Grid-based ATM Switch Architecture: a new fault-tolerant space-division switch fabric architecture

H.S. Laskaridis[1], A.A. Veglis[2], G.I. Papadimitriou[3] and A.S. Pombortsis[4]
[1,3,4] Dept. of Informatics, School of Science
[2]Computer Lab, School of Journalism & Mass Communication
Aristotle University of Thessaloniki, 54006 Thessaloniki - Greece

Abstract

Asynchronous Transfer Mode (ATM) is the switching and multiplexing technology chosen to be used in the implementation of B-ISDN, because of its superiority in fast packet switching. However, the use of ATM switches with large number of input and output ports has been proven to be a bottleneck in wide area ATM networks. In this paper we propose a new space-division grid-based ATM switch architecture with fault tolerant characteristics.

1 Introduction

ATM is the switching and multiplexing technology chosen to be used in the implementation of B-ISDN, because of its superiority in fast packet switching. The use of ATM in Wide Area Networks has revealed the necessity of ATM switches with large number of input and output ports. However, it has become obvious that there is a bottleneck in Wide Area ATM Networks, which is not due to medium's bandwidth (fiber optic with extremely high bandwidth) but due to switches' throughput. More precisely, the bottleneck is located in the "cell switch fabric", which is the core of the ATM switch, as described in (Chen and Liu, 1995). A lot of research effort is now focused on switch fabric architectures in order to increase the throughput of the switch fabric, while keeping the complexity at a reasonable level.

ATM switch fabric architectures are divided into four categories by Chen and Liu (1995):

1. Shared-medium architectures,
2. Shared-memory architectures,
3. Fully interconnected architectures, and
4. Space-division architectures

The disadvantages of all proposed architectures belonging to the first three categories become obvious when we attempt to scale to switches with large number of input and output ports (e.g. 1024×1024). Firstly, throughput is found to be poor and secondly the hardware requirements are sometimes just not applicable (e.g. shared memory with extremely high transfer rate).

For the above-mentioned reasons, a great research effort is converged on space-division architectures. The majority of them are based on Banyan multistage interconnection network (e.g. Alimuddin *et al*., 1995; Tobagi *et al*., 1991; Urushidani, 1991; Giacopelli *et al*., 1991; Lee *et al*., 1994; Oktug *et al*., 1997), and their aim is to improve its throughput and alleviate its major disadvantage: being internally blocking.

E-mail addresses:
[1] haris@ccf.auth.gr
[2] veglis@jour.auth.gr
[3] gp@csd.auth.gr
[4] apombo@csd.auth.gr

In this paper, we propose a new space-division switch architecture, called *Grid-based ATM Switch Architecture (GASA)*. This architecture is not based on Banyan networks, and its topology is grid-based. The grid is formed by switching elements (SE's). One of the main characteristics of the proposed architecture is the number of switching elements, which is equal to the number of input and the number of output ports. Taking under consideration the fact that central routing control in large-scale ATM switches is usually proved to be another bottleneck, GASA is designed to be self-routing.

GASA also shows an excellent behavior regarding fault-tolerance, caused by the existence of multiple paths between any input-output pair. The minimal conditions under which we have disruption of communication between functional switching elements are studied, along with the necessary modifications in the switch architecture, in order to support fault-tolerant routing.

The rest of the paper is organized as follows: In section 2 the overall architecture is presented in detail. The routing algorithm, executed in each switching element, is described in Section 3. In Section 4 fault tolerance is briefly discussed. In Section 5 we present the shared-memory architecture of each SE, which can easily support any kind of priorities. The analytical model of the switch architecture is studied in Section 6, as well as simulation results.

2 The Overall Architecture

The main characteristic of GASA is the number of the necessary SE's, which is minimal, in comparison to other switching architectures. More specifically it is equal to the number of input ports and the number of output ports. Each SE is directly connected to an input module, an output module and its 2 – 4 (depending on its position in the grid) neighbor SE's. Links between SE's are bi-directional while links between SE's and input or output modules are unidirectional. Internal links' rate is equal to incoming and outgoing links' rate and no speed-up factor is deployed in the present implementation. A 16×16 switch is presented in figure 1. The number of switching elements, the number of unidirectional links and the number of hops are presented in tables 1, 2 and 3 respectively, in comparison to those of a Banyan network.

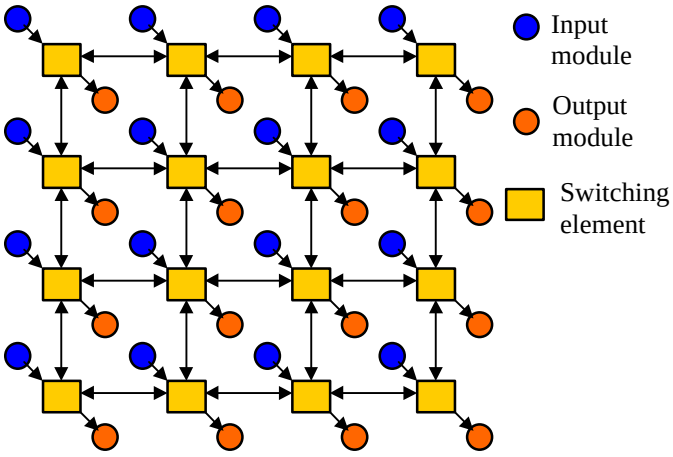

Fig. 1. A 16´16 GASA switch

Each SE has an address, a *Switching Element ID (SE ID)* which is used in routing, as explained in the following section. The SE ID is a binary number whose length (in bits) depends on the size of the switch. Addresses are assigned in a recursive way, as demonstrated in figure 2, in a way similar to the one used in the hyper-cube architecture (although the SE's are not linked in the same way).

