# A Low-Jitter Guaranteed-Rate Scheduling Algorithm for Packet-Switched IP Routers

T. H. Szymanski, *Member, IEEE*

***Abstract*—A Guaranteed-Rate scheduling algorithm for packet-switched IP routers with rate, delay and jitter guarantees is proposed. The algorithm can be used to schedule traffic with 100% throughput in Input-Queued IP routers with unity speedup. The traffic is scheduled according to transmission frames of duration F time-slots. An $NxN$ doubly stochastic traffic rate matrix specifies a traffic rate between each pair of IO ports. The matrix is quantized and recursively decomposed into a sequence of F permutations. Each permutation is used to configure the crossbar switch for one time-slot without requiring additional scheduling. The recursive fair stochastic matrix decomposition is based upon the routing of a permutation through a binary rearrangeable network. In the resulting transmission schedule, the expected Inter-Departure Time (IDT) of a cell equals the Ideal IDT (IIDT), and the maximum IDT and service lag of a cell are bounded by an integer number of IIDTs. The delay and delay jitter experienced along an end-to-end path in a packet-switched IP/MPLS network are therefore small and bounded by an integer number of IIDTs, and the buffer sizes within the IP routers are small and bounded. The proposed algorithm can be used to schedule Guaranteed-Rate traffic in packet-switched IP/MPLS networks, to provide near-optimal queueing delays and essentially-zero delay jitter along end-to-end paths when playback buffers are employed.**

***Index Terms*—Switching, scheduling, guaranteed rate, low jitter, multicasting, recursive fair stochastic matrix decomposition, quality of service, QoS.**

## I. Introduction

NEW multimedia services being offered over the Internet include telerobotic surgery [1], telerobotic mining, Voice over IP and television over IP (IPTV) [2]. Telerobotic control is very sensitive to delay variation, and will require tight Quality of Service (QoS) guarantees in IP routers and switches. Cruz established bounds on the delay in a packet-switched network when incoming traffic satisfies burstiness constraints [4,5]. Parekh and Gallager developed hard delay bounds when each IP router employs an Output Queued (OQ) packet-switch [3] using a Weighted Fair Queueing (WFQ) scheduling algorithm [6,7]. However, an OQ packet-switch requires an internal "speedup," which makes the approach infeasible for large switches. In contrast, an Input Queued (IQ) crossbar switch with unity speedup places the buffers at the input side, in a set of Virtual Output Queues [3,8,9]. The VOQs can eliminate the Head-of-Line (HOL) blocking inherent in IQ switches. A scheduling algorithm is used to determine sets of fixed-size cells to transfer from the VOQs to the outputs in each time-slot. Packet switches with Combined Input and Output Queueing (CIOQ) [10] and with internal crosspoint queues [11] have also been proposed.

There are two fundamentally different approaches to the packet-switch scheduling problem [12]: '*Dynamic Scheduling,*' and '*Guaranteed-Rate*' reservation-based scheduling. The scheduling problem is often formulated as a bipartite graph-matching problem. The IO ports of the switch are represented by sets of vertices, and the cells to transfer are represented by edges. In the Dynamic Scheduling approach, a new bipartite graph matching is recomputed for each time-slot, which is used to configure the crossbar switch. These approaches can adapt to dynamically varying traffic patterns. In [8] it was shown that 100 % throughput can be achieved for IQ packet-switches with unity speedup, if a *Maximum Weight Matching (MWM)* algorithm is used. However, such schemes are computationally intensive, ie the MWM algorithm incurs a complexity of $O(N^3)$ per time-slot. Furthermore, using standard 64 byte cells it is difficult to compute optimal matchings as the line-rate increases beyond 40 Gb/sec.

Guaranteed-Rate scheduling algorithms have been proposed for packet-based satellite systems, voice-oriented switches using TDM, frame relay switches, wireless networks, optical TDM switches, and more recently for packet-switched Internet Protocol (IP) routers. The first generation of such algorithms have been called 'Time-Slot-Assignment' (TSA) algorithms. More recently, these algorithms have been called 'Guaranteed-Rate' algorithms [13]. Many of the recent GR algorithms are based upon stochastic matrix decomposition algorithms, wherein a doubly substochastic or stochastic traffic rate matrix is decomposed in to a convex set of permutation matrices and associated weights. The matrices are then scheduled to appear in proportion to their weights. However, existing TSA and GR algorithms are relatively complex and time consuming, and they do not provide adequate bounds on the maximum delay or the delay jitter, especially when the speedup is constrained to unity. An open question remains as to whether hard delay and jitter guarantees are possible under the constraint of unity speedup. We answer this question affirmatively.

In this paper, a Guaranteed-Rate resource reservation algorithm based upon a *Recursive Fair Stochastic Matrix Decomposition (RFSMD)* is proposed. Preliminary results were presented in [45]. This paper extends the latter by providing (a) proofs for the bounds, (b) extensions of the bounds to individual competing traffic flows which share a link or a VOQ, (c) extensive simulations to demonstrate the theories, (d) a discussion of buffer requirements, and (e) extensions

Paper approved by T.-S. P. Yum, the Editor for Packet Access and Switching of the IEEE Communications Society. Manuscript received December 30, 2007; revised November 15, 2008.

The author is with the Dept. of Electrical and Computer Engineering, McMaster University, 1280 Main Street West, Hamilton, ON, Canada L8S 4K1 (e-mail: teds@mcmaster.ca).

