Low jitter guaranteed-rate communications for cluster computing systems

Ted H. Szymanski* and Dave Gilbert

Department of McMaster University,
Hamilton, ON, Canada L8S 4K1
E-mail: teds@mcmaster.ca
*Corresponding author

Abstract: Low latency high bandwidth networks are key components in large scale computing systems. Existing systems use dynamic algorithms for routing and scheduling cell transmissions through switches. Due to stringent time requirements, dynamic algorithms have suboptimal performances, which limit throughputs to well below peak capacity. It is shown that Guaranteed-Rate communications can be supported over switch-based networks with 100% throughput and very low delay jitter, provided that each switch has the capacity to buffer a small number of cells per flow. An algorithm is used to reserve guaranteed bandwidth and buffer space in the switches, resulting in the specification of a doubly stochastic traffic rate matrix for each switch. Each switch schedules the Guaranteed-Rate traffic for transmission according to a resource reservation algorithm based on Recursive Fair Stochastic Matrix Decomposition. Very low delay jitters can be achieved among all simultaneous flows while simultaneously achieving 100 % throughput in each switch. When receive buffers of bounded depth are used to filter residual network jitter at the destinations, end-to-end traffic flows can be delivered with essentially zero delay jitter. The algorithm is suitable for the switch-based networks found in commercial supercomputing systems such as Fat-Trees, and for silicon Networks-on-a-Chip.

Keywords: networks; switching; scheduling; low jitter; guaranteed rate; stochastic matrix decomposition; quality of service.

Reference to this paper should be made as follows: Szymanski, T.H. and Gilbert, D. (2008) 'Low jitter guaranteed-rate communications for cluster computing systems', *Int. J. Communication Networks and Distributed Systems*, Vol. 1, No. 2, pp.140–160.

Biographical notes: Ted H. Szymanski holds the Bell Canada Chair in Data Communications at McMaster University. He completed his PhD at the University of Toronto, and has taught at Columbia and McGill Universities. From 1993 to 2003, he was an Architect in a National Research Programme on Photonic Systems funded by the Canadian Networks of Centers of Excellence. Industrial and academic collaborators included Nortel Networks, Newbridge Networks (now Alcatel), Lockheed-Martin/Sanders and McGill, McMaster, Toronto and Heriot-Watt Universities. The programme demonstrated a free-space 'intelligent optical backplane' exploiting emerging optoelectronic technologies with 1024 optical channels packaged per square centimeter of bisection area.

Dave Gilbert received his PhD at the Department of Computing and Software in McMaster University in 2007, in the area of nuclear reactor modelling. He is currently a Post-Doctoral Fellow at the Department of Electrical and Computer

Engineering. His research interests include nuclear reactor modelling, software-based problem-solving environments and network performance modelling.

1 Introduction

Existing supercomputing facilities typically use clusters of computers interconnected with high-bandwidth switch-based networks. The NASA Supercomputing Facility (NAS) provides data on current system configurations and benchmark programs. These systems typically use the Message Passing Interface (MPI) protocol for inter-processor communications, supported over switch-based IP networks. References NAS, Neworking Resource (2007), Shalf et al. (2005) and Reisen (2006), describe tools to monitor IP packet flows in such networks, providing information of traffic patterns, utilisation, latency, QoS, jitter and packet loss parameters between competing IP flows, as well as end-to-end performance measures. To achieve QoS, traffic is typically routed through the network using dynamic IP shortest path algorithms, and packets are scheduled for transmission across switches according to a dynamic scheduler.

Fat-Trees (Leiserson, 1985) and other 'Fully Connected Networks' (FCNs) are popular networks in High Performance Computing systems. Fat-Trees and other high-bisection bandwidth FCNs simplify the mapping of parallel applications with arbitrary communication topologies onto processors. As of 2004, 94 of the world's top 100 supercomputing systems employ FCNs, of which 92 are Fat-Trees (Shalf et al., 2005) Fat-Trees can offer relatively simple routing: A packet from source A to destination B travels an upward path in the tree until it reaches a common ancestor, at which point it follows the unique downward path to the destination. However, the use of dynamic algorithms for routing and switch scheduling in Fat-Trees significantly limits the performance of such networks. According to Karinieni and Nurmi (2003, 2004), throughputs vary from 10 to 45% when using dynamic routing and scheduling algorithms. Several papers have explored new topologies (Greenberg, 1994; Sethu et al., 1998) and dynamic and deterministic algorithms for improving communications in Fat-Trees (Ding et al., 2006; Gomez et al., 2007; Kumar and Kale, 2004; Lin et al., 2004; Matsutani et al., 2007; Stumpen and Krishnamurthy, 2002; Yuan et al., 2007). Fat-Trees are also attractive for Network-on-a-Chip applications (Greenberg, 1997; Kariniemi and Nurmi, 2003, 2004). In this paper, we present results of a deterministic algorithm for scheduling cell transmissions in a switch-based network such as a Fat-Tree, which can achieve low-jitter guaranteed-rate communications with 100% throughput.

The scheduling of packets through an Input Queued (IQ) crossbar switch to meet QoS constraints is an important task that switches and routers must perform. There are several substantially different approaches to the switch scheduling task. In the 'Dynamic Best-Effort' packet scheduling method, each switch has dedicated hardware to compute bipartite graph matchings between the input ports and output ports. Given a transmission line rate of 40 Gbit/sec, the duration of a time-slot required to transmit a fixed sized cell of 64 bytes is 12.8 nanosec (nsec). Due to this stringent time constraint, existing

best-effort IQ switch schedulers usually have difficulty achieving throughputs above 80% through one switch, and they generally cannot provide exceptionally high QoS with hard delay guarantees. Schemes where an IQ switch can achieve the performance of an ideal output queued switch have been proposed, but they are computationally expensive (McKeown et al., 1999).

An alternative approach to dynamic scheduling is called 'Guaranteed Rate' (GR) scheduling (Goyal and Vin, 1997). IP traffic can be grouped into two classes, 'GR' traffic and 'Best-Effort' traffic. In each IP router, the GR traffic is specified between the Input and Output (IO) ports using a doubly stochastic traffic rate matrix. This matrix can be dynamically updated by a RSVP, IntServ or DiffServ algorithm, which reserves resources along an end-to-end path of IP routers when a new GR traffic flow is admitted into the IP network. Within each IP router, this GR traffic can be independently and deterministically scheduled for transmission during the connection setup time, to meet rigorous QoS constraints. The remaining best-effort traffic can then use any unused switch capacity in each IP router.

Several GR switch scheduling algorithms which achieve varying grades of QoS have been proposed by Weller and Hajek (1997), Hung et al. (1998), Chen et al. (2000), Chang et al. (1999), Parekh and Gallager (1993, 1994), Koksal et al. (2004), Keslassy et al. (2005), Kodialam et al. (2003), Mohanty and Bhuyan (2005) and Szymanski (2006, 2008, Accepted). An algorithm called BATCH-TSA for scheduling non-uniform traffic through a switch was proposed in Weller and Hajek (1997). The algorithm uses the edge colouring of bipartite graph to find collision-free schedules. For a fully loaded switch (α = 1) with a frame size of S, the access delay was bounded by $2(\alpha S - 1)$, that is, the access delay bound is proportional to the frame size. A GR scheme which uses the Slepian-Duguid algorithm is proposed in Hung et al. (1998).

Another approach is based on the Birkhoff-von Neumann (BVN) stochastic matrix decomposition algorithm (Chang et al., 1999; Chen et al., 2000). A doubly stochastic $N \times N$ traffic rate matrix specifies the desired traffic rates between the IO ports. The matrix is decomposed into a set of $N \times N$ permutation matrices and weights, which must be scheduled to form a transmission schedule for the frame. This approach provides rate guarantees for all admissible traffic matrices. The BVN decomposition algorithm for an $N \times N$ crossbar switch has complexity $O(N^{4.5})$ time. The delay performance of BVN decomposition can be improved by scheduling the permutations to minimise delays, using Generalised Processor Sharing (Chang et al., 1999; Chen et al., 2000; Parekh and Gallager, 1993, 1994). Nevertheless, according to Koksal et al. (2004), the worst-case delay can be very high with BVN decomposition: "Therefore, a higher (possibly much higher) rate than the long term average traffic rate of a bursty, delay sensitive traffic stream must be allocated in order to satisfy its delay requirement".