It is also worth noting that for reasons of homogeneous implementation of the routing algorithm, a 2-bit prefix is added when expanding from N×N square switch to 2N×2N non-square switch (e.g.

from 16×16 to 32×32), although a single-bit prefix would be adequate for having each SE uniquely identified.

	16×16	32×32	64×64
Banyan $(N/2)\cdot\log_2 N$	32	80	192
GASA N	16	32	64

Table 1. Number of switching elements

	16×16	32×32	64×64
Banyan $N\cdot(\log_2 N+1)$	80	192	448
GASA $6N-4\sqrt{N}$ (square grid) or $6N-6\sqrt{\frac{N}{2}}$ (non-square grid)	80	168	352

Table 2. Number of links

	16×16	32×32	64×64
Banyan $\log_2 N$	4	5	6
GASA (average)	2.50	3.88	5.25

Table 3. Number of hops

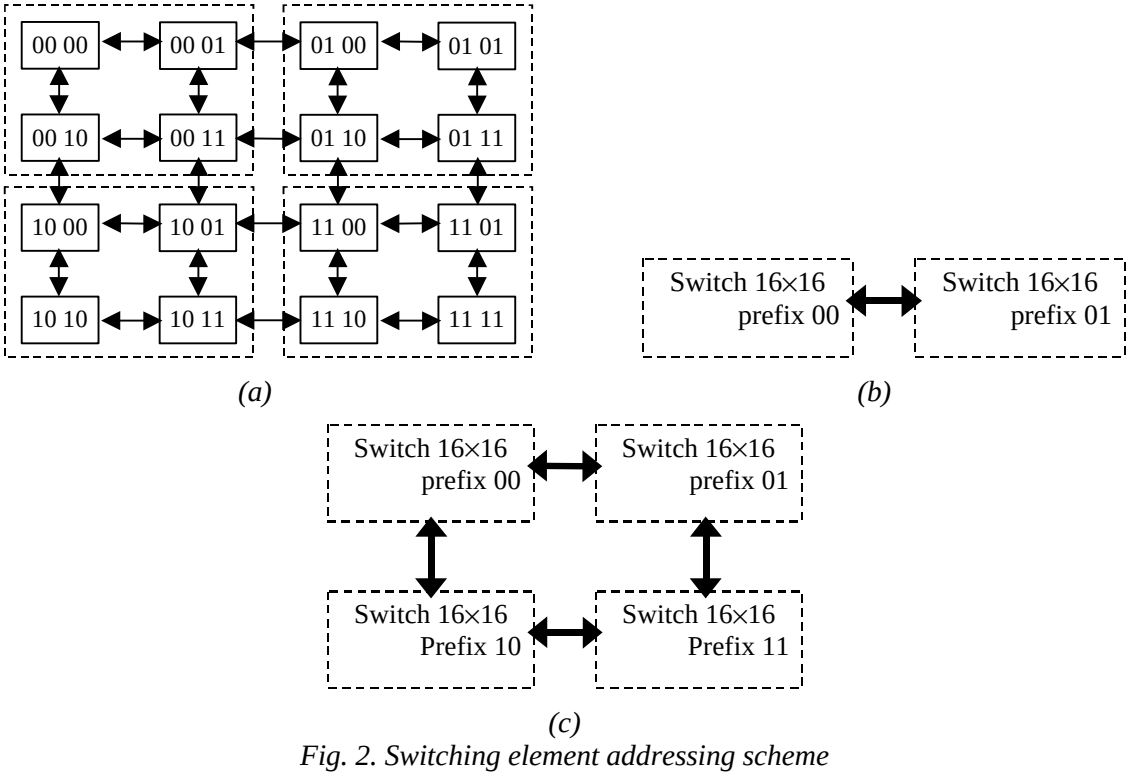

Fig. 2. Switching element addressing scheme
(a) a 16´16 switch (b) creating a non-square 32´32 switch (c) creating a square 64´64 switch

3 Routing algorithm

GASA is a self-routing architecture. Each SE is capable of routing incoming cells towards their destination, using only the destination SE ID. When a new cell arrives, the corresponding input module searches for the source VPI/VCI and port number in a lookup table, in order to find the destination output port and the corresponding destination SE ID. The destination SE ID is placed in the *tag* (header) of the cell used internally in the switch (for a complete description of the cell tag refer to Section 5).