Digital Object Identifier 10.1109/TCOMM.2009.11.070666

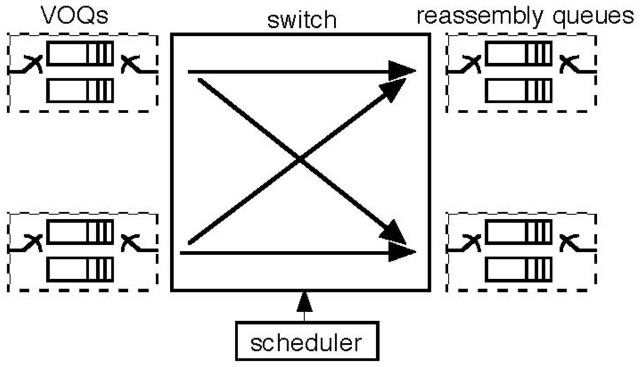


Fig. 1. IQ switch architecture.

of the results to internally buffered crosspoint switches. In the RFSMD algorithm, a doubly substochastic or stochastic NxN traffic rate matrix (ie an 'admissible' matrix) is first transformed into a quantized matrix with integer values. The quantized matrix is then decomposed into a sequence of $F$ permutation matrices in a recursive and relatively fair manner. The resulting sequence of permutations does not need to be scheduled, and is used to configure the IQ packet-switched crossbar switch directly for a sequence of F time-slots, yielding a low-jitter transmission schedule.

Fig. 1 illustrates an IQ packet-switch architecture. Arriving IP packets are segmented into fixed sized cells which are buffered in the relevant VOQs. A scheduling algorithm is used to transfer cells across the switch and place them in the reassembly buffers on the output side. Once an IP packet is reconstructed, it is transmitted. Consider the doubly stochastic traffic rate matrix for a 4x4 IQ switch as shown in Eq. (1), which was first introduced in [13,14]:

$$R = \begin{bmatrix} 0.38 & 0 & 0.22 & 0.40 \\ 0.11 & 0.24 & 0.60 & 0.05 \\ 0 & 0.53 & 0.04 & 0.33 \\ 0.51 & 0.23 & 0.04 & 0.22 \end{bmatrix} \quad (1)$$

A quantized traffic rate matrix $M$ which approximates matrix $R$, given a frame of size $F = 1024$ time-slots, is given in Eq (2):

$$M = \begin{bmatrix} 389 & 0 & 225 & 410 \\ 113 & 246 & 614 & 51 \\ 0 & 542 & 144 & 338 \\ 522 & 236 & 41 & 225 \end{bmatrix} \quad (2)$$

$$M/F = \begin{bmatrix} 0.3799 & 0 & 0.2197 & 0.4004 \\ 0.1104 & 0.2402 & 0.5996 & 0.0498 \\ 0 & 0.5293 & 0.1406 & 0.3301 \\ 0.5098 & 0.2305 & 0.0400 & 0.2197 \end{bmatrix}$$

Let a scheduling problem $P(M, F)$ represent the scheduling of traffic matrix $M$ in a frame of duration $F$ time-slots. The scheduling problem $P(M, F)$ can be viewed as a root of a binary tree, which in turn creates two smaller scheduling problems $P(A, F/2)$ and $P(B, F/2)$. Using the proposed algorithm, the decomposition of matrix $M$ into 2 matrices $A$ and $B$ is shown in Eq. (3). Observe that matrix $M$ is partitioned relatively evenly over matrices $A$ and $B$. Furthermore, when matrixes $A$ and $B$ are divided by $F/2$, the resulting matrices are doubly substochastic or stochastic, ie they represent smaller instances of the general scheduling problem. It will be established that repeated application of the recursive fair decomposition will result in low-jitter transmission schedules with bounded delay and jitter.

$$M = \begin{bmatrix} 389 & 0 & 225 & 410 \\ 113 & 246 & 614 & 51 \\ 0 & 542 & 144 & 338 \\ 522 & 236 & 41 & 225 \end{bmatrix} = (3)$$

$$\begin{bmatrix} 195 & 0 & 112 & 205 \\ 56 & 123 & 307 & 26 \\ 0 & 271 & 72 & 129 \\ 261 & 118 & 21 & 112 \end{bmatrix} + \begin{bmatrix} 194 & 0 & 113 & 205 \\ 57 & 123 & 307 & 26 \\ 0 & 271 & 72 & 129 \\ 261 & 118 & 20 & 113 \end{bmatrix}$$

In general, the traffic arriving at a packet-switched IP router consists of a mixture of *Guaranteed-Rate (GR)* traffic with rate and delay specifications, and *Best-Effort (BE)* traffic. Typically, GR traffic is a small fraction of the total load. Once the GR traffic has been scheduled within a frame, the frame may be under-utilized. This unused switching capacity can be used to transfer BE traffic using any existing hardware-based dynamic cell scheduling algorithm. Alternatively, bandwidth for BE traffic can be provisioned between IO pairs in the rate matrix, as required by the service provider. This approach of provisioning BE traffic will eliminate the need for hardware-based dynamic schedulers and will also limit denial of service attacks caused by targeted congestion at a single router.

This paper is organized as follows. Section II includes a summary of prior work. Section III formulates the GR scheduling problem. Section IV proposes a fast and relatively fair method to route permutations in 3-stage Clos rearrangeable network. Section V presents results for scheduling GR traffic in 16x16 IQ switches with low delay jitter. Section VI establishes theoretical bounds on the delay and jitter. Section VII contains concluding remarks.

## II. Prior Work

TSA and GR schemes exploit two classic underlying theories from the field of graph theory and combinatorial mathematics: (1) An integer matrix with a maximum row or column sum of $M$ can be expressed as the sum of $M$ permutation matrices [16]; and (2) A doubly substochastic or stochastic matrix can be decomposed into a convex set of permutations matrices and weights. TSA and GR schemes can generally be grouped into 3 classes of combinatorial problems, those based upon graph matching or colouring algorithms, those based upon routing permutations in a rearrangeable network, and those based upon decomposing a traffic rate matrix into a set of constituent permutation matrices and associated weights. A table summarizing several TSA and GR algorithms is shown in Table I. Some of these entries will be briefly summarized. All the results in Table I assume that incoming traffic is first shaped by a token bucket traffic shaper to limit the burstiness to some known upper value.