Another approach based on stochastic matrix decomposition was introduced in Koksal et al. (2004), which considered the problem of simultaneously minimising the service lag amongst multiple competing IP flows, while attempting to maintain high throughput. An unquantised traffic rate matrix is first quantised and then decomposed and scheduled. With speedup $S = 1+sN$ between 1 and 2, the maximum service lag over

all IO pairs is bounded by $O((N/4)(S/(S-1)))$. The speedup directly affects the QoS provided by the switch. According to Kosal et al. (2004):

> "with a fairly large class of schedulers a maximum service lag of $O(N^2)$ is unavoidable for input queued switches. To our knowledge, no scheduler which overcomes this $O(N^2)$ has been developed so far. For many rate matrices, it is not always possible to find certain points in time for which the service lag is small over all I-O pairs simultaneously."

A greedy stochastic matrix decomposition algorithm with the goal to minimise delay jitter amongst simultaneous competing IP flows was introduced in Keslassy et al. (2005) and Kodialam et al. (2003). The low-jitter GR traffic is constrained to be a relatively small fraction of the total traffic. The delay and jitter minimisation problem is first formulated as an integer programming problem which is NP-hard. They then formulate a greedy low-jitter decomposition with complexity $O(N^3)$ time. After the decomposition, the permutation matrices and associated weights must be scheduled to minimise jitter between all IO pairs. The resulting schedule requires a worst-case speedup of $O(\log N)$ and achieves throughputs of ≈80%. If the cost of the speedup is considered, the algorithm's throughput drops. However, in practice, the speedup needed is much lower than the theoretical bound. However, analytic bounds on the jitter are not available.

Another greedy stochastic matrix decomposition algorithm was proposed in Mohanty and Bhuyan (2005). This decomposition algorithm also yields a set of permutation matrices and associated weights which must be independently scheduled, as in Chen et al. (2000), Chang et al. (1999), Keslassy et al. (2005) and Kodialam et al. (2003). The algorithm is relatively quick, but it cannot guarantee 100% throughput, short-term fairness or a bounded jitter. The jitter increases as the size of the network N grows. The authors identify an open problem: "to determine the minimum speedup required to provide hard guarantees, and whether such guarantees are possible at all".

Szymanski (2006, 2008, Accepted), introduces a recursive fair stochastic matrix decomposition algorithm, wherein a doubly stochastic $N \times N$ traffic rate matrix is quantised, and decomposed directly into a sequence of F permutation matrices which form a frame transmission schedule in a recursive and relatively fair manner. No scheduling of the final permutation matrices is required. At each level of recursion, the traffic reservations in one rate matrix are split fairly evenly over two resulting rate matrices. The decomposition proceeds until the resulting matrices specify partial or complete permutation matrices. The algorithm is unique in that it is deterministic, it achieves 100% throughput through an IQ switch, and it achieves a speedup = 1 provided that the traffic-rate matrix can be quantised: Each guaranteed traffic rate is an integer multiple of a minimum bandwidth allotment, which equals a fraction $(1/F)$ of the transmission line rate, where F is a user-defined frame size. In addition, an expression for the maximum service lag and delay jitter over all simultaneous IO pairs is achievable (Szymanski, 2008, Accepted). The lag is bounded by a small number of ideal cell inter-departure times.

The delivery of traffic over any switch-based IP network such as a Fat-Tree network with very low delay jitter may be achievable by a GR scheme if certain conditions can be met. GR schemes precompute the transmission schedule for each IP router in advance,

and packets move through each IP router according to the deterministic pre-computed transmission schedule. Therefore, very low jitter delivery may be possible if:

1 the transmission schedule is fair such that the maximum service lag is bounded by a small amount.

2 each IP router buffers a sufficient number of cells to compensate for the service lag and to keep the transmission pipeline active.

Furthermore, if each destination also employs a 'receive buffer' with sufficient capacity to filter out any residual jitter, then essentially zero jitter may be achievable. Under these conditions, cells will be transmitted through each IP router along an end-to-end path in a deterministic pattern, where the transmission times within each IP router deviate from the ideal times only by the imperfections, or the 'service lead/lag', of the transmission schedule. The scheduling algorithm in Szymanski (2006, 2008, Accepted) exhibits relatively small service lags, which suggests that achieving very low delay jitter may be feasible. A proof that all simultaneous multimedia traffic flows in general packet-switched networks can be delivered with essentially-zero delay jitter, given a receive buffer of finite size for each flow, has been submitted (Szymanski, 2008, Submitted).

In this paper, it is shown that low-jitter guaranteed-rate communications can be achieved in Fat-Tree networks. Section 2 introduces the GR problem formulation. Section 3 introduces the cluster-computing traffic model. Section 4 presents the results for scheduling traffic in a Fat-Tree network. Section 5 contains concluding remarks.

2 Prior work

2.1 The guaranteed rate scheduling problem for input-queued switches

An $N \times M$ switch has N input and M output ports, for which a traffic rate matrix is specified. Each input port j $0 \le j < N$ has M Virtual Output Queues, one for each output port k, $0 \le k < M$. The GR traffic requirements for an $N \times N$ switch can specified in a doubly substochastic or stochastic traffic rate matrix Λ:

$$\Lambda = \begin{pmatrix} \lambda_{0,0} & \lambda_{0,1} & \cdots & \lambda_{0,N-1} \\ \lambda_{1,0} & \lambda_{1,1} & \cdots & \lambda_{1,N-1} \\ \cdots & & & \cdots \\ \lambda_{N-1,0} & \lambda_{N-1,1} & \cdots & \lambda_{N-1,N-1} \end{pmatrix}, \quad \begin{matrix} \sum_{i=0}^{N-1} \lambda_{i,j} \le 1 \\ \sum_{j=0}^{N-1} \lambda_{i,j} \le 1 \end{matrix} \tag{1}$$

Each element $\lambda_{j,k}$ represents the fraction of the transmission line rate reserved for GR traffic between IO pair (j, k). The transmission of cells through the switch is governed by the transmission schedule, also called a frame schedule. In an 8 × 8 crossbar switch with F=128 time slots per frame, the minimum allotment of bandwidth is $1/F < 1\%$ of the line rate, which reserves one time-slot per frame on a recurring basis. Define a new quantised traffic rate matrix R where each traffic rate is expressed as an integer number of the minimum bandwidth allotment:

$$R = \begin{pmatrix} R_{0,0} & R_{0,1} & \ldots & R_{0,N-1} \\ R_{1,0} & R_{1,1} & \ldots & R_{1,N-1} \\ \ldots & & & \ldots \\ R_{N-1,0} & R_{N-1,1} & \ldots & R_{N-1,N-1} \end{pmatrix}, \quad \begin{matrix} \sum_{i=0}^{N-1} R_{i,j} \le F \\ \sum_{j=0}^{N-1} R_{i,j} \le F \end{matrix} \tag{2}$$

Several of the following definitions will be useful (see Koksal et al., 2004; Szymanski, 2006) for similar definitions).

Definition 1: *A 'Frame schedule' of length F is a sequence of partial or full permutation matrices (or vectors) which define the crossbar switch configurations for F time slots within a frame. Given a line rate L, the frame length F is determined by the desired minimum allotment of bandwidth = L/F. To set the minimum quota of reservable bandwidth to ≤ 1 % of L, set F ≥ 100, that is F = 128.*

Definition 2: *The 'Ideal Inter-Departure Time' (IIDT) of cells in a GR flow between IO pair (j,k) with quantised rate R(i,j), given a frame of length F, line rate L in bytes/sec and fixed sized cells of C bytes, is given by: IIDT = F/R(i, j) time-slots, each of duration (C/L) sec.*

Definition 3: *The 'Received Service' of a flow with guaranteed rate R(i,j) at time slot t within a frame schedule of length F, denoted Sij(t), equals the number of permutation matrices in time slots 1...t, where t≤F, in which input port i is matched to output port j.*

