evaluations and simulation results are given in section IV. Finally section V gives the conclusion.

Supported by the NSFC under Grant No. 60373007 and No. 60573121;China-Ireland Science and Technology Collaboration Research Fund (CI-2003-02) and Specialized Research Fund for the Doctoral Program of Higher Education of China (No. 20040003048);985 Fund of Tsinghua University(No. JCpy2005054)

## II. The Architecture of Banyan-Memory-Crossbar Switches

In this paper, we assume that only fixed sized packets, also called cells, are transferred inside the switch.Time is divided into time slots and one time slot is equal to the transmission time of a cell.

### *A. How to Emulate an OQ Switch with FIFO Policy*

Our goal is to design an OQ emulated switch. Here, we give the definition of OQ emulated switches firstly.

Definition 1: OQ emulated switches (in narrow sense)[5]:

A switch is said to be an OQ emulated switch, if under identical inputs, the departure time of every cell from this switch is exactly identical to that from an OQ switch (The detailed process to compute each cell's departure time can be referred to [8].). If a cell is dropped in an OQ switch, it will be dropped in this OQ emulated switch too.

Definition 2: OQ emulated switches (in broad sense):

A switch is said to be a broad-sense OQ emulated switch, if under identical inputs, the departure time of every cell from this switch adds a fixed delay to that from an OQ switch. If a cell is dropped in an OQ switch, it will be dropped in this OQ emulated switch too.

In the rest of this paper, OQ emulated switches refer to OQ emulated switches in broad sense, if there is no other special notation.

### *B. Banyan-Memory-Crossbar Architecture*

Here, we present a novel switch architecture named Banyan-Memory-Crossbar (BMC) to emulate the OQ switch, which is composed of three parts: distributing banyan network, shared memories, and crossbar fabric. Fig. 1. depicts the architecture of a BMC switch with 8 ports, which is composed of three parts:

1) Distributing banyan network
2) Shared memories
3) Crossbar

Distributing banyan network is a $2N \times 2N$ banyan network, which connects $N$ input ports and $2N$ shared memories ($N$ is assumed to be a power of 2). It consists of $\log_2 N+1$ phases. Each phase contains $N$ $2\times 2$ switch elements (SE) with two states: either cross or bar. The SE located at row $j$ and column (phase) $j$ is marked as $SE_{i,j}$, $(0 \le i \le N-1, 0 \le j \le \log_2 N)$.

Cells are assumed to be transferred from current phase SE to the next phase SE every time slot. So it needs $\log_2 N+1$ time slots for a cell to be transferred from the input ports to the shared memories crossing the banyan network.

Assume there exists a shadow OQ switch with FIFO policy, where the length of each queue is $L$. To use BMC to emulate the shadow OQ switch, arrival cells must be put into memories satisfying the following two requirements:

(1) Cells arriving simultaneously should be put into different memories.

(2) Cells with the same departure time should also be put into different memories.

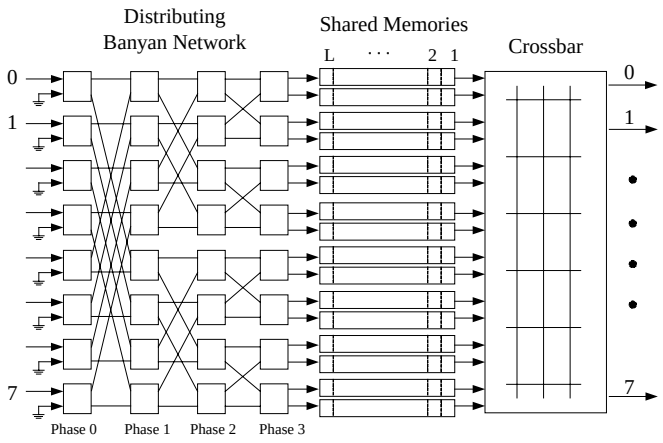


Fig. 1. The architecture of Banyan-Memory-Crossbar switch (an example with $N$=8).