### A. The Slepian-Duguid Theorem

A circuit-switch is said to be *'rearrangeably nonblocking'* if a new connection between idle IO ports can always be established, although some existing connections may have to be rearranged [17][18][19]. Slepian and Duguid established that the generalized 3-stage Clos network in Fig. 2 is rearrangeably nonblocking. The switches in the first and third

TABLE I
SUMMARY OF LOW-JITTER GUARANTEED-RATE SCHEDULING ALGORITHMS

| Authors | Time Complexity | Num. Matchings | Speedup | Service Lag Bound | IDT Jitter Bound |
|---|---|---|---|---|---|
| Slepian-Duguid 1960 [16] | $O(N^{4.5})$ | $O(N^2)$ | 1.0 | $O(N)$ | $O(N^2)$ |
| Inukai, 1978 [20] | $O(N^{4.5})$ | $O(N^2)$ | 1.0 | $O(N^2)$ | $O(N^2)$ |
| Weller, Hajek, 1999 [21] | $O(N^{4.5})$ | $O(N^2)$ | - | $O(\alpha N - 1)$ | $O(N^2)$ |
| Chen, Chang, Huang (BVN)1999 [24][25] | $O(N^{4.5})$ | $O(N^2)$ | 1.0 | $O(N^2)$ | $O(N^2)$ |
| Towles, Dally 2003 [23] | $O(N^{4.5})$ | $O(N^2)$ | 2 | $O(N^2)$ | $O(N^2)$ |
| Kedliam, Lakshamn, Stilliadis, 2003 [14] | NP | $O(N^2)$ | $logN$ | $O(N^2)$ | $O(N^2)$ |
| Koksal, Gallager, Rohrs 2004 [12] | $O(N^k)$ | $O(N^2)$ | $1 \leq s \leq 2$ | $O(\alpha N/(S/(S-1)))$ | $O(N^2)$ |
| Mohanty, Bhuyan 2005 [26] | $O(N^k)$ | $O(N^2)$ | $> 1$ | - | - |
| This paper | $O(NFlogNF)$ | $O(F)$ | 1 | $O(logNF)$ IIDT | 4 IIDT |

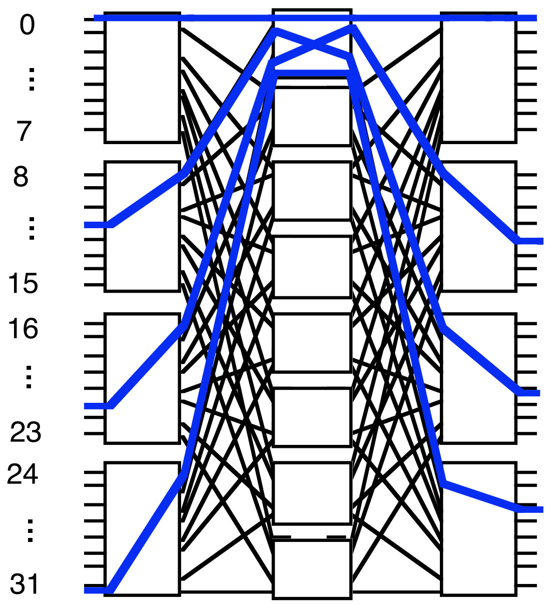


Fig. 2. Rearrangeable 3-stage Clos network (8,4,8).

stages can be represented as vertices in a bipartite graph, and the permutation to be realized reflects the edges. Using Hall's theorem on Systems of Distinct Representatives [16], they established that a sequence of matchings can be found to saturate the bipartite graph. Each matching is routed through one of the middle stage switches, after which the IO ports and relevant middle stage switch are removed from further consideration. The induced bipartite subgraph is then a smaller instance of the same problem. By induction, they established that the 3-stage Clos network is rearrangeably nonblocking. There is no notion of QoS or delay minimization in their problem formulation. Over the years, many algorithms have been developed to route permutations in the 3-stage Clos rearrangeable network.

The Clos network has been thoroughly studied over the last 50 years and has been used for space-domain (S) circuit-switching, Time-Space-Time (TST) switching, and multirate circuit-switching over the last 5 decades [19,20]. According to Mellon and Turner [20]: *"We can also expect that the Clos networks will retain their central place in the design of high performance switching systems of all kinds, and that the intellectual framework created to model their performance will continue to develop and evolve to meet the needs of new technologies and applications."* In this paper, we extend the classic Clos theory to the new problem of low-jitter scheduling in packet-switched IP routers.

## B. The Birkhoff Von Neuman (BVN) Matrix Decomposition

A method for calculating transmission schedules for packet-switched IP routers based on Birkhoff Von-Neumann (BVN) stochastic matrix decomposition was introduced in [24,25]. Let $R$ be a traffic rate matrix for an $N \times N$ crossbar switch, where $R(i,j)$ represents the traffic requirement between IO pair $(i,j)$. Under the following 2 conditions, $\sum_{i=1}^{N} R(i,j) \leq 1, \forall j$, and $\sum_{j=1}^{N} R(i,j) \leq 1, \forall i$, then there exists a set of positive numbers $\theta_k$ and permutation matrices $P_k, k = 1, ..., K$ for some $K \leq N^2 - 2N + 2$ that satisfies the following two equations: $R \leq \sum_{k=1}^{K} \theta_k P_k$ and $\sum_{k=1}^{K} \theta_k = 1$. The first 2 conditions imply that no IO port is overloaded, ie the traffic matrix is admissible. For an NxN crossbar switch, the BVN algorithm finds $O(N^2)$ maximum size matchings each requiring $O(N^{2.5})$ time, for a resulting complexity of $O(N^{4.5})$ time. Once the decomposition is computed, the matrices must be scheduled such that the permutation matrix $P_k$ appears proportional to its weight $\theta_k$, for $k = 1...K$. There is no notion of fairness in the BVN decomposition and $O(N^2)$ matrices may be generated, leading to potentially lengthy delays. According to [12], 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."*

