A High-Speed Router Featuring Minimal Delay Variation

Martin Collier
Research Institute for Networks and Communications Engineering (RINCE)
Dublin City University
Dublin 9 Ireland
collierm@rince.ie

Abstract-**This paper describes a technique for implementing the switch fabric of a high-speed router (with a throughput in excess of 600 Gb/s based on the current state of the art), with the following properties. Delay performance is virtually identical to that of a standard output-buffered switch, and the switch fabric preserves packet sequence, so that no resequencing is required for segmented packets. Clock rates are moderate except at ingress and egress points. This is achieved by distributing traffic across a number of crossbar switches operating at a low bit rate. The techniques used to resolve contention in the crossbar switches are described, and the bottlenecks limiting the capacity of the switch are discussed.**

I. Introduction

The exponential growth in Internet traffic has led to a demand for core routers of ever increasing capacity. Meeting this demand has required innovations to ensure that packets flow through the router at "wire speed". One of the major challenges is to design the switch fabric used in the router to support ever-increasing bit rates.

A typical switch fabric in today's routers segments IP packets into fixed-length cells and routes the cells using a shared-bus, output-buffered architecture. The choice of fixed-length cells simplifies the switch fabric design, and allows the use of switching chip-sets originally designed for ATM. The architecture does not scale well, and recent high-speed switching fabrics have utilised an input-buffered crossbar architecture, featuring Virtual Output Queuing so as to provide a throughput (as a fraction of link speed) similar to that of an output-buffered switch [1-2].

Such switches offer an increased throughput (in bits/sec) compared to the earlier designs because the incoming traffic is not all multiplexed onto a common bus. However, scalability is still poor, and the queuing discipline used will typically increase packet delay variation compared to output buffered schemes, as well as making it difficult to implement per-flow queuing for performance guarantees.

An architecture for the switch fabric is presented in this paper which offers higher capacity than an input-buffered crossbar switch, whilst featuring negligible buffering at the inputs. It distributed traffic across a number of switch planes, but avoids the possibility of packet re-ordering which can have an adverse impact on network performance [3].

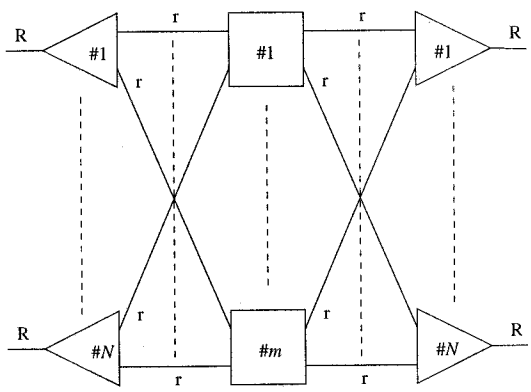

Fig. 1. The high-speed router.

II. Architecture

The switch fabric is shown in Fig. 1. It features N inputs and outputs, each operating at a bit rate R. Packets are assumed to be of fixed length. The traffic from each of the inputs is demultiplexed across m planes of switching. These planes operate at a slower bit rate of r. The relevant outputs of each plane are multiplexed together to feed the higher-speed output at rate R. The concept of using multiple switch planes like this is not new, and is, for example, being explored in the Fork/Join Router project at Stanford University [4]. Indeed, the use of such techniques, where each switch plane operates independently, is straightforward, but can give rise to performance problems at the transport layer [3].

The internal details of the input port are shown in Fig. 2. It features virtual output queuing – a routing tag associated with each incoming packet is used to demultiplex the packet onto the relevant virtual output buffer. The task of determining the routing tag, using a longest prefix match on the packet address(IP), or looking up a table indexed by a label (MPLS) or a VCI (ATM), as well as the segmentation of variable-length packets, is assumed to occur prior to entry into the switch fabric, and will not be discussed further here.

