Fault tolerance in Clos–Knockout multicast ATM switch

K. S. Chan¹,†, Sammy Chan²,*,‡ and K. T. Ko²,§

¹ Department of Electrical and Electronic Engineering, The University of Hong Kong, Pokfulam Road, Hong Kong
² Department of Electronic Engineering, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong

SUMMARY

In this paper, we propose a new architecture for multicast ATM switches with fault tolerant capability based on the Clos–Knockout switch. In the new architecture, each stage has one more redundant switch module. If one switch module is faulty, the redundant module would replace the faulty one. On the other hand, under the fault-free condition, the redundant modules in the second and third stages will provide additional alternative internal paths, and hence improve the performance. The performance analysis shows that the cell loss probability is lower than the original architecture when all modules are fault free, and the reliability of the original architecture is improved. Copyright © 2002 John Wiley & Sons, Ltd.

KEY WORDS: ATM switch; fault tolerant; multicast

1. INTRODUCTION

In the past decade, ATM switches (including unicast and multicast switches) have been a hot research area. Many new switch architectures have been proposed. Most of these studies targeted to achieve high throughput, low cell loss, low delay and delay deviation, as well as scalability. Now the switch architectures are moving from research to production stage, which has brought some issues which were not the main focus of the earlier research stage. Among these, one of the most important issues is system reliability and fault tolerance.

To provide fault tolerance for a switch, two approaches are commonly used: time-redundancy and space redundancy. In the time-redundancy method, the operation speed is much higher than the external speed, and output ports are connected back to input ports. If a cell cannot be sent to the destined output port directly from the input port, due to some faults, it would then be sent to an intermediate output port, and fed back to an input port. In the next internal time slot, the cell would be sent again to the destined output. Because the external speed is already very high in the

*Correspondence to: S. Chan, Department of Electronic Engineering, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong.
†E-mail: kschan@eee.hku.hk
‡E-mail: eeschan@cityu.edu.hk
§E-mail: eektko@cityu.edu.hk

Contract/grant sponsor: City University of Hong Kong; contract/grant number: 7001191

Received October 2000
Revised June 2002
Accepted August 2002

Copyright © 2002 John Wiley & Sons, Ltd.
ATM environment, this method requires much higher internal speed still and is thus difficult to implement.

The current fault tolerance researches focus on space redundancy. In space redundancy, multiple paths are provided from an input port to each output port by using redundant hardware—extra stages of switch modules, extra switch modules in each stage and/or extra links in each switch module. In the presence of faults, an alternate path can be chosen to provide the connection. For example, in the extra-stage Cube (ESC) network [1] and multipath multistage interconnection network (MMIN) [2], some extra stages are added to the original networks to provide alternative paths and the added stages can distribute traffic. But the extra stages would increase the delay and if no internal buffer is used, the throughput would drop quickly as the number of stages increases. And if there are faults in the routing part, the performance would degrade quickly. So, it is desirable to keep the number of stages as small as possible. In Reference [3], the author proposed a fault-tolerant switch by adding one more link between each input and the switch fabric and between each output and switching fabric. So if the internal path between a pair of input–output of the switch is faulty, the traffic can choose another input to output path. In this method, the performance of the switch under the fault-free condition is better than the original architecture as the incoming cell can be distributed between two connected switch input ports. But, as in the case of the extra stage switch, if one switch module within the switch is faulty, the performance would degrade very quickly. In other words, the performances of the switch under the tolerable faulty state and the fault free state differ significantly. This is a phenomenon which we would not like to see in a fault tolerant switch. We certainly hope the performances are similar in fault free and tolerable faulty states of a fault tolerant switch.

In Reference [4], the author proposed a fault-tolerant ATM switch based on an Banyan network. Within the banyan structure, some redundant switch modules are placed for fault tolerance. Also, the switch modules are modified to $3 \times 3$ switches. In the fault-free state, the redundant switch modules can provide alternative paths for contention-lost cells, so the performance would be better than the original Banyan networks. In faulty states, the performance is similar to the original Banyan networks. But the disadvantage of this architecture is that cells may come to the output out of sequence. So, at the destination, a re-sequencer is required, and the cell choosing the alternative path experiences longer delay.