When the first requirement is satisfied, we say arrival conflicts are avoided, and when the second is satisfied, departure conflicts avoided. For there are at most $N$ cells arriving simultaneously, and at most $N$ cells having the same departure time, it can be easy proven that when $M \ge 2N-1$ ($M$ is the number of shared memories), it is enough to put a new arrival cell into an available memory without conflict[7][10]. In this way, the operations of such a two-stage switch in each time slot can be divided into two parallel processes/jobs:

(1) In the first stage switching, new arrival cells are scheduled into shared memories without arrival and departure conflict.

(2) In the second stage, cells whose departure time is equal to the current time slot are read out from the memories and sent to the corresponding output ports. This job is easy to be done for all these cells are staying in different memories, and destined to different outputs.

So the challenge in BMC mainly lies in the first stage where cells must be scheduled without arrival and departure conflict. In the left of this paper we focus on the scheduling in the first stage switch.

### *C. Why employing banyan network*

The banyan type of interconnection network was originally defined in [13]. It has the property of exactly one path from any input to any output, which can be used to avoid arrival conflict.

Here we make a comparison between traditional conflict-avoiding solutions (such as RiPSS/PRiPSS) and our banyan-based method. In traditional solutions, each memory first finds out which inputs are departure-time compatible to it, and then a bipartite graph without departure conflict is constructed. Then the scheduling algorithm is executed to find a matching without arrival conflict based on that bipartite graph. In these methods, the time complexity to construct a bipartite graph without departure conflict is very expensive ($O(N)$ for each memory), and we call them as *departure-conflict-avoiding-first* method.

Our method, however, utilizes the characteristic of banyan networks that there is exactly one path from any input to any output to avoid the arrival conflict firstly. The remainder work for the banyan network is to find an appropriate method to avoid the departure conflict. We call it *arrival-conflict-*

*avoiding-first* method. The strongpoint of *arrival-conflict-avoiding-first* methods is that it avoids the computing overhead to construct a bipartite graph without departure conflict, so that it is more practical in high-speed environment.

## III. DISTRIBUTING BANYAN NETWORK

In this section, we present several kinds of Banyan network, which are used in BMC switches to avoid the departure conflict.

When a cell arrives at an input port, its departure time is first computed based on the shadow OQ switch[9], and a label will be tagged to its header before it enters the BMC switch. The format of a tagged cell is shown in Fig. 2, where *DP* is the destination port of the cell, with the length of $\log_2 N$, and *DT* presents the departure time of this cell in the shadow OQ switch, with the value denoted as follows.

$$DT=\text{DepartureTime(cell)} \bmod L. \quad (1)$$


Fig. 2. Format of Cells entering the BMC switch.

### A. Random Banyan Network

We first use a Random Banyan Network (RANDBN) to solve the departure conflict. Denote the probability of state 'cross' in $SE_{i,j}$ at time slot $t$ as $Pr(SE_{i,j}=Cross, t)$, and state 'bar' as $Pr(SE_{i,j}=Bar, t)$. Then in RANDBN, we let

$$Pr(SE_{i,j}=Cross, t)= Pr(SE_{i,j}=Bar, t)=\frac{1}{2},$$

Where $0 \le i \le N-1,\ 0 \le j \le \log_2 N,\ t \in Z^+ \bigcup \{0\}$.

When a cell arrives at a SE, it will be forwarded away according to the state of SE in a random manner. After traversing all the phases of RANDBN and reaching one certain shared memory, the cell will be accepted if no cell with the same *DT* is in this memory. Otherwise, it will be dropped, and we call this a *departure conflict*.

### B. Flip-Flop Banyan Network

It will be later seen in section IV that the cell conflict probability in RANDBN is quite high. To decrease the conflicts, we ask the banyan network remember the path each cell traversed, so that when a new cell arrives the banyan network can try its best to prevent it from reaching a memory that has already been occupied by another cell with the same departure time.

Here, we give some useful definitions firstly, and then introduce a novel Flip-Flop Banyan Network ($F^2BN$).

*Earliest Departure Time* ($EDT_i(t)$): in time slot $t$, to SEs in the $i$-th phase, the earliest departure time of all possible arrival cells.

*Latest Departure Time* ($LDT_i(t)$): in time slot $t$, to SEs in the $i$-th phase, the latest departure time of all possible arrival cells.