0-7803-6711-1/01/$10.00 (C) 2001 IEEE

The concentrator contains a route vector, a table of outputs, indicating from which virtual output buffer each switch plane should read a packet. Operation of the switch fabric is slotted, and the route vector is updated before each time slot. A scheduling algorithm is thus required to determine how to select the switch planes through which to route packets.

This paper addresses the practical issues raised when a central scheduling algorithm is used to assign packets to the switch planes. Such an algorithm was used by the author in his path allocation switch, in the context of ATM switching [5]. Centrally scheduled operation is normally disparaged in the context of packet switching because of the shorter periods over which routing decisions must be made. However, it will be shown that a switch of considerable capacity can be practical when using such an algorithm.

III. The Scheduling Algorithm.

The algorithm is a version of that presented in [5], modified to reduce the amount of hardware required. The scheduling engine comprises an array of $N \times m$ processors, each of which processes three quantities, viz. K_{ij}, the number of packets from input port i requesting output port j, A_{ip}, a binary quantity indicating if a packet has been scheduled from input port i via plane p and B_{pj} which similarly indicates whether plane p has been assigned a packet intended for output port j. In the typical case (as dictated by practical constraints) where $N>m$, each processor must iterate N times through an algorithm whereby it decrements K and resets A and B if $AB=1$ (otherwise leaving all quantities unchanged), then passes the K and B values to neighbouring processors, whilst retaining A. At the conclusion of the algorithm, each processor will have scheduled a packet from at most one virtual output queue.

Fig. 3 shows the internal structure of the processor. An example of a 4×4 scheduling engine is shown in Fig. 4. The arrows indicate the direction in which the updated K and B values are passed after each iteration. The array will require an additional $(N-m) \times m$ delay elements for correct operation when $N>m$. The two busses shown in Fig. 3 are used to initialise the array and to extract the routing vectors when the algorithm has concluded. Details of their operation will be given at the end of Section IV.

IV. Maximum Capacity

Practical considerations will limit the size of router which can be constructed using the above architecture. It is not scaleable in the sense of allowing an arbitrarily large-scale switch fabric to be constructed, although it is large-scale. The two primary constraints to consider are the following.

- The peak aggregate bit rate which can be supported on a parallel data bus, denoted T.
- The peak clock rate of circuitry internal to the switch fabric, denoted C.

Note that the peak throughput achievable with a shared-bus switch is essentially T assuming the use of dual-ported memory.

The implications of these constraints on the size of the switch fabric shall now be determined.

In reading data from the input ports to the virtual output buffers, we require

$$R \leq T.$$

In writing data from the virtual output buffers to the switch planes, we require

$$rm \leq T.$$

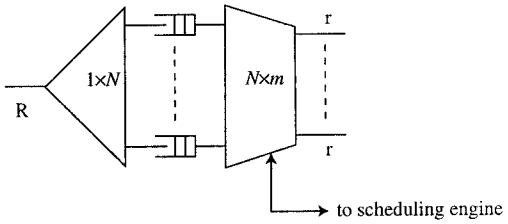

Fig. 2. Input port detail.

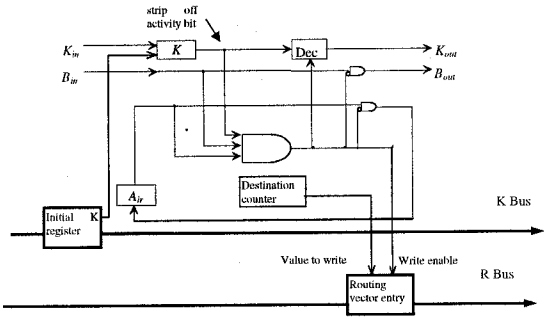

Fig. 3. Processor internals

In writing data from the switch planes to the output ports, we require

$$R \leq T.$$

We cannot proceed further without some rationale for dimensioning the switch fabric. For the moment, we shall

simply assume that the aggregate capacity of the switch planes must be double the switch capacity. We shall revisit this topic in Section VI.