A fault-tolerant switch architecture called FAUST is proposed in Reference [5]. It placed redundant modules in each stage and redundant slices in each switch module. So the performances in fault free and faulty states are the same. However, as the redundant modules and slices are only in hot stand-by state, these redundant hardwares normally do not involve in traffic switching. This leads to resource wastage, especially in ATM environment. It is very likely that additional alternative paths are required temporarily due to the effect of statistical multiplexing in ATM environment. Therefore, if the redundant modules can be used during normal operation, the cell loss and delay performance can be improved.

Here we propose a new fault-tolerant multicast ATM switch architecture that has the advantages of the above two switches. It is based on the multicast switch called Clos–Knockout that we proposed in Reference [6]. In each stage, we place a redundant switch module for fault-tolerant purpose. As in Reference [5], there is no re-sequencing problem. Under the fault-free
condition, the redundant modules would provide alternative paths for those cells which have lost contention, so the performance is better than the original Clos–Knockout switch. The remainder of this paper is organized in this way: in Section 2, we briefly describe the architecture and principle of Clos–Knockout switch first; the modified architecture is introduced in Section 3; in Section 4, we explain the operation principle of the switch; in Section 5, we analyse the performance of the proposed switch in terms of both the cell loss probability and switch reliability. Finally, we conclude this paper in Section 6.

2. CLOS–KNOCKOUT SWITCH

Figure 1 shows the architecture of the $N \times N$ Clos–Knockout switch ($N = n^2$). It consists of three stages: the first stage is a distribution stage; the second and third stages are responsible for packet replication and switching. The structure of the switches in the second and third stages are shown in Figures 2 and 3, respectively, each consists of $n$ switch sub-modules labelled from 0 to $n - 1$. We also label the switch modules in each stage from 0 to $n - 1$, and each input/output of each module from 0 to $n - 1$.

The operation of switch modules in the second and third stages is as follows. Cells arriving to a switch module contain $n$-bit routing headers. Each bit corresponds to an output port. If the cell is destined for output port $i$, then the $(i + 1)$th least significant bit in the $n$-bit routing header

![Figure 1. The architecture of a Clos–Knockout multicast switch.](image-url)
is 1, otherwise it is 0. Having entered the switch module, the cell is first processed by the multicast pattern masker (MPM) in each sub-module. In sub-module \(i\), for example, only the \(i\)th bit preserves its original value after the processing, while all other bits become 0. Similarly, the Address Broadcaster (AB) of sub-module \(i\) will create empty cells containing routing headers with the \(i\)th bit set to be 1 and other bits set to be 0. Using these empty cells, the array of switch elements can filter out cells destined to other output ports of other sub-modules. At the same time, it will retain at most \(L_1\) cells destined to its output port in each time slot. The filtering process is as follows. Each switch element can only be in one of the two states: cross or toggled. When the routing headers of two input cells match, the switch element will be in the toggled

---

state. On the other hand, if the headers do not match, the switch element will be in the cross state. Therefore, in each sub-module, cells destined to output ports of other sub-modules will travel in the horizontal direction and eventually leave the sub-module.

Now we give an example to show how the switch operates. Assume the switch size is $64 \times 64$, $n = 8$. A cell destined to a set of output ports $D = \{ (0, 3), (0, 4), (3, 0) \}$ comes to input port $(0, 0)$. This packet should be delivered to output ports 3 and 4 of output module 0, and output port 0 of output module 3, respectively. We should decompose the destination addresses into binary sequences first. We can use three subsets to represent the output address set. The first subset $D_0 = \{0, 3\}$ identifies the output modules containing the destined output ports, while the other two subsets, $D_1 = \{3, 4\}$, $D_2 = \{0\}$ denote the destined output ports in each destined output module. Furthermore, we can use three 8-bit binary sequences (in the format $b^7, b^6, b^5, b^4, b^3, b^2, b^1, b^0$) to represent these subsets. We use $B_0 = \{00001001\}$ to represent $D_0$ where bits 0 and 3 are set 1 as 0 and 3 are included in subset $D_0$. Similarly, we use $B_1 = \{00010000\}$ and $B_2 = \{00001001\}$ to represent $D_1$ and $D_2$, respectively. These binary sequences are used in different switch modules as routing headers, which will be explained later.