Furthermore, according to [14]: *"Therefore, independent of the type of algorithm used to schedule the permutation matrices, there is no control on when individual entries in the rate matrix will be scheduled. It is possible to derive bounds on the jitter, but it is not possible to ensure that the jitter is low... The jitter problem becomes severe when the number of ports is large. The BV decomposition, therefore, results in poor jitter performance especially when there is a large number of ports in the switch."*

### C. Related Prior Work

In [13,14] the delay jitter minimization problem is formulated as an NP-Hard integer-programming problem. A *greedy low-jitter decomposition (GLJD)* algorithm with complexity $O(N^3)$ time is then proposed. After the decomposition, the permutation matrices and associated weights are scheduled to minimize the jitter between all IO pairs. The resulting schedule requires a worst-case speedup of $O(logN)$ and achieves throughputs of up to 80%.

Another stochastic matrix decomposition algorithm was introduced in [12]. In this algorithm, a traffic rate matrix is quantized and then decomposed into a convex set of permutation matrices and weights, which must then be scheduled. With speedup $S = 1+sN$ between 1 and 2, the maximum *'Service Lag'* over all IO pairs is bounded by $O((N/4)(S/(S-1)))$ time-slots. The speedup directly affects the QoS provided by the switch. According to [12]: *"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."* The authors present experimental results in their paper. For a speedup approaching 2 the service lag does not exceed roughly $N/2$ time-slots, whereas is can go as high as $O(N^2)$ time-slots when no speedup is allowed. For a 256x256 switch with a speedup of 2, the service lag bound is 128 time-slots, whereas with unity speedup it can be as high as 65,000 time-slots.

Another greedy stochastic matrix decomposition algorithm was proposed in [27]. This decomposition algorithm also yields a convex set of permutation matrices and associated weights which must be independently scheduled, as in prior methods. The algorithm is relatively quick but it cannot guarantee 100 % throughput or short-term fairness. The authors establish a jitter bound, but their bound grows as the switch size $N$ increases. The authors identify an open problem: *"to determine the minimum speedup required to provide hard guarantees, and whether such guarantees are possible at all."*

## III. GR Scheduling Problem

The Guaranteed-Rate traffic requirements for a packet-switched NxN crossbar switch can specified in a doubly substochastic or stochastic traffic rate matrix $\Lambda$ , as shown in Eq. 4. Each element $\lambda_{i,j}$ denotes the GR between IO pair $(i, j)$.

$$\Lambda = \begin{bmatrix} \lambda_{0,0} & \lambda_{0,1} & ... & \lambda_{0,N-1} \\ \lambda_{1,0} & \lambda_{1,1} & ... & \lambda_{1,N-1} \\ ... & ... & ... & .... \\ \lambda_{N-1,0} & \lambda_{N-1,1} & ... & \lambda_{N-1,N-1} \end{bmatrix} \quad (4)$$

where $\sum_{i=0}^{N-1} \lambda_{i,j} \leq 1$ and $\sum_{j=0}^{N-1} \lambda_{i,j} \leq 1$.

Define a new quantized traffic rate matrix $R$ where the traffic rates are expressed as an integer number of requested time-slots within a frame. The minimum quota of reservable bandwidth is one time-slot within a frame, representing $(1/F)$ of the line rate.

$$R = \begin{bmatrix} R_{0,0} & R_{0,1} & ... & R_{0,N-1} \\ R_{1,0} & R_{1,1} & ... & R_{1,N-1} \\ ... & ... & ... & .... \\ R_{N-1,0} & R_{N-1,1} & ... & R_{N-1,N-1} \end{bmatrix} \quad (5)$$

where $\sum_{i=0}^{N-1} R_{i,j} \leq 1$ and $\sum_{j=0}^{N-1} R_{i,j} \leq 1$.

The following notations related to scheduler performance will be adopted. Several of the following definitions have been adapted from [12].

**Definition:** A *'Frame transmission schedule'* of length $F$ is a sequence of permutation matrices (or permutation vectors) which define the crossbar switch configurations for $F$ time slots within a scheduling frame. Given a line-rate $L$, the frame length $F$ is determined by the desired minimum allotment of bandwidth = $L/F$. For example, to set the minimum quota of reservable bandwidth to 3 % of the line-rate, set $F = 32$. To set the minimum quota of reservable bandwidth to 0.1 % of the line-rate, set $F = 1K$.

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

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

**Definition:** The *'Service Lag'* of a traffic flow between input port $i$ and output port $j$, at time $t$ given a frame transmission schedule of length $F$, denoted $Lij(t)$, equals the difference between the requested service prorated by the time $t$ within the current frame, and the received service, ie $Lij(t) = (mod(t, F)/F) \cdot R(i, j) - Sij(t)$. The *'Normalized Service Lag'* is defined as the service lag divided by the $IIDT$.

**Definition:** The row and column norms of a quantized matrix are defined as follows;

$$\|R(:, j)\| = \sum_{i=0}^{N-1} R(i, j) \leq F$$

$$\|R(i, :)\| = \sum_{j=0}^{N-1} R(i, j) \leq F (6)$$

The following notations related to multistage switching networks will be adopted.

**Definition:** Let vector elements $I(i)$ and $O(i)$ represent input port $i$ and output port $i$ in an NxN crossbar switch, for $0 \leq i < N$ . In an IQ switch, each input port $i$ also contains $N$ Virtual Output Queues, $VOQ(i, j)$, one for each output port $j$ for $0 \leq j < N$, as shown in Fig. 1.