This assumption gives us the additional constraint

$$Nrm \geq 2NR$$

or

$$rm \geq 2R.$$

Combining these inequalities gives us the constraint

$$T \geq rm \geq 2R.$$

Since our goal is to maximise the switch throughput, we choose the largest possible value for R. Hence

$$R=T/2. \quad (1)$$

We next consider the number of switch planes required. Each switch plane comprises an unbuffered crossbar switch. Such a switch can be built with a higher switching capacity than a shared-bus switch. Let us assume that the maximum capacity of such a switch is αT where $\alpha>1$. Then each switch plane has capacity

$$rN = \alpha T$$

so that

$$r=\alpha T/N. \quad (2)$$

(Note that, in the case where $N<\alpha$, r would exceed T. The switch fabric discussed here is inappropriate in such a case, and a single input-buffered crossbar should be used.)

Since we require $rm \geq 2R$, it follows that m is given by

$$m \geq 2R/r = TN/\alpha T$$

or (choosing the minimum value for m)

$$m = \left\lceil \frac{N}{\alpha} \right\rceil. \quad (3)$$

It remains to determine the maximum number of switch ports N. This is based on the constraints imposed by the operation of the scheduling engine.

The scheduling engine runs for N iterations (assuming $N>m$). Let us denote the number of clock cycles required by each iteration as I. The value of I depends on how the processor is implemented. The critical path through the processor is that where the K value is decremented. A bit-serial implementation will typically require $1+\lceil \log_2 N \rceil$ cycles. A bit-parallel implementation requires just 1 cycle. A serial bit-slice approach requires an intermediate number of clock cycles.

The execution time of the algorithm is thus $T_s = NI/C$ seconds. The duration of a time slot must be no less than the execution time of the scheduling algorithm[1]. For maximum capacity, we will set the two to be equal. Thus the number of bits which can be transmitted on an internal link of the switch during one time slot is

$$FL=\frac{\alpha T}{C} I. \quad (4)$$

We regard these bits as a packet comprising a frame of F cells, each with a fixed length of L bits. All the cells in a frame are routed to the same output port. Ideally $F = 1$.

Note that FL is independent of N for bit-parallel implementations of the engine.

We can now determine the number of packets which are received by the input port at peak transmission rate in one time slot. Clearly this is given by

$$P = \left\lfloor \frac{RT_s}{FL} \right\rfloor = \left\lfloor RT_s \frac{C}{\alpha T} \right\rfloor = \ldots = \left\lfloor \frac{N}{2\alpha} \right\rfloor.$$

The remaining constraints on the scheduling engine are its overall size (already discussed) and the bit rates required for input and output. Transmission of initialisation data to and results from the engine must take place within one time slot so as to avoid a pipeline stall. The request vector from each input port comprises $P(1+\lceil \log_2 N \rceil)$ bits. These are the routing tags including an activity bit for each of the possible P packets received. The route vector comprises $m(1+\lceil \log_2 N \rceil)$ bits, indicating for each of the m planes, which virtual output buffer to read. Multiplying these values by $1/T_s$ gives the required clock speed. Thus the clock rates required are C_{in} and C_{out} where

$$C_{in} = \frac{PC(1+\lceil \log_2 N \rceil)}{IN} \quad (5)$$

and

$$C_{out} = \frac{mC(1+\lceil \log_2 N \rceil)}{IN}. \quad (6)$$

[1] Otherwise the throughput of the system is compromised. However, it is acceptable for the *latency* of the scheduling engine (including the time taken to initialise the array via the request vectors and to delivery the route vectors to the input ports) to exceed one time slot, since this introduces only a constant latency in the virtual output buffers.

For a bit-serial implementation these simplify to *PC/N* and *mC/N* respectively.

There is no simple formula for determining the maximum number of switch ports. We shall consider this problem by reference to a particular example.

V. AN EXAMPLE.

Consider the constraints T = 10 Gb/s, α=4 and C = 100 MHz, which are representative of today's state of the art. We will attempt to design a switch fabric with 64 inputs and outputs.