For example in time slot $t$, the departure times of cells at $SE_{*,0}$ range from $(t+1) \bmod L$ to $(t+L) \bmod L$ (Cells whose departure time is larger than $t+L$ will be dropped in the shadow OQ switch). So $EDT_0(t)= (t+1) \bmod L$, and $LDT_0(t)= (t+L) \bmod L$.

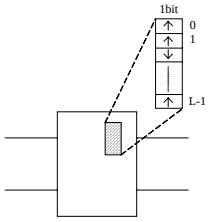


Fig. 3. Forwarding information table in each SE.

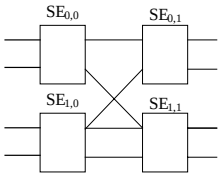


Fig. 4. A $4\times4$ $F^2BN$.

More generally, for SEs in phase $i$,

$$EDT_i(t)= (t+1-i) \bmod L, \quad (2)$$

$$LDT_i(t)= (t+L-i) \bmod L. \quad (3)$$

#### 1) The Structure and Behavior of Each SE

To record the path information, in $F^2BN$, we give each SE an independent Forwarding Information Table (FIT), and denote FIT at $SE_{i,j}$ as $FIT_{i,j}$. FIT is used to remember the outlet of cells with different *DT*s (between *EDT* and *LDT*), and the sketch map of FIT is shown in Fig. 3.

Each FIT has $L$ entries, and each entry only has 1 bit. We assume '0' presents 'up', and '1' 'down'. In the beginning, all entries in FITs have an initial value, e.g. 'up' or 'down'. When a cell with $DT = t_k$ arrives at $SE_{i,j}$, $SE_{i,j}$ will check the $t_k$-th entry in $FIT_{i,j}$. If the value of this entry is '0', the cell will be forwarded to the upper line of the SE, otherwise it will be forwarded to the lower line of the SE. After that, the value of the $t_k$-th entry in $FIT_{i,j}$ will be reversed.

$F^2BN$ has a very useful property:

**Theorem 1**. Assume $N$ same-departure-time cells arrive at a $2N\times2N$ $F^2BN$ network slot by slot. If there is no other cell arriving, these $N$ cells will be routed definitely to different outlets.

*Proof*: Let the number of cells traversing $SE_{i,j}$ be $X_{i,j}$.

For simplicity, we only give the proof of $N$=2 (as shown in Fig. 4).

Because each entry in FIT works as the round-robin manner, so long as $X_{0,1} \le 2$ and $X_{1,1} \le 2$, each outlet will receive no more than one cell.

Similarly, we can get

$$\begin{aligned} X_{0,1} &\le \left\lceil \frac{X_{0,0}}{2} \right\rceil + \left\lceil \frac{X_{1,0}}{2} \right\rceil \\ &\le \frac{X_{0,0}+X_{1,0}}{2}+1 \end{aligned} \quad (4)$$

Since there are at most two cells with the same departure time, $X_{0,0}+X_{1,0} \le 2$. Substitute it into (4), we get

$$X_{0,1} \le 2 \quad (5)$$

In the same way,

$$X_{1,1} \le 2 \quad (6)$$

Combining (5) and (6), it can be concluded that each outlet will receive no more than one cell.

□

Theorem 1 shows the fact that all cells can be routed to different memories if there is no collision along their paths. But actually cells with different departure times coexist in the distributing banyan network, the collision is inevitable when cells travel through the distributing banyan network. Hence two points must be stressed:

(1) In each time slot, there are at most two cells arriving at a certain SE simultaneously. If these two cells are all scheduled to the same outlet, collision will occur. So one of them must be dropped or shifted to the unwilling outlet, in this case we call it a *bump*. In $F^2$BN no cell will be dropped in the SE, and when a *bump* happens, the cell with low priority will be shifted to the idle outlet, and the corresponding entry in the FIT will not be changed.

(2) Because there are only $L$ entries in each FIT, these entries must be reused every $L$ time slots. In $F^2$BN, entries in a FIT are all set to the same initial value ('up' or 'down'). At the end of time slot $t$, the entry whose index is $EDT_j(t)$ in $FIT_{i,j}$ should be reinitialized.

Now, the behavior of each SE in each time slot can be summarized as follows:

(1) Receiving new arrival cells.

(2) Checking FIT, forwarding cells, and updating FIT.