The following routing algorithm for a 16×16 switch is executed in each SE independently of its position on the grid. In this algorithm, `SE_ID[a]` denotes the a^{th} bit of the SE ID of the current SE, `SE_ID[a,b]` denotes the a^{th} and b^{th} bits of the SE ID, and `Dest_SE_ID[a]` denotes the a^{th} bit of the SE ID of the destination SE.

```
If Dest_SE_ID[1,2] <> SE_ID[1,2] then
   Route on (1,2)
Else
   If Dest_SE_ID[3,4] <> SE_ID[3,4] then
      Route on (3,4)
   Else
      Send to output module

Procedure Route on (a,b)
If Dest_SE_ID[a] <> SE_ID[a] then
   If Dest_SE_ID[a] = 0 then
      Send to North
   Else    {Dest_addr[a]=1}
      Send to South
Else {Dest_SE_ID[a] = SE_ID[a], but Dest_SE_ID[b] <> SE_ID[b]}
   If Dest_SE_ID[b] = 0 then
      Send to West
   Else
      Send to East
```

Each cell is initially routed to the proper "quadrant" of the grid, based on the first two bits of the destination SE ID, then routed to the proper SE, based on the last two bits, and then moved to the corresponding output module.

The routing algorithm has the following characteristics:

- all cells follow *shortest paths*,
- all cells do not follow paths of the same length, but this is not of major importance because
- all cells from SE A to SE B follow *the same path*, which justifies the absence of reassembly buffers.

Next we demonstrate the expandability of the routing algorithm. The algorithm for a 32×32 or a 64×64 switch is presented:

```
If Dest_SE_ID[1,2]<>SE_ID[1,2] then
   Route on (1,2)
Else
   If Dest_SE_ID[3,4]<>SE_ID[3,4] then
      Route on (3,4)
   Else
      If Dest_SE_ID[5,6]<>SE_ID[3,4] then
         Route_on (5,6)
      Else
         Send to output module
```

(The `Route on (a,b)` procedure remains the same.)

Example: The following example demonstrates the routing process of a cell from SE 14 (11 10) to SE 0 (00 00).

SE_ID	Conditions checked	Decision
11 10	Dest_SE_ID[1,2]≠SE_ID[1,2] Dest_SE_ID[1]≠SE_ID[1] Dest_SE_ID[1]=0	Send to North
11 00	Dest_SE_ID[1,2]≠SE_ID[1,2] Dest_SE_ID[1]≠SE_ID[1] Dest_SE_ID[1]=0	Send to North
01 10	Dest_SE_ID[1,2]≠SE_ID[1,2] Dest_SE_ID[2]≠SE_ID[2] Dest_SE_ID[2]=0	Send to West
00 11	Dest_SE_ID[1,2]=SE_ID[1,2] Dest_SE_ID[3,4]≠SE_ID[3,4] Dest_SE_ID[3]≠SE_ID[3] Dest_SE_ID[3]=0	Send to North
00 01	Dest_SE_ID[1,2]=SE_ID[1,2] Dest_SE_ID[3,4]≠SE_ID[3,4] Dest_SE_ID[4]≠SE_ID[4] Dest_SE_ID[4]=0	Send to West
00 00	Dest_SE_ID[1,2]=SE_ID[1,2] Dest_SE_ID[3,4]=SE_ID[3,4]	Send to output module

The disadvantage of the architecture is that it is internally blocking. Two to four cells may require to be transmitted simultaneously from a SE over the same outgoing link. This problem can be solved by using an internal shared buffer in each SE. Manhattan Street Networks, which have similar topology but different routing algorithm, employ *deflection routing* to overcome this problem, as described and analyzed in (Choudhury and Li, 1991; Choudhury and Maxemchuk, 1991). The drawback of deflection routing is that cells from a source to a destination do not always follow the same path. Different paths that cells may follow, may have different lengths, which makes the use of reassembly buffers mandatory. This way hardware complexity increases. This is the reason why a deterministic algorithm was preferred in GASA.

4 Fault tolerance

The fact that GASA is not “single-path” architecture (i.e. there are more than one paths between each pair of SE’s) enables GASA switches to show great characteristics regarding fault tolerance. Neither additional links between SE’s nor additional SE’s are necessary, in contrast to the so-called “dilated” networks (e.g. Lee *et al.*, 1994; Itoh, 1991). However, in order to enable fault tolerance in GASA, some adjustments should be made:

1. There should be a central unit controlling and resolving faults, from now on called *"Fault Recovery Control Unit" (FRCU)*.
2. There should be bi-directional links connecting each switching element to the FRCU.
3. There should be hardware in each port of each switching element able to identify and report faulty links.
4. The routing algorithm should be adjusted, in order to be able to operate under a slightly different way when FRCU announces a fault.

The adjusted architecture is presented in figure 3 (the input and output modules are omitted for simplicity reasons).

When a fault occurs in a SE, its neighbors identify the problem by the fact that the corresponding link goes down. FRCU considers the combination of reports that it receives and concludes whether

there is a faulty link or a faulty SE. In the former case, nothing has to be done by FRCU, while in the latter case, FRCU has to "inform" all SE's to disregard cells that are destined to the faulty switching element.