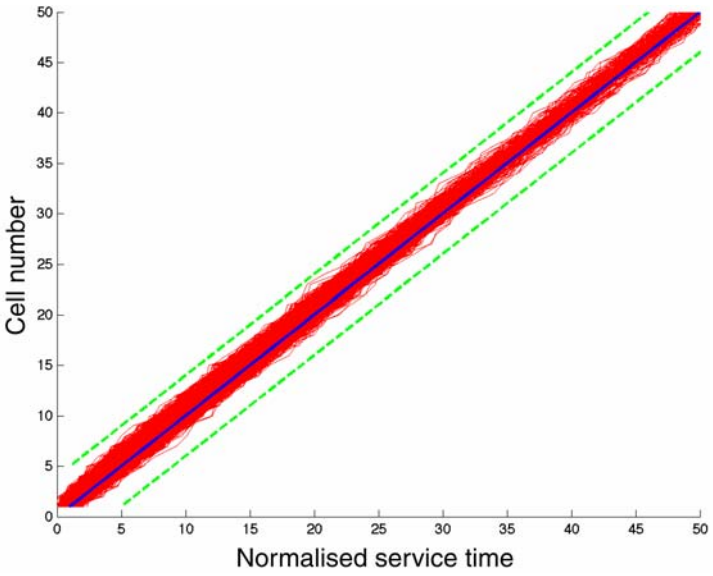
Definition 4: *The 'Service Lag' of a flow between IO pair (i,j), at time-slot t within a frame schedule of length F, denoted Lij(t), equals the difference between the requested service prorated by t/F, and the received service at time-slot t, that is, Lij(t) = (t/F)*R(i, j) – Sij(t).*

Example #1: Consider a crossbar switch with F = 1024, operating at 100% utilisation for GR traffic, a heavy load. The normalised received service for a 16 × 16 switch, given 100 randomly generated doubly stochastic traffic rate matrices with 100% utilisation, is shown in Figure 1(a). Each traffic rate matrix specifies 256 simultaneous GR flows to be scheduled which saturate the switch, and all 100 matrices represent 25,600 GR traffic flows. Each flow contains 64 cells on an average, so all matrices represent 1.6384 million cells to schedule. The normalised service received by each flow is illustrated by a red service line in Figure 1(a). The solid blue diagonal represents ideal normalised service, where the actual departure time of cell j equals the ideal departure time of $j \cdot \text{IIDT}$. The upper/lower dashed green diagonals represent a service lead/lag of 4 IIDTs. The received service is normalised by the ideal IDT, such that a cell which departs 2 IIDTs after its ideal departure time has a service lag of 2 units. The results in Figure 1(a) indicate that the service lead/lag of the scheduling algorithm is small, and suggests that GR traffic can be transported across an IP network with very low delay jitter provided each IP router has sufficient buffer space to compensate for any service lead/lag it may experience. According to Figure 1(a), the service lead/lag is typically less than 4 IIDTs, suggesting that the buffering of 4 cells per flow may suffice for most traffic.

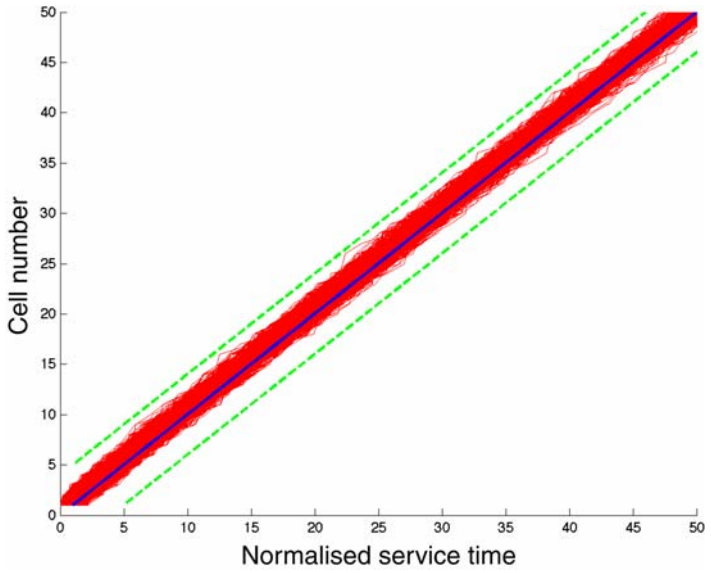
The normalised received service for a 64 × 64 crossbar switch, given 100 randomly generated doubly stochastic traffic rate matrices with 100% utilisation, is shown in

Figure 1(b). Each traffic rate matrix specifies 4096 simultaneous GR traffic flows to be scheduled, which saturate the switch, and all 100 matrices represent 409,600 GR traffic flows. Each flow contains 16 cells on an average, so all matrices represent 6.5536 million cells to schedule. Figure 1(a) and (b) indicate empirically that the service lags achieved by the algorithm in Szymanski (2006, 2008, Accepted), are relatively small, regardless of the switch size or traffic pattern.

Figure 1 (a) Service lead/Lag, 16 × 16 switch and (b) service lead/Lag, 64 × 64 switch (see online version for colours)



(a)



(b)

2.2 Fat-tree networks

A network with 2D VLSI area A is said to be area universal if it can simulate any other network with equivalent area, with at most a polylogarithmic slowdown *S*, that is, where $S \leq O(\log^n A)$ for constant n (Leiserson, 1995). A class of Universal networks called Fat-Tree interconnection networks were introduced by Leiserson (1985). Fat-Trees differ from conventional trees in that the bisection bandwidth increases at the upper levels of the trees. Due to the high bisection bandwidths, Fat-Trees represent a reasonable choice for a cluster-based supercomputing systems and silicon Networks-on-a-Chip.

Leiserson (1985) established that any set of M messages can be delivered by a Fat-Tree within d 'delivery cycles', provided that the number of messages originating and terminating at any processor is bounded and equals λ. Each delivery cycle $j \leq d$ delivers a subset of messages M_j, such that the original message set M is decomposed into a sequence of message sets $M_1, M_2, ..., M_d$ to be delivered. The number of delivery rounds $d = O(\lambda \log N)$, where N is the number of processors and λ is the load factor. A delivery round corresponds to a duration of time sufficient to allow the transfer of a packet of data between any two nodes in the Fat-Tree. All the packets within a message set are delivered in the same delivery cycle, that is the destinations of the messages within each set form a partial or full permutation. Leiserson's proof was theoretical and established the existence of an 'off-lin' algorithm to achieve the bound. There has been considerable research in the community in the search for efficient online and offline routing and scheduling algorithms which can achieve Leiserson's theoretical bounds.

In this paper, we summarise a two-phase offline algorithm consisting of

1 a 'global routing phase'

2 a 'local switch scheduling phase', which can deliver a message set M in a buffered Fat-Tree using Input Queued switches.

The scheduling algorithm used in the second phase achieves 100% throughput through each IQ switch with unity speedup, and it allows for the pipelined transmission of multiple messages (cells) through the packet-switched network simultaneously. Furthermore, the transmission schedule is low-jitter, in that the maximum jitter amongst all simultaneous competing flows through the Fat-Tree is bounded by a small number of 'Ideal Inter-Departure Times'.

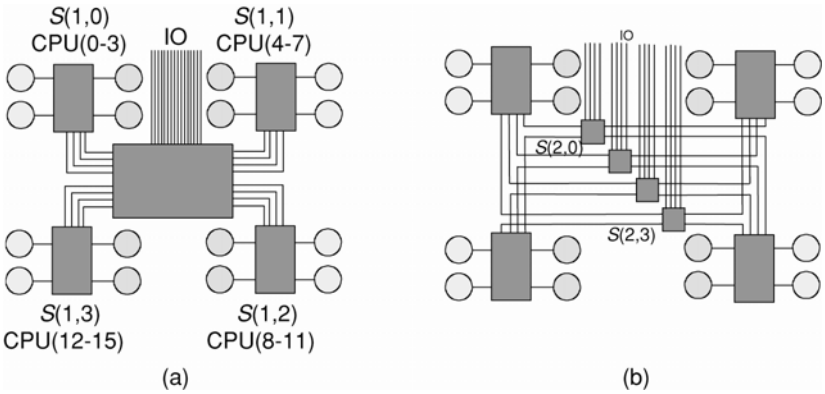
3 The traffic model

Consider a Fat-Tree based architecture consisting of 4 processor clusters as shown in Figure 2(a). Each cluster is interconnected via an 8 × 8 crossbar switch in level 1, denoted $S(1,j)$, for $0 \leq j < 4$, which provides 4 channels for local inter-processor communications within the cluster, and 4 channels for global communications between clusters. Assume that each logical channel is a 10 Gigabit/sec (Gbps) link. Each level-1 8 × 8 crossbar switch therefore provides an aggregate bandwidth of 80 Gbps for inter-processor communications.

Each level-1 crossbar switch is also interconnected to a level-2 root crossbar switch, which provides the global communication bandwidth between clusters. In Figure 2(a), the root switch is a 32 × 32 crossbar switch, with an aggregate bandwidth of 320 Gbps, where 16 channels are reserved for inter-cluster communications while 16 channels are available for parallel IO. The bisection of the network in Figure 2(a), is 16 channels, corresponding to a bisection bandwidth of 160 Gbps.