After the distribution stage, assume the cell comes to module 0 of the second stage. A routing header $B_0 = \{00001001\}$ is assigned to the cell which will lead copies of this packet to output modules 0 and 3. Then after the MPM of the first sub-module, the header becomes $B_0^2 = \{00000001\}$. The header of the created empty cells by the AB of the first sub-module is also $\{00000001\}$. So the cell will come to the output port of the first sub-module. Similarly, the cell will come to the output port of the fourth sub-module. After the process of other sub-modules’ MPMs, the header of the cell will become all-0. So the cell will not appear at other sub-modules’ output ports.

Now the cell will come to modules 0 and 3 in the third stage. The copy coming to module 0 will be assigned a routing header $B_1 = \{00011000\}$ as it should be delivered to output ports 3 and 4. So the cell would appear at the output ports of sub-modules 3 and 4. The copy coming to module 3 would be assigned a routing header $B_2 = \{00000001\}$, so it will appear at the output port of sub-module 0.

The advantages of Clos-Knockout switch are: (1) it is modular and suitable for large-scale deployment; (2) no dedicated copy network is required since copying and switching are performed simultaneously; (3) two-stage packet replication is used which gives a maximum fan-out of $n^2$; (4) translation tables are distributed which gives manageable table sizes; (5) high throughput performance for both uniform and non-uniform input traffic; (6) self-routing scheme is used.

3. FAULT-TOLERANT SWITCH ARCHITECTURE

Figure 4 shows the modified $N \times N$ ($N = n^2$) switch architecture which still consists of three stages of switch modules. At each stage we add one redundant module for fault tolerance. The first stage consists of $(n + 1)$ distribution modules, each with $n$ inputs and $(n + 1)$ outputs (we also call the switch modules in stage 1 as input modules). Both the second and third stages consist of $(n + 1)$ modified knockout switches without output buffers. Note that we move the output buffers out of the third stage modules. We also label the switch modules in each stage from top to bottom as 0 to $n$, and the input and output ports of each module from top to bottom by 0 to $n - 1$ or $n$. We can label the switch’s input $i$ ($0 \leq i < N$) by $(i_a, i_b)$ where $i_a = \lfloor i/n \rfloor$ and
There are \( n \) on–off switches before the \( n \) inputs of input modules 0 and \( n \); \( n \times 2 \times 1 \) selectors before the \( n \) inputs of input modules 1 to \( n - 1 \). Again, here we use similar notation to represent an on–off switch or a selector. That is, if a selector (or an on–off switch) is connected to input module \( i \)'s input port \( j \), then the selector's (or the on–off switch's) label is \((i, j)\). The upper input of selector \((i, j)\) is connected to the switch's input \((i - 1, j)\), the lower input is connected to the switch's input \((i, j)\). The switch's input \((0, i)\) (\(0 \leq i < n\)) is connected directly to on–off switch \((0, i)\). Input \((n - 1, i)\) (\(0 \leq i < n\)) is connected to on–off switch \((n, i)\). We also label output \( o \) of the switch as \((o_a, o_b)\) where \( o_a = \lfloor o/n \rfloor \) and \( o_b = o - o_a \times n \). At each output, there is an output buffer which can simultaneously receive up to \( L \times 2 \) packets. We denote the first \( n \) output ports as output group 0, the first output group; the next \( n \) output ports as output group 1, the second output group, and so on. There is a \( 2L \times 2L \) concentrator connected to each output buffer. We use \((c_a, c_b)\) to label a concentrator. If the concentrator is related to output port \((i, j)\), then \((c_a, c_b) = (i, j)\). The first \( L \) inputs of concentrator \((i, j)\) are from the third stage module \( i \)'s output port \( j \), and the lower \( L \) inputs are from the third stage module \( n \)'s output port \( j \). There is also an on–off switch before each of the lower \( L \) inputs.

This switch architecture can tolerate at most three faulty modules, one at each stage. If a module in the first stage (stage 1), say module \( i \), is faulty, we can avoid delivering traffic to this faulty module via properly setting the on–off switches and selectors. If a module in the second stage (stage 2), say, module \( j \), is faulty, we then configure module \( n \) in stage 2 to replace module \( j \). The proper setting in the modules of stage 1 can avoid delivering traffic to module \( j \) of stage 2. Similarly, if a module in stage 3, say module \( k \) is faulty, then module \( n \) of stage 3 is properly configured to replace module \( k \). All traffic destined to output module \( k \) will then be diverted to module \( n \) in stage 3. In the following we explain the operations in more details.