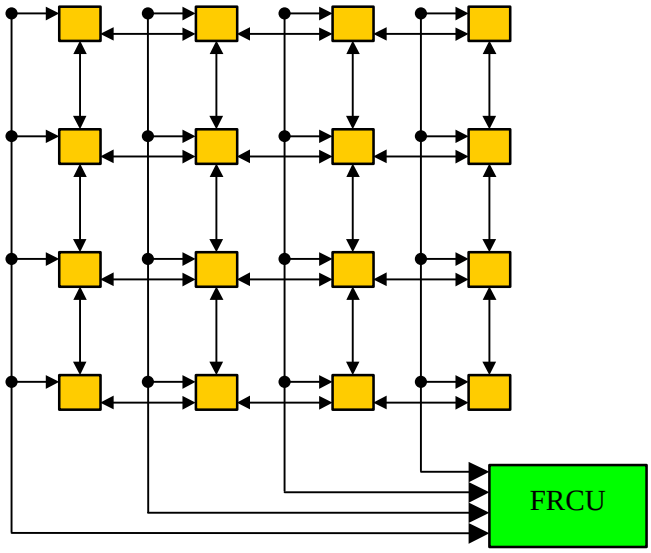

Fig. 3. Fault-tolerant adjusted GASA

If a SE discovers that one of its links is not operational, and it has not received a notification from FRCU that the corresponding neighbor is faulty, it concludes that there is only a link fault. Cells that should normally be routed over the faulty link, are sent to some other output port (different from the one that the cells arrived from).

Considering that each switching element has at least two neighbors, we conclude that the *minimal conditions* (i.e. worst case scenarios) under which there is disruption of communication between operational switching elements SE_x and SE_y are the following:

- Fault in 2 links: source switching element SE_x is in grid 's corner and both outgoing links are down, or destination switching element SE_y is in grid 's corner and both incoming links are down.
  Probability of this scenario to occur:
  $$\frac{2}{\binom{\text{total \# links}}{2}} \cdot \text{prob (link failure)}^{\,2}$$
- Fault in 2 switching elements: source switching element SE_x or destination switching element SE_y is in grid 's corner and both neighbor switching elements are down.
  Probability of this scenario to occur:
  $$\frac{2}{\binom{N}{2}} \cdot \text{prob (SE failure)}^{\,2}$$

If the SE's under study are not in grid 's corners, then 3 or 4 links and/or neighbor switching elements should go down in order to have disruption of communication.

5 Switching Element Architecture

Although the number of SE's in GASA is significantly smaller than the corresponding in Banyan networks, the hardware complexity is quite higher. There are SE's with 2, 3 or 4 neighbor SE's. However the architecture deployed should be similar in all SE's, independently of the number of

inputs/outputs ports. For this reason, a shared memory architecture is deployed. Specifically, in (Hashemi *et al.*, 1997) a shared-buffer switch architecture is proposed, which is deployed in our switch.

The disadvantage of using a shared-buffer architecture for an n×n switch is that the shared buffer has to operate at a rate n times higher than each link rate. In other words the throughput of the shared buffer has to be n times greater than each link rate. However, this is not a major problem in our switch, because $3 \leq n \leq 5$.

On the other hand the advantage of using a shared buffer architecture in contrast to using separate queues (one for each output port) is that we have better memory utilization. In other words, we need less memory in order to obtain the same cell loss probability. Next the operation of the shared queue is described.

Each queue consists of groups of cells. Each group is consisted of cells that will be transmitted over different outgoing links during the same timeslot. Cells entering the SE enter the queue from the head of the queue and start searching for their position (figure 4). Each cell is placed in the end of the first group that does not have a cell for the same outgoing link and makes all cells buffered in the following slots to move one slot backwards.

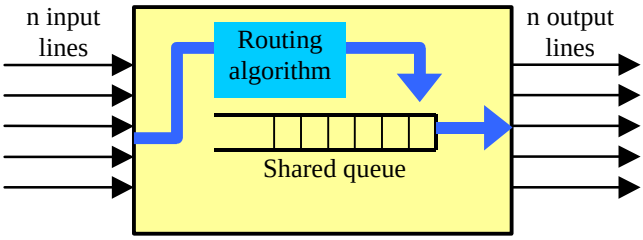

Fig. 4. Shared-buffer switching element architecture

If priorities are used, then each incoming cell is not simply looking for the first group with no other cell trying to use the same outgoing link, but its *Priority Control* field (figure 5) is also compared to the corresponding field of already buffered cells that will use the same outgoing link. If the new cell has higher priority than the buffered cell, the new cell replaces the old one, which starts searching for a new position moving backwards. Otherwise, the new cell keeps searching its position in the buffer.

In order for the above scheme to operate, the use of a tag is necessary. The internal structure of a cell is presented in figure 5. The cell tag is inserted by the input module, when the cell enters the switch. It has 4 fields. The values of the first two fields, *"Destination SE ID"* and *"Priority Control"*, are assigned only once by the input modules. On the other hand, the values of the third and fourth field are updated by each SE and are used in the operation of the SE. The *"Output port ID"* field is set by the routing algorithm, based on the next SE the cell has to move to (north, south, west, east, or output module). The *"End of group"* field is set by the queue control to delimit each group.