**Definition:** Given a multistage switching network, let $S(i, s)$ represent switch $i$ in stage $s$, for $0 \leq i < N/2$ and $1 \leq s \leq 2log_2N - 1$. Let $ip(i, s)$ represent input pin $i$ in stage $s$, and let $op(i, s)$ represent output pin $i$ in stage $s$, for $0 \leq i < NF$ and $1 \leq s \leq 2log_2N - 1$.

**Definition:** Let vector $\pi w(i)$ represents the wiring pattern (permutation) between the vector of output pins in stage $s$ denoted $op(:, s)$, and the vector of input pins in stage $s + 1$ denoted $ip(:, s + 1)$, for $1 \leq s \leq 2log_2N - 1$.

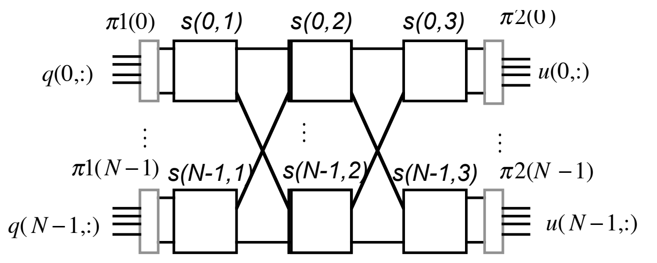


Fig. 3. Time-space-time switching problem formulation.

### *A. Scheduling Problem Formulation, Time-Space-Time (TST) Switch*

Let all the match requests originating at input port $i$ given a traffic rate matrix $R$ in a scheduling problem $P(R, F)$ be ranked in a canonical order, first by the input port index $i$ and then by a unique match number (or rank) within the range $1...R(i, j)$. A match request associated with input port $i$ can thus be uniquely identified by introducing a first canonical set $Q$ with elements $q(i, j)$, where $0 \leq i < N$, and $0 \leq j < \|R(i, :)\|$. Let all match requests associated with output port $j$ also be ranked in a second canonical order, first by the output port index $j$ and then by their unique rank within the range $1...R(i, j)$. A match request associated with output port $j$ can thus be uniquely identified by a second canonical set $U$ with elements $u(j, k)$, where $0 \leq j < N$, $0 \leq k < \|R(:, j)\|$ .

Let the notation $(F, N, F)$ denote a 3-stage rearrangeable circuit-switched network with $NF$ IO pins, with $N$ switches of size $F \times F$ in the 1st and 3rd stages, and $F$ switches of size $N \times N$ in the middle stage. (This notation is slightly different from the classic notation of a symmetric Clos network, but it allows for the specification of recursive constructions which will be useful later.) Fig. 2 shows a general 3-stage Clos network denoted $(8, 4, 8)$. According to the Slepian-Duguid theorem, the 3-stage Clos network is rearrangeably nonblocking i.e., it is possible to find a routing to satisfy a given permutation under the following conditions: (1) no IO port is overloaded, (2) the number of middle stage switches is equal or greater than the size of the switches in the 1st and 3rd stages, ie $F \geq N$.

### *B. The Linear Matrix-to-Permutation Mapping*

Fig. 3 illustrates how the quantized traffic rate matrix is mapped onto a permutation. Each input port $I(i)$ of the switch will reserve up to $F$ time-slots within a frame and is represented by one $FxF$ switch in stage 1 of the 3-stage Clos network. Let the $\|R(i, :)\| < F$ match requests associated with $I(i)$ be mapped onto the input pins of first stage switch $S(i, 1)$, in an order determined by a small permutation of length $F$. Similarly, each output port $O(j)$ of the crossbar switch will reserve up to $F$ time-slots within a frame, and is represented by one $FxF$ switch in the 3rd stage. Let the $\|R(:, j)\| < F$ match requests destined for $O(j)$ appear at the output pins of the third stage switch $S(j, 3)$ in an order determined by a small permutation of length $F$.

Each match request in matrix $R$ appears once in sets $Q$ and $U$. Define $N$ small permutations of length $F$ denoted $\pi1(0), ..., \pi1(N-1)$, which can be concatenated to form one large permutation $\Pi(1)$ of length $NF$. Similarly, define $N$ small permutations of length $F$ denoted $\pi2(0), ..., \pi2(N-1)$, which can be concatenated to form one large permutation $\Pi(2)$ of length $NF$. The set $Q$ is mapped onto the set of input pins $i(0...NF-1, 1)$ by permutation $\Pi(1)$, and the set $U$ is mapped onto a set of output pins $o(0...NF-1)$ by permutation $\Pi(2)$, as shown in Fig. 3. These mappings imply the generation of a new permutation $\Pi$ of length $NF$ from the traffic rate matrix, since each match request in the matrix appears once in sets $Q$ and $U$. Using the *'Random Mapping' (RM)* algorithm, the small permutations $\pi1(:)$ and $\pi2(:)$ are selected at random. Using the *'Linear Mapping' (LM)* , the small permutations are linear. The permutation generated by the mapping algorithm is then routed through the 3-stage rearrangeable network to determine a stochastic matrix decomposition. The $RM$ and $LM$ algorithms are summarized as $RM(Q, U) \rightarrow ((i(:, 1), o(:, 3))$ and $LM(Q, U) \rightarrow ((i(:, 1), o(:, 3))$ respectively.

Many algorithms have been proposed for routing permutations in the 3-stage rearrangeable Clos network over the last few decades, ie see [28-38]. The combinatorial problem is non-trivial, as evidenced by the number of papers addressing the topic. Many algorithms require backtracking due to the need to rearrange previously established connections, which adds complexity to the algorithms. Some nonbacktracking algorithms have been proposed, but they are generally complex.