The hardware cost of the topology in Figure 2(a), can be reduced, by exploiting the fact that processors within one cluster will never communicate amongst themselves through the root switch. Therefore, the single 32 × 32 root switch can be replaced by 4 parallel 8 × 8 crossbar switches, as shown in Figure 2(b), Furthermore, if the channels for the external IO are not required, the root can be replaced by four 4 × 4 crossbar switches. However, the topology in Figure 2(b), introduces a multiplicity of paths between any source and destination, which complicates the global routing phase.

Figure 2 (a) Fat-Tree, single root switch and (b) fat-Tree, parallel root switches



Reisen (2006) examined the NAS benchmarks and demonstrated that many NAS applications exhibit global traffic rate matrices with pronounced diagonals (Reisen, 2006). For example, in the CG and BT benchmarks on 16 processors, the 4 processors in a cluster communicate amongst themselves intensely, while communications between clusters is less intense. The traffic rate matrix places most of the communication intensity only on the main diagonals and subdiagonals.

Shalf et al. (2005) also examined communication topologies in a different set of supercomputing applications, and also demonstrated that many applications have predominantly diagonal matrices. Prior research has identified several generic traffic matrices which are known to be hard to schedule through an IQ crossbar switch. The 'Log-Diagonal' traffic pattern is one such pattern and is shown in Equation (3):

$$LD = \left(\frac{\lambda}{1-2^{-N}}\right)\begin{bmatrix} 2^{-1} & 2^{-2} & \mathrm{L} & 2^{-N+1} & 2^{-N} \\ 2^{-N} & 2^{-1} & 2^{-2} & \mathrm{L} & 2^{-N+1} \\ 2^{-N+1} & 2^{-N} & 2^{-1} & 2^{-2} & \mathrm{L} \\ \mathrm{L} & 2^{-N+1} & 2^{-N} & 2^{-1} & 2^{-2} \\ 2^{-2} & \mathrm{L} & 2^{-N+1} & 2^{-N} & 2^{-1} \end{bmatrix} \tag{3}$$

Let λ denote the utilisation of a processor, that is the probability it transmits (or receives) a message in a time-slot. The matrix is doubly substochastic, given that the sum of each row or column $= \lambda \leq 1$. In this paper, we assume that the local communications within a cluster follow the Log-Diagonal traffic pattern. Furthermore, we assume that the global communication between clusters exhibits a log-diagonal trend, with bands of intensity as shown in Equation (4). This choice is arbitrary, since the algorithm in Szymanski (2006, 2008, Accepted), is applicable to any doubly substochastic or stochastic matrix.

The local 4 × 4 traffic rate matrix b for a cluster with 4 processors is given by Equation (4). Element $b(j, k)$ represents the fraction of local traffic leaving processor j which is directed to processor k. The Log-Diagonal matrix in Equation (3) is assumed for local communication.

$$b = \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \end{bmatrix} \tag{4}$$

The global 16 × 16 traffic rate matrix M for the system with 16 processors is given by Equation (5), where each matrix $B_{j,k}$ is a 4 × 4 traffic rate matrix that represents the relative traffic rates between the sending processors in cluster j and the receiving processors in cluster k, and scalar c_d indicates the traffic intensity along each diagonal.

$$M = \begin{bmatrix} c_1 B_{0,0} & c_2 B_{0,1} & c_3 B_{0,2} & c_4 B_{0,3} \\ c_4 B_{1,0} & c_1 B_{1,1} & c_2 B_{1,2} & c_3 B_{1,3} \\ c_3 B_{2,0} & c_4 B_{2,1} & c_1 B_{2,2} & c_2 B_{2,3} \\ c_2 B_{3,0} & c_3 B_{3,1} & c_4 B_{3,2} & c_1 B_{3,3} \end{bmatrix} \tag{5}$$

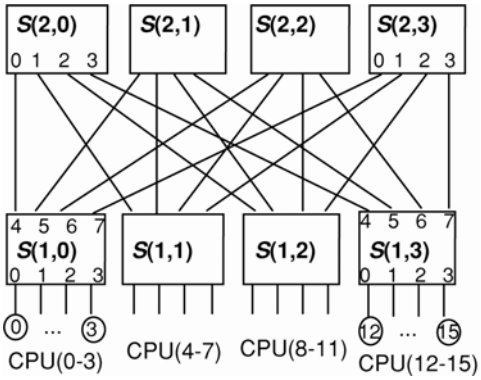
Assuming a frame size of 1024, and coefficients $c_1 = c_2 = c_3 = c_4 = 1.0$, then the quantised 16 × 16 traffic rate matrix G specifying the global traffic is given by Equation (6):

$$G = \left[\begin{array}{cccc|cccc|cccc|cccc} 291 & 146 & 73 & 36 & 146 & 73 & 36 & 18 & 073 & 36 & 18 & 9 & 036 & 18 & 9 & 5 \\ 36 & 291 & 146 & 73 & 18 & 146 & 73 & 36 & 9 & 073 & 36 & 18 & 5 & 036 & 18 & 9 \\ 73 & 36 & 291 & 146 & 36 & 18 & 146 & 73 & 18 & 9 & 073 & 36 & 9 & 5 & 036 & 18 \\ 146 & 73 & 36 & 291 & 73 & 36 & 18 & 146 & 36 & 18 & 9 & 073 & 18 & 9 & 5 & 036 \\ \hline 036 & 18 & 9 & 5 & 291 & 146 & 73 & 36 & 146 & 73 & 36 & 18 & 073 & 36 & 18 & 9 \\ 5 & 036 & 18 & 9 & 36 & 291 & 146 & 73 & 18 & 146 & 73 & 36 & 9 & 073 & 36 & 18 \\ 9 & 5 & 036 & 18 & 73 & 36 & 291 & 146 & 36 & 18 & 146 & 73 & 18 & 9 & 073 & 36 \\ 18 & 9 & 5 & 036 & 146 & 73 & 36 & 291 & 73 & 36 & 18 & 146 & 36 & 18 & 9 & 073 \\ \hline 073 & 36 & 18 & 9 & 036 & 18 & 9 & 5 & 291 & 146 & 73 & 36 & 146 & 73 & 36 & 18 \\ 9 & 073 & 36 & 18 & 5 & 036 & 18 & 9 & 36 & 291 & 146 & 73 & 18 & 146 & 73 & 36 \\ 18 & 9 & 073 & 36 & 9 & 5 & 036 & 18 & 73 & 36 & 291 & 146 & 36 & 18 & 146 & 73 \\ 36 & 18 & 9 & 073 & 18 & 9 & 5 & 036 & 146 & 73 & 36 & 291 & 73 & 36 & 18 & 146 \\ \hline 146 & 73 & 36 & 18 & 073 & 36 & 18 & 9 & 036 & 18 & 9 & 5 & 291 & 146 & 73 & 36 \\ 18 & 146 & 73 & 36 & 9 & 073 & 36 & 18 & 5 & 036 & 18 & 9 & 36 & 291 & 146 & 73 \\ 36 & 18 & 146 & 73 & 18 & 9 & 073 & 36 & 9 & 5 & 036 & 18 & 73 & 36 & 291 & 146 \\ 73 & 36 & 18 & 146 & 36 & 18 & 9 & 073 & 18 & 9 & 5 & 036 & 146 & 73 & 36 & 291 \end{array}\right] \tag{6}$$

A plot of the traffic intensity for the global matrix is shown in Figure 4. The plot is similar to Reisen's data for the NAS benchmarks, and Shalf's data from the Lawrence Berkeley National Labs. In the offline *global routing phase*, the flows specified in G must be routed through the switches in the network to ensure that no switch is overloaded. This task can be accomplished by the optimising compiler, which maps processing tasks onto processors in a manner to minimise global communication requirements. The compiler will have knowledge of the traffic rates between processors, and can allocate tasks to processors to determine the global traffic rate matrix. Once the matrix G is determined, the optimising compiler can allocate traffic flows to routes through the network. There are several well-known methods to maximise flows in a network. After the global routing phase, the local traffic rate matrices are specified for each switch and the traffic can be independently scheduled through each switch. As an alternative to the offline methodology, the local traffic rate matrix for each switch can be maintained online dynamically by an RSVP or DiffServ algorithm, which uses a dynamic shortest-path algorithm (wherein the shortest path has the least queueing delay for the desired flow) in conjunction with an online resource reservation mechanism to search for and reserve bandwidth along paths in an IP network.