(3) Reinitializing FIT.

The details of step (2) and (3) are described by pseudocode in Fig. 5. It can be seen that only one checking operation, two update operations and one reinitializing operation are needed in each time slot at each SE.

*2) The Initial Value and Priority Assignment in $F^2$BN*

In $F^2$BN, there must be an initial value assignment for each entry of FITs to perform reset, and a priority assignment for each SE to perform the *bump* when two cells arrive at a SE simultaneously.

We divide banyan network into interconnect-groups (ICG). Each group is composed of four SEs, which are interconnected by four wires. As Fig. 6 shows, $SE_{0,0}$, $SE_{0,1}$, $SE_{2,0}$, and $SE_{2,1}$ compose a single ICG. Here, we use the position of the top left SE to mark an ICG. For example, the ICG in Fig. 6 is marked as $ICG_{0,0}$.

In each ICG two left SEs share the inlets of two right SEs. In order to balance the traffic load between the two right SEs, it's better to assign different initial values to the two left SEs. The initial value assignment is explained with pseudocode in Fig. 7, and an example of initial value assignment is shown in Fig. 6.

In the SE, when a bump happens, one of the two cells must be shifted to another outlet according to the cells' priorities. There are several kinds of methods to assign priorities, such as: high priority to upper inlet, high priority to lower inlet, or high priority to random inlet. The first two methods will cause unfairness, and the random method is difficult to implement in hardware. So we assign the priorities still based on ICG, just like the method used in initial value assignment. The priority assignment is described with pseudocode in Fig. 8, and an example is shown in Fig. 6, where the upper inlet of $SE_{0,1}$ and the lower inlet of $SE_{2,1}$ have high priority, the other two inlets of $SE_{0,1}$ and $SE_{2,1}$ have low priority.

*3) Simple Flip-Flop Banyan Network*

The RANDBN is costly to implement in hardware for the use of random, we can just use a simple version of Flip-Flop Banyan Network ($SF^2$BN) to mimic a RANDBN. $SF^2$BN has almost no difference with $F^2$BN, except that there is only one entry in each FIT, which is shared by cells with different departure times. In $SF^2$BN, there is no need to reinitialize FIT.

**Checking FIT, forwarding cell, and updating FIT**
At time $t$, for all $SE_{i,j}$ in parallel
  Upon cells arrival
  Case: one cell $c_1$ arrives with $DT=t_1$
    Check the $t_1$-th entry in FIT, and return $direct_1$
    Forward $c_1$ to $direct_1$
    Reverse the $t_1$-th entry in FIT
  Case: two cell $c_1$, $c_2$ arrive with $DT=t_1$, $t_2$ respective, and whose priorities are $pri_1$, $pri_2$
    If $pri_1>pri_2$
      Check $t_1$-th entry in FIT, and return $direct_1$
      Forward $c_1$ to $direct_1$
      Reverse the $t_1$-th entry in FIT
      Forward $c_2$ to another direction
      Update the $t_2$-th entry in FIT to $direct_1$
    Else
      Check $t_2$-th entry in FIT, and return $direct_2$
      Forward $c_2$ to $direct_2$
      Reverse the $t_2$-th entry in FIT
      Forward $c_1$ to another direction
      Update the $t_1$-th entry in FIT to $direct_2$

(a) Checking FIT, forwarding cell, and updating FIT.

**Reinitializing FIT**
At time $t$, for all $SE_{i,j}$ in parallel
  Set the $EDT_j(t)$-th entry in $FIT_{i,j}$ to initial value

(b) Reinitializing FIT

Fig. 5. Pseudocode for the behavior of each SE in $F^2$BN.

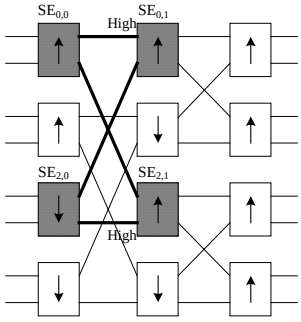


Fig. 6. An interconnect-group in an $8\times8$ banyan network.