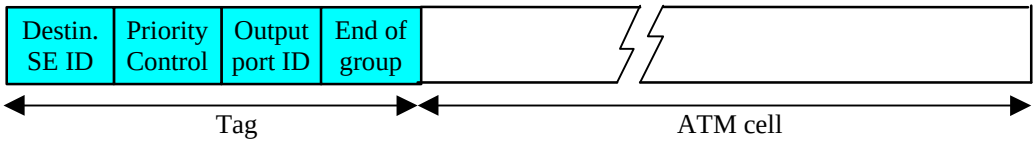

Fig. 5. Internal cell structure

It is worth noting that there are several ways that the "Priority Control" field could be used. For example the value assigned by the input module could be based on the Quality of Service that is requested for the corresponding virtual connection. Or it could be a function of the "age" of the cell in the switch (which means that each SE should be able to update this field). The field could also be split into two sub-fields in order to combine the two priority control schemes. However, priority control

schemes will not be further discussed in this paper. In the following we assume that all cells are of equal priority.

6 Performance analysis

6.1 Analytical model

The analytical model is based on the analysis presented in (Valdimarsson, 1995). The following assumptions concerning the operation of the switch are made:

1. Cells' arrivals from input x to output y follow a Bernoulli distribution with rate r_{xy}. No assumption about uniform distribution of incoming traffic to the outputs is made.
2. During a timeslot 0 to n cells arrive in each SE from its incoming links, and 0 to n cells depart from the SE over its outgoing links.
3. No "grant" signals are exchanged between the SE's prior to cell transmissions. An arriving cell that cannot find space in the shared buffer is discarded.
4. The analytical model is a discrete time model, although technically there is nothing preventing the switch from operating asynchronously.

All the notations employed in the analysis can be found in table 4.

N	Number of inputs – Number of outputs – Number of switching elements
SE_x	Switching element with SE ID x
n_x	Number of inputs / outputs in SE_x
B	Buffer size in each SE
pl_{xy}	Path length between SE_x and SE_y
$R = [r_{xy}]$	*"Load matrix"*: the traffic load from SE_x to SE_y is r_{xy}
$L = [l_{xy}]$	*"Link load matrix"*: the traffic load over the link from SE_x to SE_y is l_{xy} $l_{xy} = 0$, if SE_x to SE_y are not neighbors $l_{xx} = r_{xx}$ (for conventional reasons)
$l_{input,x}$	Load on link connecting SE_x to its input module
$l_{x,output}$	Load on link connecting SE_x to its output module
$ó$	Current length of a shared queue
s	Current length of a virtual queue
$(s, ó)$	State of a virtual queue: virtual queue contains s cells, while there are $ó$ cells in the switching element
$ð_i(s, ó)$	State probability of virtual queue i
$ë_i(s_2, ó_2 \mid s_1, ó_1)$	Transition probability of virtual queue i, from state $(s_1, ó_1)$ to $(s_2, ó_2)$
$p_i(k_i, k)$	*"Arrival probability"*: probability that cells arriving at the SE during a timeslot will lead to transition of virtual queue i from state $(s, ó)$ to state $(s + k_i, ó + k)$, under the assumption that there are no cell departures and there is infinite buffer space (B).
$q_i(m_i, m)$	*"Departure probability"*: probability that cells departing from the SE during a timeslot will lead to transition of virtual queue i from state $(s, ó)$ to state $(s - m_i, ó - m)$, under the assumption that there are no cell arrivals.
t_x^i	Throughput of virtual queue i, which belongs to SE_x under the specific load
t_x	Total throughput of SE_x
Lpr_x	Cell loss probability of SE_x
$\bar{ı}_x$	Average queue length of SE_x

Table 4. Notations - Definitions

In order to perform our analysis we employ an equivalent model for the SE: we consider each SE having a dedicated queue per output port (from now on called "*virtual queue*"). However we do not pose any restrictions on the length of each virtual queue. We only pose restrictions on the summation of lengths of the queues belonging to the same switching element:

$$\text{ó} = \sum_{j=0}^{n_x - 1} s_j \leq B$$

This way the model remains accurate.

First of all we have to calculate the link load matrix. Each element of this matrix represents the load of a link:

$l_{xy} = \sum r_{vw}$, ∀ v, w: the path from SE_v to SE_w passes over the link (x,y) (1)

(i.e. the link from SE_x to SE_y).

$$l_{input,x} = \sum_{\forall y} r_{xy} \quad (2)$$

$$l_{x,output} = \sum_{\forall y} r_{yx} \quad (3)$$

The values of r_{xy} should be such that $l_{xy} \leq 1$, for each x, y, in order to prevend massive cell loss. This is something that the Connection Admission Control (CAC) algorithm should take care of.