## IV. Proposed Recursive Fair Stochastic Matrix Decomposition

In this paper, we first propose a relatively efficient, recursive and fair nonbacktracking algorithm for routing permutations in a 3-stage rearrangeable Clos network, by transforming the problem to routing a permutation in a specific binary rearrangeable network with $2logN - 1$ stages of binary nodes. A binary network is defined as one composed only of 2x2 or binary switches. A weaker result on routing permutations in Clos networks was presented by Andresen [30]. However, Andresen does not establish a topological equivalence between the 3-stage Clos network and a binary rearrangeable network which uses a Perfect-Shuffle permutation, and does not characterize the permutations realizable by this particular binary network. To achieve our main results for packet-switched IP routers, we must formally characterize the routing of certain classes of permutations (called *'conforming permutations'*) through a Perfect-Shuffle based Benes network, and we must establish the equivalence of this network to the 3-stage Clos network.

In Fig. 4, the 3-stage Clos switching network of Fig. 3 has been replaced by a similar topology, wherein each $F \times F$ switch in the first stage of Fig. 3 has been replaced by an $F \times F$ *Baseline* binary switching network (enclosed by dotted lines). An 8x8 Baseline binary network is shown in Fig. 4 and is represented using the notation $B(2, 3)$, where the 1st index is the switch size, and the 2nd index is the number of stages. Each $F \times F$ switch in the third stage of Fig. 3 is replaced by an *Inverse Baseline* network denoted $B^{-1}(2, 3)$. The resulting topology in Fig. 4 is a 7-stage switching network, denoted as a $(B(2, 3), 4, B^{-1}(2, 3))$ network.

The 7-stage network in Fig. 4 has far fewer discrete states compared to the 3-stage Clos network of Fig. 3. Each degree-$k$

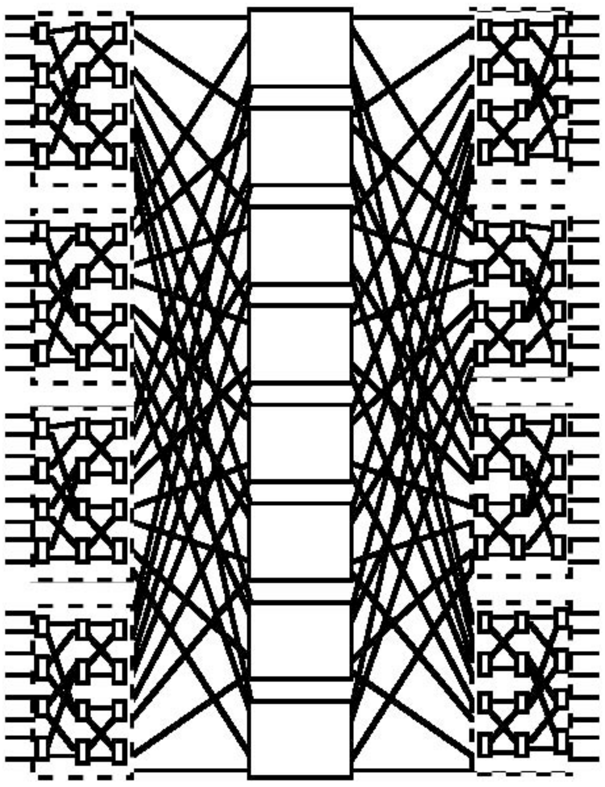


Fig. 4. A 7-stage rearrangeable network.

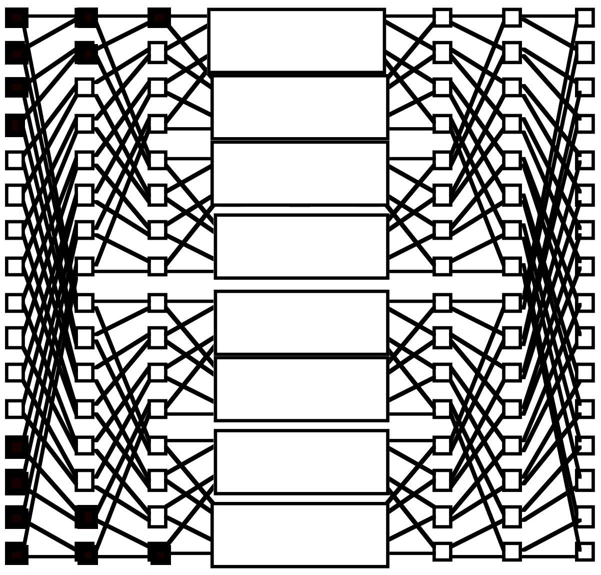


Fig. 5. A 7-stage rearrangeable network (partially expanded Benes network).

crossbar switch in Fig. 3 has $k$ factorial discrete states respectively. The number of discrete states realized by the network in Fig. 3 is thus $(8!)^8(4!)^8 \approx 10^{48}$. In contrast, each 2x2 switch in the network in Fig. 5 has 2! states, and the number of discrete states in the network is $2^{12x8}(4!)^8 \approx 10^{15}$. Due to the significant reduction in states, it is not immediately clear that the network in Fig. 4 is rearrangeably nonblocking. However, the network in Fig. 4 can be topologically transformed to the network in Fig. 5, and the network in Fig. 5 is observed to be a partially expanded Benes network, which is known to be rearrangeable. The $N \times N$ binary Benes network can be represented using the notation $(2, N/2, 2)$, where the middle stage switches of size $N/2xN/2$ are constructed in the same manner using recursion. The Benes network in Fig. 5 uses a Perfect-Shuffle wiring permutation between the 1st and 2nd stages.