Figure 5 shows the structure of the distributor in the first stage, assuming \( n = 7 \). This is a Banyan network with a connector attached to the last stage. There are seven inputs and eight outputs for this distributor. The functionalities of the distributor are different, according to whether all switch modules in the second stage are functionable or not. If all switch modules are
functionable, this distributor will uniformly distribute incoming traffic to all eight output ports. If one switch module in their second stage, say module 0, is faulty, then the distributor will uniformly distribute all incoming traffic to output ports 1–7. It should be noticed that no traffic will be delivered to the faulty module 0 of the second stage. The following paragraph will explain how we can achieve this.

To uniformly distribute traffic, each $2 \times 2$ switch element is randomly in ‘cross’ or ‘bar’ state in each time slot. The input $i$ of the connector can be connected to the output $i$ or $(i+1) \mod n+1$. By appropriately setting the connectors, we can avoid sending traffic to faulty module in the second stage. There are two operation modes for connectors: fault-free mode and faulty mode. When all modules in stage 2 are fault free, the connectors operate in fault-free mode; otherwise, the connectors operate in faulty mode. In fault-free mode, the input $i$ of the connector is connected directly to output $i$. Under faulty mode, the operation of the connectors at one time slot consists of two phases. In phase 1, the inputs try to find out their appropriate outputs to avoid sending traffic to the faulty module in stage 2; in phase 2, the inputs send packets to outputs. We now explain the setting of the connectors in faulty mode in more detail. There are two states for an input of connectors: ‘busy’ or ‘idle’. When a cell is present at the input at current time slot, the input is in ‘busy’ state; otherwise, the input is in ‘idle’ state. Assume module $k$ in stage 2 is faulty. If input $k$ of the connector is in ‘idle’ state, input $i$ of the connector is connected to output $i$ for all $0 \leq i \leq n$. As input $k$ is idle, no traffic is sent to the faulty module under this setting. If input $k$ is busy, and the subsequent $i$ inputs $k+1 \mod n+1, k+2 \mod n+1, \ldots, k+i \mod n+1$ are busy, input $k+i+1 \mod n+1$ is idle, the setting is: for $0 \leq j \leq i$, input $k+j \mod n+1$ is connected to output $k+j+1 \mod n+1$; for $i+2 \leq j \leq n$, input $k+j \mod n+1$ is connected to output $k+j \mod n+1$. As input $k+i+1 \mod n+1$ is idle, we need not consider its connecting pattern. Here we give two examples to further explain how the connector is set up when there is a faulty module in the second stage. Assume $n = 7$, module 4 in the second stage is faulty and the busy inputs of the considered connector are 0, 3, 4, 5, 6 and 7. As inputs 4, 5 (4+1) (mod 8), 6 (4+2) (mod 8), 7 (4+3) (mod 8), 0 (4+4) (mod 8) are busy, and input 1 is idle, the connecting pattern is: 4 $\rightarrow$ 5, 5 $\rightarrow$ 6, 6 $\rightarrow$ 7, 7 $\rightarrow$ 0, 0 $\rightarrow$ 1 and 3 $\rightarrow$ 3. As inputs 1, 2 are idle, we do not consider their connecting patterns here. If the busy inputs are 0, 1, 3, 6, as input 4 is idle, the connecting pattern is: 6 $\rightarrow$ 6, 0 $\rightarrow$ 0, 1 $\rightarrow$ 1, 3 $\rightarrow$ 3.
The modified structure of switch module $i$ in the second stage is shown in Figure 6. In each module, there are $n + 1$ sub-modules. The first $n$ sub-modules are the same as that in the original Clos–Knockout switch, but the last one is different. For sub-module $n$, in front of the input of each MPM, there is a $2 \times 1$ selector, one input of the selector is connected to the input of the module, another input is from the horizontal output of sub-module $i$. There are two operation modes for the last sub-module: fault-free mode and faulty mode. When all modules in stage 3 are fault free, the last sub-module operates in fault-free mode; when there is a faulty module in stage 3, the last sub-module operates in faulty mode. In fault-free mode, the address broadcaster of sub-module $n$ generates the same address as that generated by sub-module $i$, and the output of selector is connected to the lower input. Under faulty mode, if module $j$ in the third stage is faulty, the address broadcaster of the last sub-module will generate the address for output $j$, and the output of the selector is connected to the upper input. That means the last sub-module will replace sub-module $j$ when module $j$ in the third stage is faulty.