We define Adj(x) as the set of switching elements that are adjacent to SE_x. The input and output modules connected to the specific switching element are also considered neighbors and included in Adj(x). We also define:

- adj(x,N): the north neighbor of SE_x
- adj(x,S): the south neighbor of SE_x
- adj(x,W): the west neighbor of SE_x
- adj(x,E): the east neighbor of SE_x

We assume that queue i in SE_x is the queue corresponding to the link (x, y). In order to calculate the probability of k_i cells arriving in queue i, we first define:

- $p_N^i = \sum r_{vw}$, ∀v,w: the path from SE_v to SE_w passes over the links (adj(x,N), x) and (x, y) - The probability of a cell arriving in queue i from the north neighbor of SE_x.
- $p_S^i = \sum r_{vw}$, ∀v,w: the path from SE_v to SE_w passes over the links (adj(x,S), x) and (x, y) - The probability of a cell arriving in queue i from the south neighbor of SE_x.
- p_W^i , p_E^i are defined in similar way, for the west and east neighbors.
- $p_I^i = \sum r_{xw}$, ∀w: the path from SE_x to SE_w passes over the link (x,y) - The probability of a cell arriving in queue i from the input module of SE_x.

The "arrival probability", i.e. the probability that k cells arrive in SE_x during a timeslot and k_i of them arrive in virtual queue i, is:

$$p_i(k_i, k) = \sum_{\substack{\forall D \subset Adj(x) \\ |D| = k}} \sum_{\substack{\forall D1 \subseteq D \\ |D1| = k_i}} \left[\prod_{\forall d1 \in D1} p_{d1}^i \cdot \prod_{\forall d \in D - D1} \left(l_{adj(x,d),x} \cdot \left(1 - p_d^i \right) \right) \cdot \prod_{\forall d' \in \overline{D}} \left(1 - l_{adj(x,d'),x} \right) \right] \quad (4)$$

where |D| is the number of elements that subset D has. So in order to calculate this probability, we consider all the combinations of k inputs sending cells (set D in eq. 4), while the rest of the inputs (|Adj(x)| - k) do not send cells (set $\overline{D}$ in eq. 4). k_i of these are destined to the virtual queue i (set D1 in eq. 4), while the rest (k - k_i) cells are destined to other virtual queues (set D - D1 in eq. 4).

Similarly, the "departure probability", the probability that m cells depart from virtual queue i of SE_x during a timeslot is:

$$q_i(0,m) = \sum_{\substack{\forall D \subset Adj(x): \\ |D|=m,\ d1 \notin D}} \left[\prod_{d \in D} l_{x,adj(x,d)} \cdot \prod_{d' \in \bar{D}} \left(1 - l_{x,adj(x,d')}\right) \right] \quad (5a)$$

$$q_i(1,m) = \sum_{\substack{\forall D \subset Adj(x): \\ |D|=m,\ d1 \in D}} \left[\prod_{d \in D} l_{x,adj(x,d)} \cdot \prod_{d' \in \bar{D}} \left(1 - l_{x,adj(x,d')}\right) \right] \quad (5b)$$

$$q_i(m_i,m) = 0,\ m_i > 1 \quad (5c)$$

where d1 is the direction in which virtual queue i transmits. In this case we consider all the combinations of m output ports transmitting cells (set D in eq. 5), while the rest of them (|Adj(x)| - m) do not transmit.

Having calculated arrival and departure probabilities, we are now able to calculate transition probabilities and state probabilities for each queue i.

$$\ddot{e}_i(s_2,\acute{o}_2 | s_1,\acute{o}_1) = \sum_{k=0}^{n_x} \sum_{m=0}^{n_x} \sum_{k_i=0}^{k} \sum_{m_i=0}^{m} p_i(k_i,k) \cdot q_i(m_i,m) \quad (6)$$

$$k, k_i, m, m_i : \acute{o}_2 = \acute{o}_1 + k - m,\ s_2 = s_1 + k_i - m_i$$

$$\eth_i(s_2,\acute{o}_2) = \sum_{\acute{o}_1=0}^{B} \sum_{s_1=0}^{\acute{o}_1} \eth_i(s_1,\acute{o}_1) \cdot \ddot{e}_i(s_2,\acute{o}_2 | s_1,\acute{o}_1) \quad (7)$$

Each queue has (B/2 + 1)(B + 1) states. The last equation represents a linear system of (B/2 + 1)(B + 1) equations with (B/2 + 1)(B + 1) variables, the state probabilities. The system can be solved after replacing one of the equations with the following equation:

$$\sum_{\acute{o}=0}^{B} \sum_{s=0}^{\acute{o}} \eth_i(s,\acute{o}) = 1 \quad (8)$$

Troughput:

The throughput of virtual queue i under the specific load matrix, is: $t_x^i = 1 - \sum_{\acute{o}=0}^{B} \eth_i(0,\acute{o})$ (9)

The total throughput of SE_x is: $t_x = \sum_{\forall i} t_x^i$ (10)

while the total throughput of the switch can be calculated over the virtual queues that correspond to output ports connected to output modules. If such queues are called i_{out} in each switching element, then: $t = \sum_{x=0}^{N-1} t_x^{i_{out}}$ (11)