There are many binary rearrangeable networks conforming to the recursive Benes construction $(2, N/2, 2)$. Specifically, there are a large number of permutations which may be used to connect the output pins of the first stage to the input pins of the second stage, and this freedom applies recursively. The majority of these Benes-family topologies are not topologically equivalent nor functionally equivalent. For example, in a $32 \times 32$ Benes network there are $(16!)^2$ different choices of permutations to link the 1st and 2nd stages. The following results on the topological equivalence of all *'strict-buddy banyan networks'* were presented in [47] and are summarized. Analogous results for buddy networks are presented in [18].

**Definition:** A radix-$b$ *'strict-buddy banyan'* network is a multistage network constructed with radix-$b$ switches with a unique path between every input port and every output port with at least 2 stages of switches, such that all the switches in each stage can be partitioned into disjoint sets of $b$ switches, such that the $b$ switches in every set share the same $b$ successor switches in the following stage. Define a *'factor'* as a $b^n \times b^n$ multistage banyan network or a single $b^n \times b^n$ crossbar switch.

**Proposition 1:** All higher radix ($b^n \times b^n$) strict-buddy banyan networks, for $b \geq 2$, are topologically equivalent.

**Proposition 2:** Any strict-buddy $b^n \times b^n$ banyan network can be topologically rearranged into an equivalent network with two stages of factors, with factors of size $b^s \times b^s$ in the first stage, and factors of size $b^{n-s} \times b^{n-s}$ in the second stage, $1 \leq s < n$, with a radix-$b^s$ *'Perfect Shuffle'* wiring permutation between the 2 stages (defined ahead).

**Proposition 3:** Every switch in stage $j$ of a $b^s \times b^s$ banyan network, for $0 \leq j < log_2 N$ , is the root of a *'fan out tree'* which spans a subset of output pins in each stage $k$, $j \leq k$; The fan-out tree can reach exactly $b^{k-j}$ output pins in stage $k$, and it can reach all $b^{n-j}$ output pins in stage $n = logN - 1$. Furthermore, every node in stage $j$ of a $b^s \times b^s$ banyan network, $0 \leq j < log_2 N$, is the root of a *'fan-in tree'* ; the fan-in tree can reach exactly $b^{j-i}$ input pins in stage $i$, $i \leq j$, and it can reach all $b^{j+1}$ input pins in stage 0.

In the Baseline topology of Fig. 6, a permutation of wires connects the output pins of the first stage of binary switches to the input pins of the second stage of binary switches, called the binary *'Perfect-Shuffle'* permutation. Any linear sequence with an even number of elements can be viewed as forming 2 halves of a deck of cards, ie the sequence $0 \ldots 7$ can be viewed as the concatenation of two half sequences $0 \ldots 3$ and $4 \ldots 7$. A binary Perfect-Shuffle of the original sequence yields the sequence (0 4 1 5 2 6 3 7), ie the elements of each half sequence are perfectly interleaved as if a perfect binary shuffle has occurred.

A binary number in the range $0 \ldots N-1$ can be represented with $n$ binary bits, $b_{n-1}b_{n-2}...b_1b_0$ , where $n = log_2 N$. The

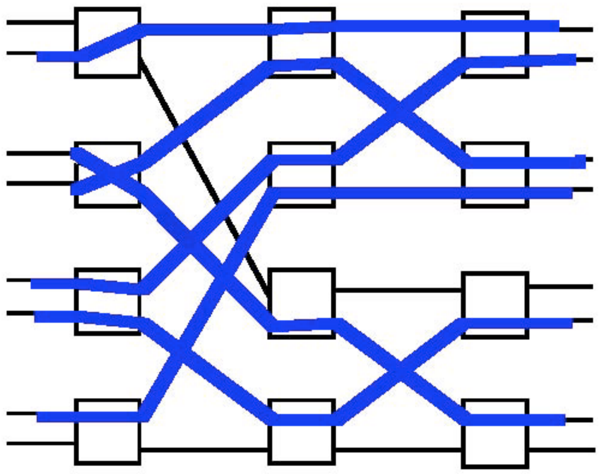


Fig. 6. An 8x8 baseline network.

binary Perfect-Shuffle is given by the mapping in Eq. (7), and the binary *'Inverse Perfect-Shuffle'* permutation is given by the mapping in Eq. (8). Using the Inverse Perfect-Shuffle, the sequence (0 1 2 3 4 5 6 7) maps onto the sequence (0 2 4 6 1 3 5 7). The radix-4 Perfect-Shuffle is given by the mapping in Eq. (7), wherein the digits $b_i$ are base 4 digits.

$$P(b_{n-1}b_{n-2}....b_1b_0) = b_0b_{n-1}b_{n-2}...b_1 \quad (7)$$

$$P(b_{n-1}b_{n-2}....b_1b_0) = b_{n-2}b_{n-3}...b_1b_0b_{n-1} \quad (8)$$

Due to the topological equivalence between the generalized 3-stage Clos network as shown in Fig. 3 and the partially expanded Benes network as shown in Fig. 5, the routing of a given permutation through any 3-stage Clos network can be found by routing an equivalent permutation through the partially expanded Benes rearrangeable network as shown in Fig. 5. The solution for one network can yield the solution for the other network through a simple transformation. Furthermore, a more general statement can be made: The routing of a given permutation through any 3-stage Clos network can be found by routing an equivalent permutation through a partially expanded *Binary Rearrangeable Network* (BRN) as shown in Fig. 5; the wiring pattern between stages need not be a Perfect-Shuffle permutation. Therefore, an efficient non-backtracking algorithm for routing a given permutation $P$ in a 3-stage Clos network can be found as in the following Theorem, whose proof follows from the propositions 1-3.

**Theorem 1:** A setting of the switch states in a 3-stage Clos rearrangeable network $(M, N/M, M)$, for $N$ and $M$ powers of 2, can be determined from the settings of the switch states in a binary rearrangeable network $(2, N/2, 2)$, consisting of the concatenation of a strict-buddy banyan network and its inverse network, with the two innermost stages merged into one stage.