Referring to the Fat-Tree topology in Figure 2(a), a global routing scheme to ensure that the global traffic can be realised and that each switch is not overloaded is straight-forward. There exists a unique shortest-distance path between every source and destination. Each IP flow traverses the unique upward path from its source, until a common ancestor with the destination is reached, at which point it traverses the unique downward path. Due to the full bisection bandwidth, all flows can be simultaneously supported and no switch can be overloaded. However, the Fat-Tree topology in Figure 2(b) introduces multiple upward paths, and the global routing phase must eliminate any routing conflicts in the upper level switches. For illustrative purposes, we assume a straight-forward rule to achieve a conflict-free routing for the global traffic rate matrix in Equation (6), which exploits the fact that the Fat-Tree topology in Figure 2(b) is equivalent to a one-sided Clos network, as shown in Figure 3.

Figure 3 One-Sided 16 × 16 Clos Network



Forwarding Rule: CPU(*j*) forwards all its global traffic over the single switch $S(2, (j \bmod 4))$ in level 2, for $0 \leq j < 15$. According to this rule, CPU(0) forwards all its global traffic over the single link to switch $S(2,0)$, $CPU(1)$ forwards all its global traffic over the single link to switch $S(2,1)$, etc.

According to this rule, every switch in level 2 receives 477 units of traffic on each link from the switches in level 1. Furthermore, according to this rule, the traffic arriving at every switch in level 2 is evenly distributed amongst the 4 switches that it spans in level 1. By tracing all the flows through the Fat-Tree network in Figure 3 according to the above forwarding rule, a traffic rate matrix can be specified for each crossbar switch in levels 1 and 2. In this example, due to the symmetry of the global traffic rate matrix in Equation (6), all of the crossbars in levels 1 and 2 share the same local traffic rate matrices $M1$ and $M2$, respectively:

$$M1 = \begin{bmatrix} \begin{bmatrix} 291 & 146 & 73 & 36 \\ 36 & 291 & 146 & 73 \\ 73 & 36 & 291 & 146 \\ 146 & 73 & 36 & 291 \end{bmatrix} & \begin{bmatrix} 477 & 0 & 0 & 0 \\ 0 & 477 & 0 & 0 \\ 0 & 0 & 477 & 0 \\ 0 & 0 & 0 & 477 \end{bmatrix} \\ \begin{bmatrix} 255 & 127 & 63 & 32 \\ 32 & 255 & 127 & 63 \\ 63 & 32 & 255 & 127 \\ 127 & 63 & 32 & 255 \end{bmatrix} & \begin{bmatrix} 000 & 0 & 0 & 0 \\ 0 & 000 & 0 & 0 \\ 0 & 0 & 000 & 0 \\ 0 & 0 & 0 & 000 \end{bmatrix} \end{bmatrix}, \quad M2 = \begin{bmatrix} 0 & 273 & 136 & 68 \\ 68 & 0 & 273 & 136 \\ 136 & 68 & 0 & 273 \\ 273 & 136 & 68 & 0 \end{bmatrix} \tag{7}$$

There are a few variations to consider.

1 When the Fat-Tree topology in Figure 2(b) is used with an arbitrary global traffic rate matrix, a maximum flow algorithm can be used in the global routing phase to determine a routing for every flow which ensures that no switch is overloaded.

2 Many applications may generate a non-saturated (i.e. doubly substochastic) global traffic rate matrix, that is, some applications may require only 50% of the network capacity, that is $\lambda = 0.5$ in Equation (3). It is desirable to 'over-provision' the traffic rates allocated for inter-processor communications in such cases, by converting the doubly substochastic traffic rate matrix into a doubly stochastic traffic rate matrix, by setting $\lambda = 1$ in Equation (3).

3 Reserved bandwidth which is unused by a flow may be used by other flows.

4 A parallel algorithm may have several steps each requiring a different global traffic rate matrix. Parallel tasks often synchronise their execution using barrier synchronisation primitives in MPI; the optimising compiler can revise the matrix G and revise the switch schedules during such synchronisation points.

4 End-to-end jitter analysis

Assume a computing cluster with 16 processors (CPUs) interconnected with a 2-level radix-4 Fat-Tree network, as shown in Figure 2(b). The representative global traffic pattern is based upon NAS benchmarks as shown in Equation (6). Each processor is allocated a low-jitter GR flow to every other processor, with a bandwidth determined by the global stochastic traffic rate matrix G, resulting in 256 GR flows to be routed and scheduled. Many flows will pass through the root of the Fat-Tree. All switches in the

Fat-Tree must be scheduled to support the traffic specified by the 256 simultaneous guaranteed-rate traffic flows.

The following assumptions are also made:

1 the links between switches are 10 Gbps fibres

2 all IP packets have fixed size of 64 bytes, which correspond to the IP router cell size

3 a frame size of $F=1024$ is selected at each IP router.

Assumption (2) is made to simplify the presentation, and to reduce the delay associated with disassembling and reassembling variable size IP packets in routers. The frame size determines the minimum increment of GR bandwidth which equals 10 Mbps, that is each time-slot reservation in frame reserves 10 Mbps of guaranteed bandwidth.

Given the global traffic rate matrix in Equation (6), every processor is both transmitting and receiving at essentially 100% capacity. This assumption represents a heavy load for GR traffic, not likely to be seen in a real IP network. Given this worst-case load, the IP routers should find it more challenging to schedule the traffic to meet QoS guarantees. The global matrix in Equation (6) allows for self-directed traffic, that is processor 0 may transmit to processor 0 through the level 1 switch. We allow self-directed traffic for 2 reasons:

1 It is part of the Log-Diagonal traffic pattern, which is known to be hard to schedule.

2 Reisen's data for the NAS benchmark indicates self-directed traffic exists.

The scheduling algorithm in Szymanski (2006, 2008, Accepted) was used to schedule the traffic in each switch of the Fat-Tree. The service lead/lag for the switches in levels 1 and 2 are shown in Figure 4(a) and (b), respectively. We observe that the service lead/lag is very low, typically less than 4 IIDTs, for all traffic flows across the network. This result is consistent with the results shown earlier in Figure 1(a) and (b) for randomly generated matrices and is consistent with the theory established in Szymanski (2006, 2008, Accepted).

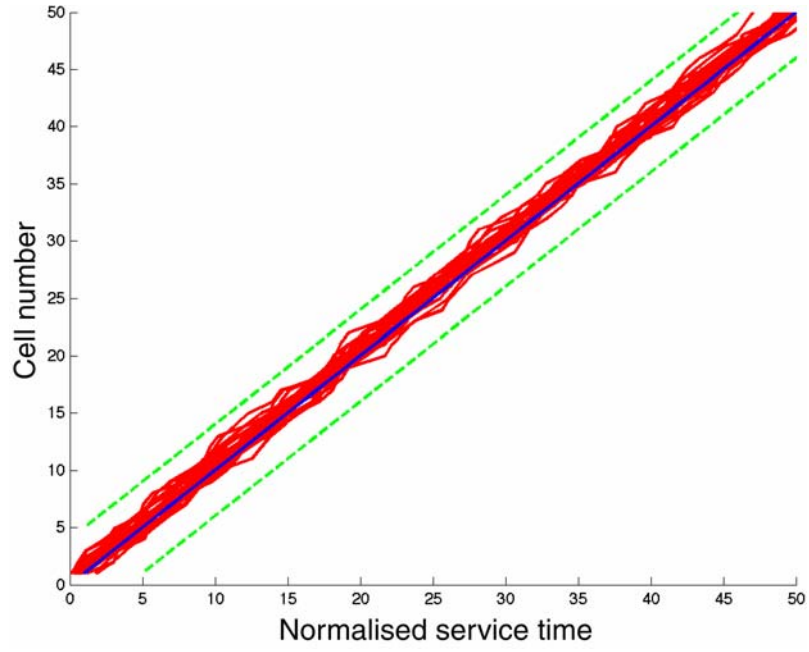
A flow which reserves j time-slots per frame belongs to class j. The observed IDTs for all classes of traffic flows in the Fat-Tree are given in Table 1(a) and (b). We observe that:

1 the average IDT for cells in a flow equals the ideal IDT

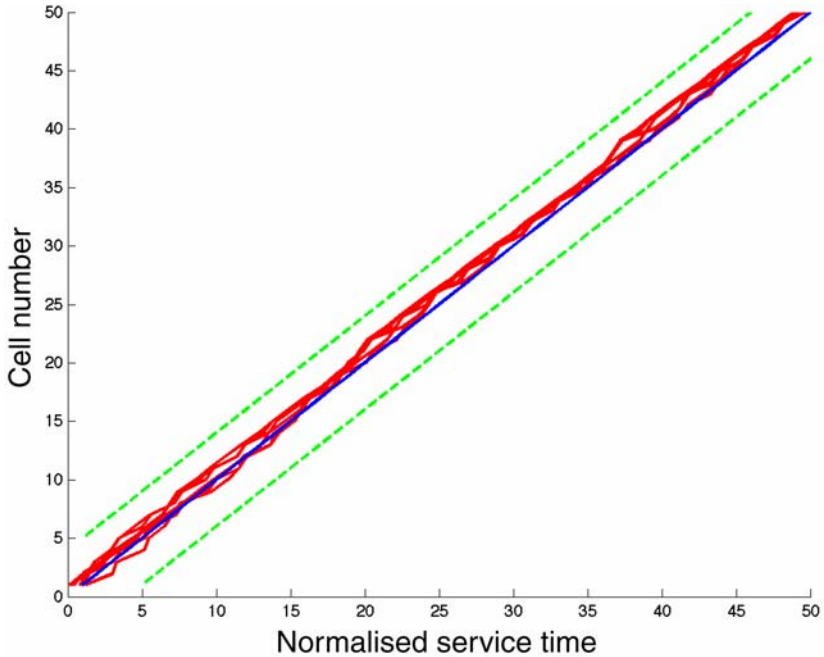
2 the standard deviation of the IDT for every flow is very small, approximately one-half of an ideal IDT.

Consider the flows with rate 127 time-slots per frame. The IIDT equals $1024/127 = 8.062$ time-slots. The standard deviation in the observed IDTs is 4.22 time-slots, indicating that most cells depart within 4.22 time-slots of their ideal departure times.

Figure 4 (a) Service lead/lag, 8 × 8 switch, level 1 and (b) Service lead/lag, 4 × 4 switch, level 2 (see online version for colours)



(a)



(b)

Table 1 Observed IDT and service lead/lag, for switches in levels 1 and 2, with $F = 1024$

Flow Class	*IIDT in time-slots*	*Observed IDT time-slots*	*SD timeslots*	*MinIDT timeslots (IIDT)*	*Max IDT time-slots (IIDT)*	*Min Norm. Lead (IIDT)*	*Max Norm. Lag (IIDT)*
Fat-Tree, Level 1 Observed IDT and Service Lead/Lag, N=8, F=1024							
32	32.000	32.000	8.19	9 (0.281)	64 (2.00)	–1.16	0.969
36	28.444	28.444	12.90	7 (0.246)	64 (2.25)	–2.8	0.02
63	16.254	16.254	5.39	1 (0.0615)	34 (2.09)	–1.73	0.845
73	14.027	14.027	6.02	4 (0.285)	39 (2.78)	–1.36	1.55
127	8.063	8.063	4.22	2 (0.248)	22 (2.73)	–1.81	1.70
146	7.014	7.014	3.15	2 (0.285)	20 (2.85)	–3.59	0.949
255	4.016	4.016	1.69	1 (0.249)	10 (2.49)	–2.72	0.731
291	3.519	3.519	1.34	1 (0.284)	8 (2.27)	–3.27	1.39
477	2.147	2.147	0.604	1 (0.466)	6 (2.79)	–1.92	1.3
Fat-Tree, Level 2 Observed IDT and Service Lead/Lag, $N = 4$, $F = 1024$.							
68	15.059	15.059	6.73	4 (0.266)	32 (2.12)	–1.81	1.18
136	7.529	7.529	2.79	4 (0.531)	12 (1.59)	–1.55	0.297
273	3.751	3.751	1.18	2 (0.533)	8 (2.13)	–2.87	0.613

The minimum observed IDT was 2 time-slots, corresponding to 0.248 IIDT, and the maximum observed IDT was 22 time-slots, corresponding to 2.73 IIDT. The minimum service lead was 1.81 IIDT, and the maximum service lag was 1.7 IDT. (By convention, the service lead is negative).

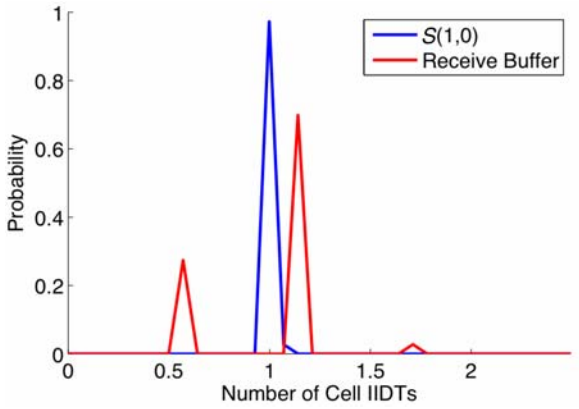
A computer program which simulated the movement of packets along all 256 end-to-end path through the Fat-Tree was generated. The simulator gathers detailed statistics on delay and jitter and has over 20,000 lines of code. Each switch reserves guaranteed bandwidth according to the traffic rate matrices presented earlier. Each switch is an IQ crossbar with unity speedup. The packet movement over all 256 simultaneous flows was simulated from the source CPU, through the switches of the Fat-Tree, arriving at a Receive-Buffer associated with each destination CPU. Each switch had an IQ associated with each input port, which could buffer multiple cells. We assume that each of the 256 distinct flows has its own receive buffer at the relevant destination CPU. The 'time-of-flight' delay over the fibers is not reported, as it is a fixed value.

Each time-slot has a duration of ≈51.2 nsec, and a frame with $F = 1024$ time-slots has a duration of ≈52.4 μsec. For simplicity, we assume that cells are generated at every source CPU at the rate of one cell every IIDT interval. This assumption lets us isolate the jitter introduced by the switch schedules. Each IP router transmitted cells through the switch according to its precomputed frame transmission schedule.

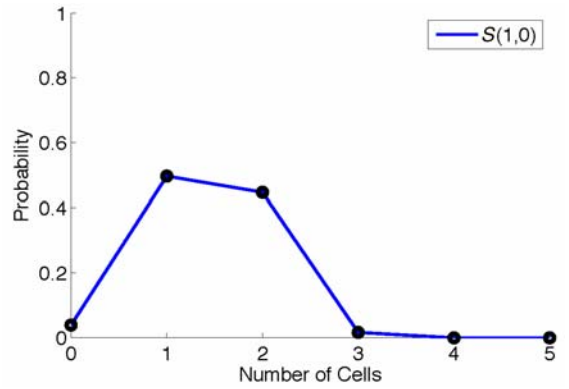
Figure 5(a) illustrates the Inter-Arrival Time (IAT) Probability Density Function (PDF) for a typical 1-hop path, for flow(0,2) from CPU(0) to CPU(2), and Figure 5(b) illustrates the queue occupancies along this path. According to Equation (6), the traffic rate for flow(0,2) along this path is 73 cells per frame, or 730 Mbps, with an IIDT = 14.03 time-slots. Cells arrive at the receive buffer at an average rate of 1 cell every IIDT interval, with a relatively small jitter: The minimum and maximum IATs are

0.15 IIDT and 1.75 IDT respectively. Figure 5(b) illustrates the PDF for the number of cells belonging to flow (0, 2) queued in the level-1 switch *S(*1, 0). The input queue of the switch buffers between 1 and 2 cells for flow(0, 2). With this queue occupancy, the queue length remained stable, the data transmission followed a deterministic pattern and the delay jitter observed at the receive buffer was relatively small, ≤ 2 IIDT. Using Little's Law, the average delay along this end-to-end path for flow(0, 2) ≈ 0.98 μsec. This average delay corresponds to 19.6 time-slots, indicating that every cell waits slightly more than one IIDT in the queue in the level-1 switch. This result is consistent with Table 1 and Figures 1 and 2, which indicate that most cells will receive service within 4 IIDTs on average.