Figure 7 shows the structure of switch modules in the third stage which is very similar to that of Clos–Knockout switch, except that the output buffer is taken away. The operation of switch...
modules 0 to \( n - 1 \) is similar to the operation in Section 2, so here we only explain the operation of module \( n \). There are also two operation modes for module \( n \): fault-free mode and faulty mode. If all modules in the third stage are fault free, module \( n \) operates in fault-free mode; when one module (except module \( n \)) in stage 3 is faulty, module \( n \) operates under faulty mode. Under faulty mode, module \( n \) will replace the faulty module in stage 3. Assume module \( k \) in stage 3 is faulty. By setting the last sub-modules in stage 2 to operate in faulty mode, traffic sent to the faulty module \( k \) before is now sent to module \( n \). Module \( n \) now is set to operate in the same as that of module \( k \). The on–off switches related to output group \( k \) are also set to ’on’. So the traffic destined to output group \( k \) is now switched through module \( n \) to destinations. The operation of module \( n \) under fault-free mode is a bit complicated. As we have seen before, when all modules in the third stage are fault free, all cells coming from module 0 in the second stage to module \( n \) are destined to the first output group, the next \( L_1 \) input links carry cells destined for the second output group, and so on. To avoid cell misdelivery, here we set the MPMs in module \( n \) to be ’open’ or ’closed’ cyclically. An MPM is said to be ’open’ if it operates like MPMs in other modules in the third stage. On the other hand, an MPM is said to be ’closed’ if it sets the routing header of the cell to be all-0s. Therefore, a cell coming to a closed MPM will not be delivered to the output ports. The MPMs work as follows. At time slot 1, for example, the first \( L_1 \) MPMs of each sub-module are open, others are closed. At the same time, all on–off switches related to the first output group are set to ’on’, and others are set to ’off’. Under this setting, only cells destined to the first output group will be sent to their destinations. In time slot 2, the second \( L_1 \) MPMs of each sub-module are open, while others are closed. The on–off switches related to the second output group are set to ’on’ while others are set to ’off’. This process continues until time slot \( n \). In time slot \( n + 1 \), the cycle repeats.

4. SWITCH OPERATION

As shown in Figure 8, the operation time frame of the proposed switch consists of a testing slot and \( m \) switching slots. In the testing time slot, each module (including the input processors, but excluding the switching modules in the third stage and the output buffers) will detect faults in the next stage. According to the detection results, the switch can reconfigure automatically to exclude the faulty ones. In the subsequent switching slots, incoming traffic is switched to the corresponding output buffer by the reconfigured switch.

We explain the operations of the proposed switch in frame \( N \), as shown in Figure 8. Assume the input \((i,j)\) is connected to input module \( i \) during frame \( N - 1 \). Input module \( n \) is in hot

![Figure 8. The operation time frame of the proposed switch.](image-url)
stand-by state. The input processor related to input \((i, j)\) sends a test cell to input module \(i\) at the beginning of the testing slot. This test packet is forward error control (FEC) encoded to detect data corruption. If the input module is fault free, it can determine the received packet is error-free and replies to this test cell by returning an acknowledgement before the preset time-out. Otherwise, the input module cannot acknowledge this test cell. Therefore, the input processors can determine if there are faulty modules in the first stage according to the received feedback information and appropriately set up the selectors or on–off switches to avoid sending traffic to faulty modules. If all modules are fault free, the selectors and on–off switches are set to send traffic from \((i, j)\) to input module \(i\) and no traffic is sent to input module \(n\). This can be achieved by setting the \(n\) on–off switches \((0, j)\) \((0 \leq j \leq n - 1)\) to be ‘on’ and all selectors choosing upper inputs. The \(n\) on–off switches \((n, j)\) are set to ‘off’. As an example, if there is a faulty module in the first stage, say, module \(k\), the input processors can set up these on–off switches or selectors to avoid sending traffic to the faulty module \(k\). The setting is as follows: if \(k > 0\), on–off switch \((0, j)\) \((0 \leq j \leq n - 1)\) are set to ‘on’, otherwise, set on–off switch \((0, j)\) to ‘off’; for \(0 < i < k\), selector \((i, j)\) \((0 \leq j \leq n - 1)\) chooses lower input. Selector \((k, j)\) \((0 \leq j \leq n - 1)\) is set to be in ‘disconnect’ state (‘disconnect’ means the output of the selector does not connect to the upper input nor the lower input). For \(k < i < n\), selector \((i, j)\) \((0 \leq j \leq n - 1)\) chooses upper input. Also, on–off switch \((n, j)\) \((0 \leq j \leq n - 1)\) is set to ‘on’. According to the above setting, all traffic sent to input module \(k\) before will now be sent to input module \(k + 1\) \((i \leq k \leq n)\).