Loss probability:

We define $\eth_x(\acute{o}) = \sum_{s=0}^{\acute{o}} \eth_i(s,\acute{o})$ the probability of SE_x having $\acute{o}$ cells in its shared buffer. (The summation is calculated over anyone of its virtual queues i). The cell loss probability in SE_x is:

$$Lpr_x = \sum_{\acute{o}=0}^{\hat{A}} \sum_{\substack{k=0: \\ \acute{o}+k>B}}^{n_x} \left[\eth_x(\acute{o}) \cdot \sum_{k_i=0}^{k} p_i(k_i,k) \right] \quad (12)$$

The pass probability of a cell moving from SE_x to SE_y is calculated over all switching elements belonging to the path from SE_x to SE_y: $\prod_{\forall z \in path\,(x \to y)} (1 - Lpr_z)$

So the total cell loss probability of the switch under the specific traffic pattern is the "weighted" average of loss probabilities of all paths:

$$Lpr = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \left(\frac{r_{xy}}{total\_load} \cdot \left(1 - \prod_{\forall z \in path\,(x \to y)} (1 - Lpr_z) \right) \right) \quad (13)$$

where $$total\_load = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} r_{xy} \quad (14)$$

Delay:

In order to calculate the average delay of cells, we firstly calculate the average queue length, the average path length and average throughput. The average length of SE_x 's shared queue is:

$$\bar{l}_x = \sum_{ó=0}^{B} (ó \cdot ð_x(ó)) \quad (15)$$

and the average queue length of all switching elements is: $\bar{l} = \frac{1}{N} \sum_{x=0}^{N-1} \bar{l}_x$ (16)

The average path length is also calculated as a "weighted" average, taking under consideration the traffic pattern (i.e. the load matrix): $$pl_{avg} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \frac{r_{xy}}{total\_load} \cdot pl_{xy} \quad (17)$$

where pl_{xy} is the length (i.e. number of hops) of the path from SE_x to SE_y.

The average throughput is $\bar{t} = \frac{1}{N} \sum_{x=0}^{N-1} \bar{t}_x$ (18)

Finally, average delay according to Little 's law is: $\bar{ð} = \frac{\bar{l} \cdot pl_{avg}}{\bar{t}}$ (19)

6.2 Simulation results

Based on the above analytical model it is difficult to evaluate the performance of the proposed architecture. In order to overcome this obstacle we employ a simulation tool, simulating a 16 by 16 GASA switch. Several simulations with different traffic loads were contacted. More specifically we use two loads for uniform traffic distribution with total load 0.5 and 0.65 respectively. Figure 6 presents the cell loss probability versus buffer capacity (in cell slots), and figure 7 the delay (in time slots) again versus buffer capacity. From the figures it is obvious that for uniform load 0.5, GASA appears to be working very smoothly with minimum loss of cells and also minimum delay. The increase of the uniform load causes an increase of both cell loss and delay. This can be easily explained due to the fact that many links between SE's saturate and that causes an increase in the buffer queues' length.

To further investigate the behavior of the GASA we have also run simulations for two random traffic patterns, with total loads 0.56 and 0.71 (figures 8 and 9). Checks were made to assure that $l_{xy} \leq 1$. The architecture exhibits similar performance in the case of random loads. Cell loss probability and delay are slightly higher because of the non-uniform distribution of load.

Overall the GASA architecture appears to have good performance in middle to high loads in both uniform and random loads without the use of large buffer queues. We must emphasize that the buffer size appearing in the figures is the buffer size of each SE.