Figure 5 (a) IAT PDF, 1-hop path and (b) Queue occupancy, 1-hop path (see online version for colours)



(a)

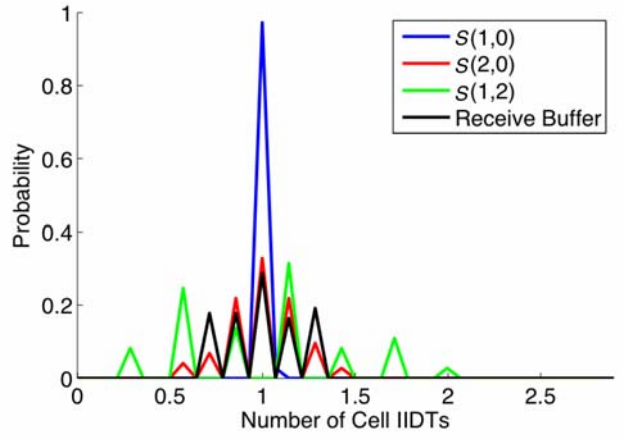


(b)

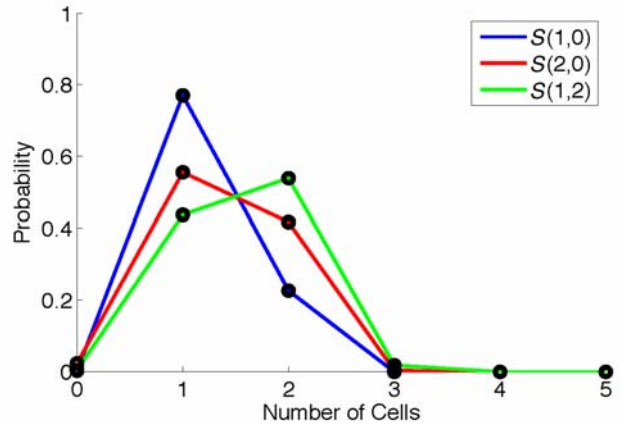
Figure 6(a) illustrates the IAT PDF for a typical 3-hop path for flow(0,8), from CPU(0) to CPU(8), and Figure 6(b) illustrates the queue occupancies along this path for flow(0,8). According to Equation (6), the traffic rate for flow(0,8) along this path is also 73 cells per frame, or 730 Mbps, with an IIDT = 14.03 time-slots. Cells arrive at the level 2 switch *S*(2,0) at an average rate of 1 cell every IIDT interval, with a minimum and maximum IAT of 0.1 IDT and 4 IIDT, respectively. Figure 6(b) illustrates the PDF for the number of cells from flow (0,8) queued in switches *S*(1,0), *S*(2,0) and *S*(1,2). The

input queues these 3 switches buffer at an average of 1.2, 1.4 and 1.6 cells, respectively for this flow. With these queue occupancies, the delay jitter of cells arriving at the receive buffer for flow(0,8) was relatively small, ≤ 1 IIDT. The queue lengths remained stable, and the data transmission followed a deterministic pattern.

Figure 6 (a) IAT PDF, 3-hop path and 6 (b) Queue occupancy, 3-hop path (see online version for colours)



(a)

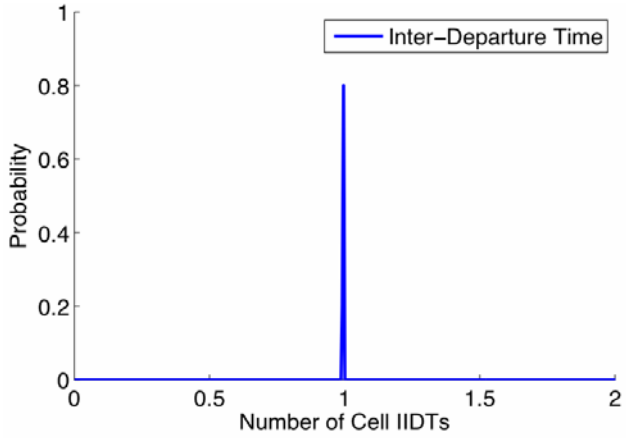


(b)

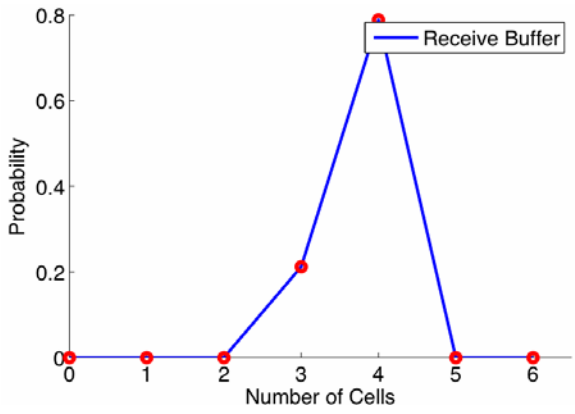
The simulation for all 256 simultaneous flows was repeated when each receive buffer acted as a 'playback buffer' in a video application. The objective of the playback buffer is to deliver cells to the end-user with very low (essentially zero) delay jitter, so cells are constrained to depart with an IDT = 1 IIDT, when they are available. For flow (0,8), cells where released from the receive buffer to the destination application at a constant rate of one cell every IIDT interval, with the first cell being released 4 IIDT after its arrival time at the receive buffer. The inter-departure time observed at the receive buffer is shown in Figure 7(a), and the queue occupancy PDF at the receive buffer is shown in Figure 7(b). The receive buffer has an average occupancy of 3.8 cells, and a maximum occupancy of 4 cells. Figure 7(a) shows that all cells are delivered at their ideal times, and that the

delay jitter is precisely zero. Our simulations indicate that all 256 simultaneous traffic flows can be delivered with precisely zero delay jitter, using relatively small playback buffers. We have simulated many longer end-to-end paths of routers, with different GR traffic matrices and computed frame schedules, and observed similar results. The number of cells queued at each router needed to provide very low jitter is typically ≤ 4–8 cells per flow, from our simulations. A proof that all network-introduced delay jitter can be removed over all simultaneous traffic flows, using playback buffers of bounded size, is presented in Szymanski (2008, Submitted).

Figure 7 (a) IDT PDF, destination, 3-hop path and (b) queue occupancy, destination (see online version for colours)



(a)



(b)

4.1 Extensions

The proposed scheduling methodology is applicable to all switch-based networks, including Fully Connected Networks, meshes, tori, hypercubes, multistage networks and Clos networks. Once the traffic is routed through the network in the global routing phase such that no switch is overloaded, the switches can be scheduled to transmit low-jitter

guaranteed rate traffic using the algorithm presented in Szymanski (2006, 2008, Accepted). In a HPC environment, the optimising compiler maps tasks onto processors, and will be able to compute a global traffic rate matrix for the application. The traffic rate matrix allows for the specification of point-to-point communications, as well as point-to-multipoint broadcasting, and multi-point-to-point gatherings.

According to Shalf et al. (2005), the largest parallel supercomputing systems today consist of up to 64 K processors. A cluster-based supercomputer with 64 K processors can be interconnected using a 2-stage radix-256 Fat-Tree, and the methodology presented herein can be used to realise low-jitter Guaranteed-Rate communications. This topology would require 256 switches of size 512 × 512 in the first level and 256 switches of size 256 × 256 in the second level. Switches with these sizes are readily available today: Myrinet has released a 1024-port switch with 10 Gbps links in 2007. This system could offer 100 % network throughput with low delay and jitter.

5 Computational complexity

In a general IP network where traffic flows change dynamically, the GR traffic matrix for each switch can be incrementally updated *online* by an RSVP, IntServ or DiffServ algorithm, when a new GR flow is added or removed from the network, or if its rate changes substantially. The update involves changing one entry in the rate matrix, after which a new frame schedule can be computed. In the current offline application, the optimising compiler will allocate tasks to processors to determine the global traffic rate matrix, determine a routing for the traffic flows through the network, and then precompute the frame transmission schedule for every switch. The scheduling algorithm in Szymanski (2006, 2008, Accepted), has a similar recursive structure as the well-known FFT algorithm. The FFT must perform several complex floating point multiply/add operations per node. The algorithm in Szymanski (2006, 2008, Accepted), performs several integer reads/writes per node. The time complexity of a serial implementation of the algorithm is estimated to be 1/5th that of the serial FFT algorithm, and we hypothesise that it may run potentially five times faster when compiled in C. An FFT of 8 K elements requires ≈1 millisec on a typical dual-core laptop using a single processor, and we estimate the stochastic matrix decomposition for $N = 8$, $F = 1024$ will require ≈200 μsec on a single processor. The execution time can be reduced by using a smaller frame size, that is, F=128, which will allocate link bandwidth with a resolution of ≈1%. We estimate that the stochastic matrix decomposition for a larger 16 × 16 switch with $F = 128$ will require ≈100 μsec. A multiple processor or dedicated parallel hardware implementation should run much faster, and our graduate students will embark in these directions. In any case, the time to compute the frame schedules is very small and is potentially negligible relative to the time an optimising compiler typically requires, which can be as high as several minutes or hours depending on the program complexity.