The switch modules in the first stage also detect the errors of the switch modules in the second stage during the testing slot in the same way. The feedback information will affect the setting of the connectors in the first stage. If all modules in the second stage are functionable, the connectors in stage 1 operate in fault-free mode. If there is a faulty module in the second stage, say module \(k\) is faulty, the connectors in the first stage operate in faulty mode to avoid sending traffic to this faulty module, as described in the previous section.

At the same time, the switch modules in the second stage also send test cells to stage 3 modules. If all modules in the third stage are fault free, the last sub-modules of the stage 2 switch modules operate in fault-free mode; if there is a faulty module in the third stage, modules in the second stage operate in faulty mode, as described in the previous section. Module \(n\) in the third stage also operates in the same mode.

5. PERFORMANCE ANALYSIS

In this section, we analyse the performance of the proposed switch. First, we analyse the cell loss probability; then the reliability and mean time to failure.

5.1. Cell loss probability

Here we assume the input traffic is uniform. We also assume the traffic at each input port is independent and the average cell arrival rate is \(\lambda\) cells per time slot. The mean copy number of a cell entering a switch module in the second stage is denoted as \(E[se]\), and these copies are uniformly delivered to all output ports of the module. The mean copy number of a cell entering a module in the third stage is denoted as \(E[th]\), and these copies are also uniformly destined to all output ports of the module. Using these assumptions and also neglecting cell loss in the switch, the mean output traffic of an arbitrary output port is \(\lambda E[se] E[th]\). For stability, we should have \(\lambda E[se] E[th] \leq 1\).
As the cell loss probability in the third stage is only decided by the expander $L_2$ in the third stage modules, and the parameter in our modified switch architecture is the same as that in our original Clos–Knockout switch, here we will focus on the analysis of cell loss probability in the second stage. Consider the modified switch without faulty module. The input traffic to an input port (except the last input port) of a second stage module is $n/(n+1)$. There is no traffic on the last input port, because there is no traffic coming to the last module in the first stage. Without loss of generality, let us consider the first module in the second stage. Assume the expander in the second stage is $L_1$. There are three cases need to be considered here. Case 1: Cells destined to the output group 0 and the MPMs of module $n$ in stage 3 connected to the considered module are open. These cells can be delivered to the destined outputs through module 0 and module $n$ in the third stage. Case 2: Cells destined to output group 0 and the MPMs of module $n$ in stage 3 connected to the considered module are close. These cells can only be delivered to the destined output buffers through module 0 in the third stage. Case 3: Cells destined to output group $i$ (1 ≤ $i$ ≤ $n-1$). These cells can only be delivered to destined output buffers through module $i$ in stage 3.

In case 1, there are totally $2L_1$ output lines for these considered cells, therefore the cell loss probability of these cells is

$$ P_{2\text{case 1}} = \frac{\sum_{k=2L_1+1}^{n} (k - 2L_1)A_k}{n\rho_2} $$

where

$$ \rho_2 = \frac{\lambda E[se]}{n+1} $$

and

$$ A_k = \binom{n}{k} \rho_2^k (1 - \rho_2)^{n-k} \tag{3} $$

The probability that a cell is destined for output group 0 is $1/n$, and the probability that when the cell coming to the first module in the second stage, the first $L_1$ MPMs of module $n$ in the third stage are 'open' is also $1/n$, so the probability that a cell belongs to case 1 and it will be lost is

$$ \frac{1}{n} \sum_{k=2L_1+1}^{n} (k - 2L_1)A_k \frac{1}{n\rho_2} $$

For those cells belonging to case 2, there are only $L_1$ output links available for these cells, therefore the cell loss probability is

$$ P_{2\text{case 2}} = \frac{\sum_{k=L_1+1}^{n} (k - L_1)A_k}{n\rho_2} $$

The probability that the first $L_1$ MPMs of module $n$ in the third stage are 'close' is $1 - 1/n$, so the probability that a cell belongs to case 2 and will be lost is

$$ \left(1 - \frac{1}{n}\right) \frac{1}{n} \sum_{k=L_1+1}^{n} (k - L_1)A_k \frac{1}{n\rho_2} $$
For those cells belonging to case 3, there are only $L_1$ output links available for these cells, therefore the cell loss probability is

$$P_{2\text{case 3}} = \frac{\sum_{k=L_1+1}^{n} (k - L_1)A_k}{np_2}$$

(5)