It follows that R = 5 Gb/s, r = 625 Mb/s, m = 16, P = 8 and FL = 2800 bits (for a bit-serial implementation) or 400 bits (bit parallel). Also C_{in} and C_{out} are 12.5 (87.5) Mhz and 25 (175) MHz respectively for serial (parallel) implementation of the scheduling engine.

(In practice, the incoming rate R would be implemented by multiplexing together two STM-16 streams.)

Why resort to a bit-parallel implementation? It allows the router to elegantly switch ATM cells. Supporting ATM requires a minimum L of 424 bits. This requires a frame of 7 cells to be assembled into each packet to meet the 2800 bit minimum for a bit-serial implementation. This complicates the design of the input port (since it must accumulate ATM cells in the virtual output buffers prior to transmission through the switch fabric) and reduces throughput (if cells are transmitted in partially filled frames).

The cost of a bit-parallel implementation is that more gates are required to implement the decrement function, and that, since C_{out} exceeds C, a 2-bit bus is required to transmit the route vectors internally in the scheduling engine.

The scheduling engine requires 64 inputs and outputs, 3072 1-bit delays (comprising 64 48-bit shift registers) and 1024 processors. Each processor requires less than two hundred gates [6] even when implemented in bit-parallel form and each is essentially identical; only the initialisation of the destination counter in Fig. 4 varies among processors. An engine of such a scale should prove feasible to implement with today's technology.

The resulting switch has been simulated with a 100% offered load at each input, balanced across all output ports, assuming that successive packets have uncorrelated destinations. The throughput obtained was 100%, as expected, with a mean delay in the virtual output buffer of just 9.0×10^{-4} time slots. Thus the vast majority of packets experience no queuing delay at the input port, indicating that the switch offers virtually the same performance as a true output-buffered switch. Packet sequence preservation is guaranteed since the switch planes are unbuffered.

Since there is a low probability that a packet will be delayed at the input side of the switch, and since no buffering occurs in the crossbars, the switch offers virtually the same performance as an output-buffered switch, and hence features minimal delay variation.

VI. INCREASING THE CAPACITY, AND REDUCING THE COST.

The switch considered in Section V has a capacity of 320 Gb/s. This can be doubled without breaching the constraint on T by building two copies of the switch fabric, and multiplexing their outputs together. In Fig. 1, this would mean that the first two stages would be replicated, but the output multiplexors would merge packets from $2m$ rather than m crossbars. This is permissible since the output multiplexors in Fig. 1 operate at half the permitted rate. However, the switch is now asymmetric, with 256 STM-16 inputs and 64 STM-64 outputs. Modifying the switch to feature 64 STM-64 inputs requires demultiplexing each STM-64 stream onto *two* input ports operating at 5 Gb/s.

It is not immediately apparent that packet sequence preservation is still guaranteed in this case, since two successive packets on an STM-64 stream may be presented simultaneously to the switch fabric. However, we can use standard techniques to ensure FCFS operation of the two adjacent VOQ buffers, and, because of the highly structured way in which the routing algorithm operates, no additional hardware is required to ensure FCFS arrival at the output buffer when two successive packets are launched into the crossbars in the same time slot.

Reducing the number of switch planes obviously reduces cost. To see whether this can be done, the rationale for dimensioning the switch must be considered. This is based on a general formula for blocking in multi-rate three-stage switches presented by the author in Section 2.1 of [7]. This formula can be applied to the switch in Fig. 1 to show that that the number of switch planes required to ensure 100% throughput is upper bounded by $2P+1$. The formula is applied to the case of a multi-rate switch (of internal rate r) where each input module has P inputs, which is a valid interpretation of Fig. 1 from the perspective of the middle stage. The requirement on m is then

$$m > 2 \max_{0<u<r} \left\lfloor \frac{rP-u}{r-u} \right\rfloor$$

which reduces to $m>2P$, which is similar to, but not identical to Clos' classical result . Dimensioning on the basis of double the bandwidth in the centre stage gave

$$P = \left\lfloor \frac{N}{2\alpha} \right\rfloor$$

and

$$m = \left\lceil \frac{N}{\alpha} \right\rceil$$

which reduces to $m=2P$ or $m=2P+1$ depending on rounding.