**Initial Value Assignment**
From phase 0 to phase $\log_2 N-1$
For all ICGs in this phase
  Let the initial value of the top left SE be 'up'
  Let the initial value of the bottom left SE be 'down'
For all SEs in phase $\log_2 N$
  Let the initial value of the SE be 'up'

Fig. 7. Pseudocode for initial value assignment.

```
Priority Assignment
From phase 0 to phase  log2 N − 1
    For all ICGs in this phase
        To the top right SE
            Give high priority to the upper inlet
            Give low priority to the lower inlet
        To the bottom right SE
            Give high priority to the lower inlet
            Give low priority to the upper inlet
For all SEs in phase 0
    Give high priority to the upper inlet
    The lower inlet is left idle
```

Fig. 8. Pseudocode for priority assignment.

## IV. Performance Evaluation

Here we analyze the cell loss rate in RANDBN. It is difficult to give the analytic solution of cell loss rate of F$^2$BN currently, so we evaluate its performance through simulations.

### A. Theoretical Analyses

Since the state of each SE in the RANDBN is selected randomly, the final destination of a certain cell can be regarded to be distributed uniformly among all of the shared memories. Let $R$ be the probability of a cell reaching one certain memory $M_i$. For there are $2N$ memories, $R = \frac{1}{2N}$.

Consider there will be at most $N$ cells with the same departure time, we present the ball-bin model, in which each of the $N$ cells is placed in any of the $2N$ independent memories with equal probability of $R$.

Let $\Pr(M_i = 0)$ be the probability that $M_i$ is empty, and $\Pr(M_i = 1)$ be the probability that $M_i$ is nonempty. Then the average number of nonempty memories is:

$$\text{E(nonempty memories)}=\text{E}(\sum_{i=0}^{2N-1}\Pr(M_i=1))=2N\cdot\Pr(M_i=1).$$

Since $\Pr(M_i = 0) = (1-\frac{1}{2N})^N$ and

$\Pr(M_i = 1) = 1 - \Pr(M_i = 0)$,

$$\text{E(nonempty memories)}= 2N\cdot[1-(1-\frac{1}{2N})^N]. \quad (7)$$

The average number of conflicts is:

$$\text{E(conflicts)}= N - \text{E(nonempty memories)}$$
$$= N - 2N\cdot[1-(1-\frac{1}{2N})^N]. \quad (8)$$

Let $c$ be the probability of conflict, then

$$c = \frac{\text{E(conflicts)}}{N} = 1-2\cdot(1-(1-\frac{1}{2N})^N). \quad (9)$$

When $N \to \infty$, $c = 2e^{-\frac{1}{2}} - 1 \approx 21.31\%$.

### B. Experiments

In this subsection, the performance of the banyan network is evaluated in a series of simulations. Since BMC is to emulate OQ switches, we focus on the metric of cell loss rate. There are two factors that might cause a cell loss in BMC. Firstly, bump may occur when two cells select paths on a same SE, and one cell may not be able to select its preferred path. So when a cell reaches the memory, another cell with the same DT may already exist. In this situation, the new arrival cell will be dropped. Secondly, the queue length of the shadow OQ switch is finite, so there's unavoidable congestion caused by multi-cells simultaneously arriving at different inputs addressing to the same output, then drop appears. Much work has been done on the second situation to address the cell loss problem in OQ[1]. Here we just study the cell loss caused by the first factor.

In simulations, two typical traffic patterns are used: Bernoulli i.i.d. uniform traffic and bursty traffic. The simulation progress lasts 1000000 time slots when $N \le 256$, and 200000 time slots when $N \ge 512$.

#### 1) Performance of RANDBN and SF$^2$BN

We first make the study of RANDBN and SF$^2$BN. Denoting $\rho$ as the offered load, we observe the relationship between cell loss rate $c$ and $\rho$ under Bernoulli i.i.d. uniform traffic, with $N$ to be 32 and 1024. In RANDBN, cell loss rate increases linearly as $\rho$ increases, which can be seen from the upper two lines in Fig. 9. The lower two lines depict the case in SF$^2$BN, where cell loss rate increases linearly with $\rho$ above 0.5, and when $\rho$ is below 0.5, the increase becomes much slower. Generally speaking, SF$^2$BN is always better than RANDBN, because using SF$^2$BN the cell distributing can be recorded to a certain extent, and through changing the state of SE, load balancing can be achieved. Fig. 10 shows the relationship between cell loss rate and $N$ in SF$^2$BN and RANDBN when $\rho$ is 1.0. The performances of these two methods are quite close. We can see the simulation results match the theoretic conclusion very well. When $\rho$ is 1.0 and $N$ approaches infinite, both of the two methods can reach the throughput of 78.69%, and the probability of cell conflict is 21.31%.