The probability that a cell is not destined to output group 0 is $1 - 1/n$. In summary, the cell loss probability in the second stage is

$$P_2 = \frac{1}{n^2} \sum_{k=2L_1+1}^{n} \frac{(k - 2L_1)A_k}{np_2} + \left(1 - \frac{1}{n^2}\right) \sum_{k=L_1+1}^{n} \frac{(k - L_1)A_k}{np_2}$$

(6)

The cell loss probability in the second stage can be derived in a similar way when some modules are faulty. For example, when one module in the first stage is faulty and all modules in the second and third stages are fault free, the cell loss probability is give by Equation (6). It can be easily shown that when only one module in the second stage is faulty, the cell loss performance is similar to that of the original Clos–Knockout switch; if only one module in the third stage is faulty, the cell loss probability is close to that given by Equation (6). The cell loss probability of the Clos–Knockout switch can be found in Reference [6].

Figure 9 plots the cell loss probability in the second stage versus the Knockout expander ratio, given $n = 32$, input traffic is 0.3 and $n = 16$, input traffic is 0.15, respectively. In this figure, we only plot the curves for the original switch and the fault tolerant switch in fault-free condition. It can be seen that when all modules are fault free, the performance is slightly better than the original switch. Note that the improvement is not very significant and this is what we really target for the fault tolerant ATM switch. To a fault tolerant ATM switch, we certainly do not want to see a big difference under fault free and tolerable faulty states. Yet, at the same time, we want to have a small improvement in performance under fault free state. Due to statistical multiplexing, the incoming traffic pattern of an ATM switch is a bit unpredictable and more bandwidth may temporarily be required. We hope the additional modules can alleviate this problem. Therefore, the performance of a fault tolerant ATM switch in fault-free state being a bit better than that in tolerable faulty states is appreciated.

Figure 9. The cell loss probability in the second stage versus expansion ratio.
5.2. Reliability

Now we analyse the reliability and the mean time to failure (MTTF). Reliability and availability analysis itself is a very complicated issue, as illustrated in Reference [7]. Here we use a simplified model. We assume that each module satisfies the exponential failure law such that the reliability of each module is \( R(t) = e^{-\lambda t} \) and no repair is carried out. We also assume the failure rate for the first stage module is \( \lambda_1 \), for the second stage module is \( \lambda_2 \), for the third stage module, \( \lambda_3 \). The system would work well if and only if at each stage, there are at least \( n \) fault-free modules.

For the first stage, at time \( t \), the probability that there are at least \( n \) fault-free modules is

\[
R_1(t) = R_0^{n+1}(t) + (n+1)R_0^n(t)(1-R(t))
\]

For the second and third stages, the expression for the reliability are similar, so the system reliability at time \( t \) is

\[
R(t) = e^{-n(\lambda_1+\lambda_2+\lambda_3)t}(n+1-ne^{-\lambda_1t})(n+1-ne^{-\lambda_2t})(n+1-ne^{-\lambda_3t})
\]

The MTTF is

\[
\text{MTTF}_F = \int_0^\infty R(t) \, dt
\]

\[
= \frac{(n+1)^3}{n(\lambda_1 + \lambda_2 + \lambda_3)} - \frac{n(n+1)^2}{(n+1)\lambda_1 + n(\lambda_2 + \lambda_3)} - \frac{n(n+1)^2}{(n+1)\lambda_2 + n(\lambda_1 + \lambda_3)} - \frac{n(n+1)^2}{(n+1)\lambda_3 + n(\lambda_1 + \lambda_2)} + \frac{(n+1)n^2}{(n+1)(\lambda_1 + \lambda_2) + n\lambda_3} + \frac{(n+1)n^2}{(n+1)(\lambda_1 + \lambda_3) + n\lambda_2} + \frac{(n+1)n^2}{(n+1)(\lambda_2 + \lambda_3) + n\lambda_1} - \frac{n^3}{(n+1)(\lambda_1 + \lambda_2 + \lambda_3)}.
\]