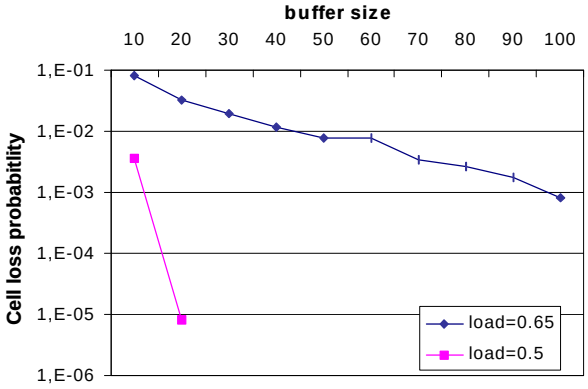

Fig. 6. Cell loss probability under uniform traffic pattern

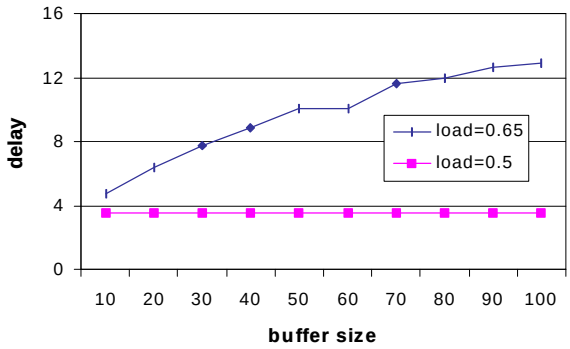

Fig. 7. Delay (in timeslots) under uniform traffic pattern

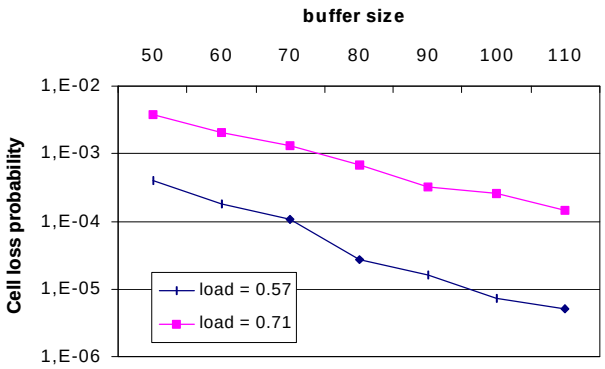

Fig. 8. Cell loss probability under non-uniform (random) traffic pattern

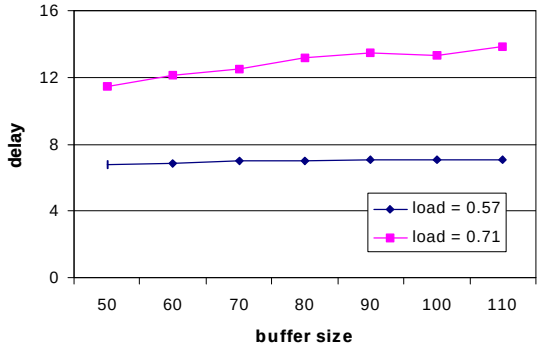

Fig. 9. Delay under non-uniform (random) traffic pattern

7 Conclusions

We have presented and studied a new grid-based ATM switch fabric architecture, with minimal number of switching elements and fault-tolerance capabilities. An analytical model was developed and the fault tolerance capabilities of the architecture were briefly discussed. The behavior of the architecture was evaluated with the help of simulation. The results indicate that the GASA architecture produces good performance without employing very large buffers. Even in the case where large buffers were employed in order to reduce cell loss probability, there was not a major increase on cell delay, while the loss probability was decreased dramatically. This behavior was exhibited in both uniform and random traffic patterns.

A future extension of this study may include the study of the architecture 's behavior under real traffic loads. As far as fault tolerance is concerned, a future study may include the development of an analytical model that will evaluate the cell availability and the reliability of the GASA architecture, as defined and analyzed in (Veglis *et al.*, 1998).

References

T. M. Chen, and S.S. Liu (1995). *ATM switching systems*, Artech House, Boston.

M. Alimuddin, H.M. Alnuweiri, and R.W. Donaldson (1995). "The Fat Banyan ATM Switch", *Proc. IEEE Infocom '95*, pp. 659-666.

F.A. Tobagi, T. Kwok, and F.M. Chiussi (1991). "Architecture, performance, and implementation of the Tandem Banyan Fast Packet Switch", *IEEE Journal on Selected Areas in Communications*, **9**, no. 8, pp. 1173-1193.

S. Urushidani (1991). "Rerouting network: a high-performance self-routing switch for B-ISDN", *IEEE Journal on Selected Areas in Communications*, **9**, no. 8, pp. 1194-1205.

J.N. Giacopelli, J.J. Hickey, W.S. Marcus, W.D. Sincoskie, and M. Littlewood (1991). "Sunshine: a high-performance self-routing broadband packet switch architecture", *IEEE Journal on Selected Areas in Communications*, **9**, no. 8, pp.1289-1298.

T.T. Lee, and S.C. Liew (1994). "Broadband packet switches based on dilated interconnection networks", *IEEE Trans. on Communications*, **42**, no. 2/3/4, pp. 732-744.

S.F. Oktug, and M.U. Caglayan (1997). "Design and performance evaluation of a Banyan network based interconnection structure for ATM switches", *IEEE Journal on Selected Areas in Communications*, **15**, no. 5, pp. 807-816.

M.R. Hashemi, and A. Leon-Garcia (1997). "The Single-Queue Switch: A Building block for switches with programmable scheduling", *IEEE Journal on Selected Areas in Communications*, **15**, no. 5, pp. 785-794.

A.K. Choudhury, and V.O.K. Li (1991). "Performance analysis of deflection routing in the Manhattan Street Network", *Proc. IEEE International Conference on Communications*, pp. 1659-1665.

A.K. Choudhury, and N.F. Maxemchuk (1991). "Effect of a finite reassembly buffer on the performance of deflection routing", *Proc. IEEE International Conference on Communications*, pp. 1637-1646.

A. Itoh (1991). "A fault-tolerant switching network for B-ISDN", *IEEE Journal on Selected Areas in Communications*, **9**, no. 8, pp. 1218 – 1226.

E. Valdimarsson (1995). "Queue analysis for shared buffer switching networks for non-uniform traffic", *Proc. IEEE INFOCOM '95*, pp. 8-15.

A. Veglis, and A. Pomportsis (1998). "Cell Availability of multiple path ATM Switch", *Proc. 9th Mediterranean Electrotechnical Conference (Melecon '98)*, pp. 1313-1317.