#### 2) Performance of F$^2$BN

##### a) Bernoulli uniform i.i.d. Traffic

As depicted in Fig. 11, cell loss rate in F$^2$BN is plotted as a function of $\rho$ with various values of $N$. When $\rho$ is 1.0, the cell loss rate is between $10^{-2}$ and $10^{-3}$. When $\rho$ decreases, loss rate drops very quickly. With $\rho$ below 0.5, there is no cell loss during the simulation.

On the other hand, with the same value of $\rho$, cell loss rate decreases as $N$ increases. The reason for such phenomenon is that, anytime with a higher $N$, the ratio between compatible and unavailable memory number is higher, so that the probability for cell to reach compatible memory is larger.

In F$^2$BN, the cell loss rate is always lower than 1%. This strongly proves that the memorial capability of SE has an effective influence on load balancing cells with the same departure time.

##### b) Bursty Traffic

We use the same bursty traffic model as in [12], and use two parameters to describe the traffic character: the mean burst length $b$ and the offered load $\rho$. The probabilities that an active or an idle period will end at a time slot are fixed, which are denoted as $p$ and $q$ respectively. We can express the mean burst length and the offered load as follows.

$$b = \sum_{i=1}^{\infty} ip(1-p)^{i-1} = \frac{1}{p}, \tag{10}$$

$$\rho = \frac{\frac{1}{p}}{\frac{1}{p} + \sum_{i=0}^{\infty} iq(1-q)^{i}} = \frac{q}{q+p-pq}. \tag{11}$$

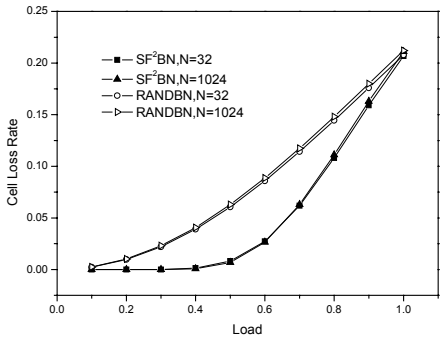


Fig. 9. Cell loss rate in SF$^2$BN and RANDBN under Bernoulli i.i.d. uniform traffic and various values of $\rho$, $N$=32, 1024.

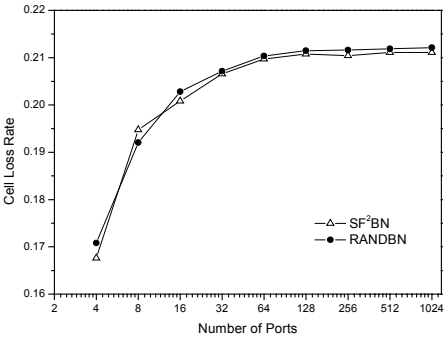


Fig. 10. Cell loss rate in SF$^2$BN and RANDBN under Bernoulli i.i.d. uniform traffic where $\rho$=1.0.

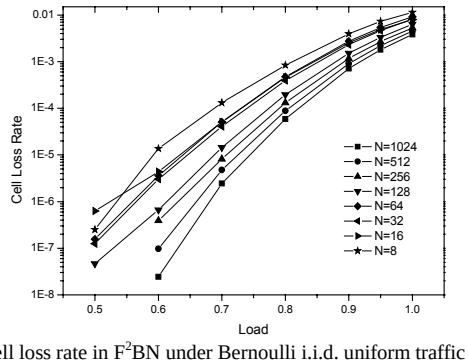


Fig. 11. Cell loss rate in F$^2$BN under Bernoulli i.i.d. uniform traffic and various values of $\rho$.