Now let us look at the reliability of the original Clos–Knockout switch. Here we also assume that each module satisfies the exponential failure law. The failure rate for the first stage module is \( \lambda_1^0 \), for the second stage module is \( \lambda_2^0 \), for the third stage module, \( \lambda_3^0 \). As the switch module in the original Clos–Knockout switch is simpler, these mean failure rates should also be a bit smaller. However, if we compare the second stage switch modules of the modified switch architecture with the modules in the original Clos–Knockout switch, the complexity increase is not very significant when \( n \) is not small. Therefore, for simplicity, here we assume that \( \lambda_1 = \lambda_1^0 \), \( \lambda_2 = \lambda_2^0 \), \( \lambda_3 = \lambda_3^0 \). Then the reliability for the original Clos–Knockout is

\[
R_c(t) = e^{-n(\lambda_1+\lambda_2+\lambda_3)t}
\]

The MTTF is

\[
\text{MTTF}_c = \frac{1}{n(\lambda_1 + \lambda_2 + \lambda_3)}
\]

Assume all modules have the same mean failure rate \( \lambda = 10^{-5} \). Figure 10 shows the reliability for different switch sizes versus time. It can be seen that for larger switch size, the reliability is lower, and the reliability of our proposed architecture is much better than the original Clos–Knockout switch architecture. As time increases, the difference becomes larger.
Figure 11 plots the mean time to failure versus the module size \( n \) (the reader is reminded that the actual switch size is \( n^3 \)). For all switch sizes considered, the proposed switch has much better MTTF performance.

6. CONCLUSIONS

In this paper, we proposed a fault-tolerant multicast ATM switch architecture with enhanced performance. In the new architecture, we have one redundant module at each stage for fault tolerance. If one module is faulty, the redundant module at the same stage would replace the faulty one. The proposed scheme can endure at most three faulty modules (one at each stage).
When all modules are functioning well, the redundant modules in the second and third stages would provide additional alternative paths, and therefore improve the performance slightly. Note that this slight improvement is one of the design objectives as we do not want to see a big difference in performances under fault free and tolerable faulty states. The reliability analysis shows that the new switch architecture’s reliability is much better than that of the Clos–Knockout switch.

ACKNOWLEDGEMENTS

The work described in this paper was supported by a grant from City University of Hong Kong (Project No. 7001191).

REFERENCES


AUTHORS’ BIOGRAPHIES

Dr King-Sun Chan received the B Eng and M Eng degrees in electronic engineering, from Shanghai University of Science and Technology (now Shanghai University), Shanghai, China, in 1989 and 1992, respectively, and the PhD degree in Information Technology from the City University of Hong Kong, in 1998. From 1998 to 2000, he was a Research Fellow in the Department of Electronic Engineering, City University of Hong Kong. Since December 2000, he has been a Research Assistant Professor in the Department of Electrical and Electronic Engineering, the University of Hong Kong. His research interests include mobile data networks, ad hoc network routing, wireless TCP, switching system and performance analysis.
Sammy Chan received his BE and M Eng Sc degrees in electrical engineering from the University of Melbourne, Australia, in 1988 and 1990, respectively, and a PhD degree in communication engineering from the Royal Melbourne Institute of Technology, Australia, in 1995. From 1989 to 1994, he was with Telecom Australia Research Laboratories, first as a research engineer, and between 1992 and 1994 as a senior research engineer and project leader. Since December 1994, he has been with the Department of Electronic Engineering, City University of Hong Kong, where he is currently an associate professor.

K. T. Ko was born in Hong Kong, and graduated from The University of Adelaide, South Australia with a First Class Honours B Eng degree in Electrical Engineering in 1978. With the Australian Commonwealth Postgraduate Scholarship, his PhD degree in communication engineering was completed in 1982. He joined Telecom Australia Research Laboratories in Melbourne. His main research topics included the design of telephone networks, packet switched networks and performance evaluation of Metropolitan Area Networks. In 1986, he joined the Department of Electronic Engineering of the City University of Hong Kong, and currently an Associate Professor of the same Department. In research, he is involved in the investigations of data networks, the designs and congestion control in broadband communication networks as well as multimedia applications. Currently, he is on the Working Group on E-Commerce for the Hong Kong SAR Government.