6 Conclusions

It has been shown that traffic flows can be delivered over a Fat-Tree network with guaranteed rate and delay and with very low delay jitter, provided that each IP router has the capacity to buffer a small number of cells per flow. Each IP router schedules the GR

traffic for transmission according to a scheduling algorithm based upon a Recursive Fair Stochastic Matrix Decomposition (Szymanski, 2006, 2008, Accepted). Simulations indicate that the number of cells queued in each router necessary to provide very low jitter delivery can be relatively small, typically about 2–4 cells per flow. (Theoretical bounds have been established (Szymanski, 2008, Submitted). The algorithm can be used to achieve essentially 100% throughput for the interconnection networks of High Performance Computing (HPC) systems. When receive buffers of modest depth are used to filter out residual network jitter, cells can be delivered along all simultaneous end-to-end flows with essentially zero delay jitter. This configuration may be attractive for silicon Network-on-a-Chip architectures and in Field Programmable Gate Array (FPGA) systems, where deterministic low-latency zero-jitter data pipelines can be configured in hardware. Additional results indicate that very low-jitter guaranteed rate communications can also be achieved under different traffic models (Szymanski and Gilbert, 2007; Szymanski, 2008 Submitted).

References

Chang, C-S., Chen, W.J. and Huang, H-Y. (1999) 'On service guarantees for input buffered crossbar switches: a capacity decomposition approach by Birkhoff and von Neuman', *IEEE iWQoS'99*, pp.79–86.

Chen, W.J., Chang, C-S. and Huang, H-Y. (2000) 'Birkhoff-von Neumann input buffered crossbar switches', *Proceedings of IEEE Infocom*, pp.1614–162.

Ding, Z., Hoare, R.R., Jones, A.K. and Melhem, R. (2006) 'Level-wise scheduling algorithm for fat tree interconnection networks', *Proceedings of ACM/IEEE Conference on Supercomputing, (Conference on High Performance Networking and Computing)*.

Gomez, C., Gilabert, F., Gomez, M.E., Lopez, P. and Duato, J. (2007) 'Deterministic versus adaptive routing in fat-trees', *IEEE International Parallel and Distributed Processing Symposium*, pp.1–8.

Goyal, P. and Vin, H.M. (1997) 'Generalized guaranteed rate scheduling algorithms: a framework', *IEE/ACM Transactions on Networking*, Vol. 5, No. 4, pp.561–571.

Greenberg, R.I. (1994) 'The fat-pyramid and universal parallel computation independent of wire delay', *IEEE Transactions on Computer*, Vol. 43, No. 12, pp.1358–1364.

Greenberg,. R.I. (1997) 'Universal wormhole routing', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 8, No. 3, pp.254–262.

Hung, A., Kesidis, G. and McKeown, N. (1998) 'ATM input buffered switches with guaranteed rate property', *Proceedings of IEEE ISCC*.

Kariniemi, H. and Nurmi, J. (2003) 'New adaptive routing algorithm for extended generalized fat trees on-chip', *IEEE International Symposium on System-on-Chip*, pp.113–118.

Kariniemi, H. and Nurmi, J. (2004) 'Reusable XGFT interconnect IP for network-on-chip implementations', *IEEE International Symposium on System-on-Chip*, pp.95–102.

Keslassy, I., Kodialam, M., Lakshamn, T.V. and Stiliadis, D. (2005) 'On guaranteed smooth scheduling for input-queued switches', *IEEE/ACM Transactions on Networking*, Vol. 13, No. 6, pp.1364–1375.

Kodialam, M., Lakshamn, T.V. and Stiliadis, D. (2003) 'Scheduling of guaranteed-bandwidth low-jitter traffic in input buffered switches', *US Patent Application.*

Koksal, C.E., Gallager, R.G. and Rohrs, C.E. (2004) 'Rate quantization and service quality over single crossbar switches', *Proceedings of IEEE Infocom*, pp.1962–1973.

Kumar, S. and Kale, L.V. (2004) 'Scaling all-to-all multicast on fat-tree networks', *Proceedings of 10th IEEE International Conference Parallel and Distributed Systems (ICPADS'04)*, pp.1–10.

Leiserson, C.E. (1985) 'Fat-trees: universal networks for hardware-efficient super-computing', *IEEE Transactions on Computer*, Vol. C-34, No. 10, pp.892–901.

Lin, X.Y., Chung, Y.C. and Huang, T.Y. (2004) 'A multiple LID routing scheme for fat-tree infiniband networks', *Proceedings of 18th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*.

Matsutani, H., Koibuchi, M. and Amano, H. (2007) 'Performance, cost and energy evaluation of fat H-tree: a cost-efficient tree-based on-chip network', *IEEE*.

Mohanty, S.R. and Bhuyan, L.N. (2005) 'Guaranteed smooth switch scheduling with low complexity', *IEEE Globecom*, pp.626–630.

McKeown, N., Mekittikul, A., Anantharan, V. and Walrand, J. (1999) 'Achieving 100% throughput in an input queued switch', *IEEE Transactions on Communication*, Vol. 47, No. 8, pp.1260–1267.

NAS Networking Resources (2007) Available at: www.nas.nasa.gov/Resources/Networks/networks.html.

Parekh, A.K. and Gallager, R.G. (1993) 'A generalized processor sharing approach to flow control in integrated service networks: the single node case', *IEEE/ACM Transactions on Networking*, Vol. 1, pp.344–357.

Parekh, A.K. and Gallager, R.G. (1994) 'A generalized processor sharing approach to flow control in integrated service networks: the multiple node case', *IEEE/ACM Transactions on Networking*, Vol. 2, pp.137–150.

Reisen, R. (2006) 'Communication patterns', *Proceedings of Workshop on Communication Architecture for Clusters (CSC), IEEE International Parallel and Distributed Processing Symposium*. Also see: Riesen, R. (2006) 'Communication patterns of the NAS parallel benchmarks', Available at: http://www.cs.sandia.gov/~rolf/NAS.

Sethu, H., Stunkel, C.B. and Stucke, R.F. (1998) 'IBM RS/6000 SP interconnection network topologies for large systems', *Proceedings of IEEE International Conference Parallel Processing*, pp.620–627.

Shalf, J., Kamil, S., Oliker, L. and Skinner, D. (2005) 'Analysing ultra-scale application communication requirements for a reconfigurable hybrid interconnect', *Proceedings of ACM/IEEE SC'05 Conference*.

Strumpen, V. and Krishnamurthy, A. (2002) 'A collision model for randomized routing in fat tree networks', MIT, Laboratory for Computer Science, *Technical Report MIT-LCS-TM-629*, pp.1–21.

Szymanski, T.H. (2006) 'QoS switch scheduling using recursive fair stochastic matrix decomposition', *IEEE High Performance Switching and Routing Conference*, pp.417–424.

Szymanski, T.H. (2007) 'Method and apparatus to schedule packets through a crossbar switch with delay guarantees', *US Patent Application*.

Szymanski, T.H. (2008, Accepted) 'A low-jitter guaranteed-rate scheduling algorithm for packet-switched IP routers', Accepted (with revision), *IEEE Transactions on Communications*.

Szymanski, T.H. and Gilbert, D. (2007) 'Delivery of guaranteed rate internet traffic with very low delay jitter', *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp.450–455.

Szymanski, T.H. and Gilbert, D. (2007, Submitted) 'Internet multicasting of IPTV with essentially zero delay jitter'.

Szymanski, T.H. (2008, Submitted) 'Bounds on the end-to-end delay and jitter in input-queued IP/MPLS networks'.

Weller, T. and Hajek, B. (1997) 'Scheduling nonuniform traffic in a packet-switching system with small propagation delay', *IEEE/ACM Transactions on Networking*, Vol. 5, No. 6, pp.813–823.

Yuan, X., Nienaber, W., Duan, Z. and Melhem, R. (2007) 'Oblivious routing for fat-tree based system area networks with uncertain traffic demands', *ACM Sigmetrics Performance Evaluation Review*, San Diego, USA, pp.337–348.