The nonblocking condition assumes worst-case routing. The routing algorithm used here is quite efficient so the amount of hardware can be reduced by reducing m. Alternatively, reducing the number of iterations performed by the scheduling algorithm (so that an exhaustive search is no longer performed) results in less efficient routing (since a packet may not be routed even though a crossbar is available for the purpose.) but enables a larger switch to be built or the bit rate requirements in the scheduling engine to be reduced. Fig. 5 shows how the delay in the switch considered in Section V increases. As the number of scheduling algorithm iterations is reduced. The switch throughput falls below 100% if the number of iterations is less than fifty five. Above this figure, we can trade off an approximate 20% variation in the clock speed required in the scheduling engine against the buffering requirements in the virtual output buffers.

The clock speed requirements in the scheduling engine can be further reduced by precomputing a schedule, and using a small number of iterations of the algorithm to 'fill in the gaps' in the precomputed schedule. A full treatment of this approach will be the topic of a future paper.

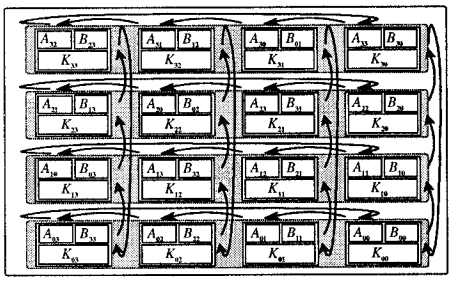

Fig. 4. The scheduling engine.

VII. Conclusions

A method for constructing a high-speed router using many switching planes of relatively low bit rate has been presented. The use of a centralised scheduling algorithm has been advocated, because it allows a switch fabric to be built whose delay performance is comparable to that of a shared-bus switch. The resulting architecture features a scheduling engine with a relatively high gate count. However, the complexity of the switching planes is greatly reduced, since they do not perform contention resolution, and a switch of less complexity would need to compromise on delay performance.

References

[1] Adisak Mekkittikul, and Nick McKeown, "A Practical Scheduling Algorithm to Achieve 100% Throughput in Input-Queued Switches," *IEEE Infocom 98*, Vol 2, pp. 792-799, San Francisco, April 1998.

[2] M. Marsan *et al.*, "Scheduling in input-queued cell-based packet switches,", *IEEE Globecom 99*, pp. 1227-1235.

[3] Jon C. R. Bennett, Craig Partridge and Nicholas Shectman, "Packet reordering is not pathological network behavior," *IEEE/ACM Trans. on Networking*, Vol. 7 , No. 6, pp. 789 – 798, Dec. 1999.

[4] "Fork/Join Router Project," http://klamath.stanford.edu/fjr/.

[5] Martin Collier, "A Three-Stage ATM Switch with Cell-Level Path Allocation", *IEEE Trans. Commun.*, vol. 45, no. 6, pp. 701-709, June 1997.

[6] Tarek Khadir and Velentin Muresan, "VHDL Modeling of a three stage ATM switch," presented at AI 2001, the International Conference on Applied Informatics, February 19-22, 2001 Innsbruck, Austria.

[7] Martin Collier and Tommy Curran, "The strictly non-blocking condition for three-stage networks", *Proceedings of the 14th International Teletraffic Congress (ITC 14)*, Antibes (France), June 6-10, 1994, pp. 635-644.

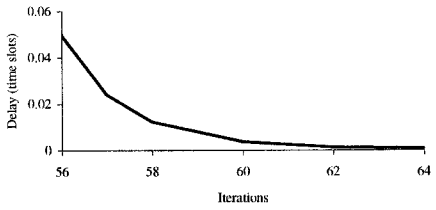

Fig. 5. Delay in virtual output buffers versus number of scheduling engine iterations.