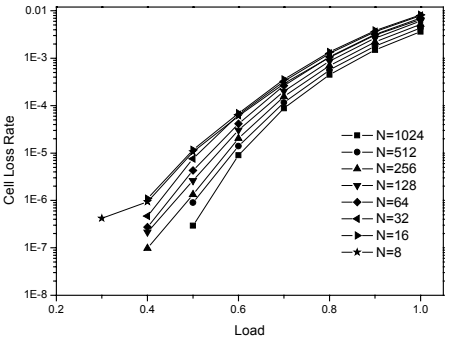


Fig. 12. Cell loss rate in F$^2$BN under the bursty traffic ($b$=16).

With $b$ to be 16, Fig. 12 depicts the relationship between cell loss rate and $\rho$ when $N$ varies. The loss rate is appreciably greater than that under Bernoulli traffic, and the trends are similar. When $\rho$ is 1.0, cell loss rate is still below 1%.

## V. Conclusions

In this paper, we present a new kind of structures using banyan network to fulfill the non-conflicted cell distributing in OQ emulated two-stage switches. Three distribute banyan networks, named RANDBN, SF$^2$BN, and F$^2$BN, are proposed respectively. By recording the outlets of cells with different departure times in SE, load balance of the cells with the same departure time among memories can be perfectly achieved by F$^2$BN. The hardware implementation is quite simple, with just one checking operation for each cell to select a path at each SE.

The performance of F$^2$BN is well enough to achieve 99% throughput under both Bernoulli i.i.d. uniform and bursty traffic. Because BMC is an OQ emulated switch architecture, it can predict and guarantee the cell delay (which equals the delay in OQ switch adding a constant of $\log_2 N$). These excellent features make BMC a quite ideal solution for multimedia communication applications.

Currently, the cell loss rate of F$^2$BN is obtained through simulations. Further work will be done to get an analytic solution of the cell loss rate of F$^2$BN.

## References


[1] M. Karol, M. Hluchyj, and S. Morgan, "Input Versus Output Queueing on a Space-Division Packet Switch," *IEEE Transactions on Communications*, vol. 35, pp. 1347-1356, 1987.

[2] Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," ACM SIGCOMM Computer Communication Review, vol. 19, pp. 1-12, 1989.

[3] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," *IEEE/ACM Transactions on Networking,* vol. 1, pp. 344-357, 1993.

[4] B. Prabhakar and N. McKeown, "On the speedup required for combined *input* and output queued switching," *Automatica*, vol. 35, pp. 1909–1920, Dec. 1999.

[5] S.-T. *Chuang*, A. Goel, N. McKeown, and B. Prabhakar, "Matching output queueing with a combined input/output-queued switch," *IEEE Journal on Selected Areas in Communications,* vol. 17, pp. 1030-1039, 1999.

[6] I. Stoica and H. Zhang, "Exact emulation of an output queueing switch by a *combined* input output queueing switch," in *Proc. IWQoS*, 1998, pp. 218-224.

[7] A. *Prakash*, S. Sharif, and A. Aziz, "An O($\log^2 N$) parallel algorithm for output queuing," in *Proc. IEEE INFOCOM*, 2002, pp. 1623-1629 vol.3.

[8] A. Aziz, A. Prakash, and V. Ramachandran, "A near optimal scheduler for switch-memory-switch routers," in *Proc. Fifteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, 2003, pp. 343-352.

[9] A. *Prakash*, A. Aziz, and V. Ramachandran, "Randomized parallel schedulers for switch-memory-switch routers: Analysis and numerical studies," in *Proc. IEEE INFOCOM 2004*, pp. 2026-2037.

[10] S. Iyer, R. *Zhang*, and N. McKeown, "Routers with a single stage of buffering," *Computer Communication Review,* vol. 32, pp. 251-264, Oct 2002.

[11] Y. Xu, B. Wu, W. Li, B. Liu, "A scalable scheduling algorithm to avoid conflicts in switch-memory-switch routers," *IEEE ICCCN 2005*.

[12] H. J. Chao, "Saturn: A terabit packet switch using dual round-robin," *IEEE Communications Magazine*, vol. 8, No. 12, pp. 78-84, Dec. 2000.

[13] L. R. Goke and G, J. Lipovski, "Banyan networks for partitioning multiprocessor *systems*," *Proc. 1st Annu. Int. Symp. Comput. Architecture*, pp.21-28, Dec. 1973.