An efficient algorithm to determine the switch settings is as follows: (1) Transform the given permutation $P$ to be realized in the 3-stage Clos network to a new permutation $P$' to be realized in the partially expanded BRN; (2) route $P$' through the BRN; (3) transform the partially expanded BRN into a topologically equivalent network constructed with 3 stages of factors, where the sizes of the factors in stages 1 and 3 equals the size of the switches in stages 1 and 3 of the 3-stage Clos network; (4) recover the states of all switches in the Clos network from the factors of the BRN. This process is illustrated next.

If we choose the partially expanded Benes topology in Fig. 5 as the BRN, then we may determine the switch states directly without applying any transformation (equivalently, the transformation is the identity permutation). In Fig. 5, consider the fan-in trees into the middle stage of factors, and fan-out trees from the middle stage of factors. Let the stages be labeled $1...logN$. By Proposition 3, every input pin of a middle stage switch/factor is the root of a fan-in tree which spans $F$ input pins in stage 1. The nodes belonging to two fan-in trees are shown in bold in Fig. 5. Due to the unique path property stated in the definition of a strict-buddy banyan network, these input pins in stage 1 cannot appear in the fan-in tree of any other input pin incident to the same middle stage switch/factor. Using the same argument, every output pin incident to a middle stage switch/factor spans a unique set of $F$ output pins in stage $log_2N$. Therefore, the routing of a permutation through the topology in Fig. 5 is equivalent to routing the permutation through the 3-stage Clos network: the states of the middle stage switches/factors are identical.

## *A. Equivalence to Recursive Fair Stochastic Matrix Decomposition*

A more powerful observation can also be made. Referring back to the TST switching problem formulation in Fig. 3, we now add the constraint that all the match requests from an input port $I(j)$ for one output port are placed in a contiguous set of input pins $\{i(k,1)|jF \leq k \leq (j+1)F-1\}$ in stage 1 of the BRN. This constraint places restrictions on the permutation to be routed. Observe that for every pair of match requests occupying adjacent input pins $i(j,1)$ and $i(j+1,1)$ for even $j$ in stage 1, exactly one match request must be routed through the top middle stage $N/2xN/2$ switch in the Benes network $(2, N/2, 2)$, while one match request must be routed through the bottom middle stage switch. Hence, by establishing all the match requests associated with $R(i,j)$ to appear at stage 1 in contiguous input pins of the Benes network by using the LM algorithm, it follows that the match requests will be split *'relatively fairly'* amongst the first and second halves of a frame schedule. This issue is discussed in more depth in Section V.

## *B. The LBL Algorithm*

By theorem 1 we may route a permutation through a BRN to determine a set of admissible switch states for a 3-stage Clos network. The traditional looping algorithm for routing permutations in a Benes network can be used to route the permutation [30]. The looping algorithm has many degrees of freedom and can result in a large number of admissible routings for one given permutation. We now add constraints to the looping algorithm so that each permutation has one unique routing. Define the *Linear-Biased-Looping (LBL)* algorithm with the following restrictions: When a new loop is started, the *LBL* algorithm selects the first unmatched input pin with a valid match request to start the new loop. (The first input pin has the lowest index.) Furthermore, the next match request to be routed in a loop consistently selects the upper path through the rearrangeable network, if that path is free. Otherwise, it selects the lower path. Define the *Random-Looping (RL)* algorithm as follows: When a new loop is started, the *RL* algorithm selects an unmatched input pin with a valid match request at random, to start each new loop. Furthermore, the next match request to be routed randomly selects between the upper and lower paths through the rearrangeable network if both paths are free, otherwise it uses the only available path. The *RL* algorithm will yield many admissible routings for a permutation.

### C. Results for the RFSMD Algorithm

One thousand randomly-generated doubly-stochastic traffic rate matrices were generated. Each matrix was mapped into a permutation using the $LM$ algorithm, which was routed through a BRN using the $LBL$ algorithm. The low-jitter frame transmission schedule for a packet-switched crossbar switch was then recovered from the states of the middle stage factors of the BRN, as specified in Theorem 1. Table II illustrates the results of the decomposition of the 1,000 randomly-generated fully-saturated 16x16 traffic matrices with $F = 1024$. Each matrix represents 256 simultaneous low-jitter GR traffic flows to be scheduled. All 1,000 matrices represent 256,000 flows to be scheduled between IO pairs, representing 16,384,000 cells to schedule. Each transmission schedule was processed to record the min/max observed IDTs between 2 consecutive cells in a GR flow, and the min/max Service Lag. By convention, a negative Service-Lag represents a *'Service-Lead'*.

Define a class $'j'$ of flows as containing all flows with exactly $j$ match requests per frame. The results for 10 classes of flows are reported in Table II. Each class of flow has an associated IIDT. The mean observed IDTs from the transmission schedules are listed in column 3 of the table. In all classes, the mean observed IDT equals the IIDT, i.e., the frame transmission schedule results in: (1) every flow receiving its guaranteed rate, and (2) every flow maintaining an average IDT equal to the ideal IDT. For example, the class of flows with 60 matches per frame has an IIDT equal to $F/60 = 17.067$ time-slots. The mean observed IDT in column 3 equals the IIDT. This class experiences a minimum (maximum) IDT of 1 (79) time-slots respectively, corresponding to 0.059 (4.63) IIDTs respectively. The standard deviation of the IDT is 9.02 time-slots corresponding to about $1/2$ an IIDT, indicating that the majority of cells depart within $1/2$ an IIDT of their ideal departure times. The minimum (or maximum) observed Service Lead/Lag is -2.23 (3.75) IIDT. These results were taken over 16 million cell transmissions at 100% load.

The concept of Service-Lag plots in packet-switched networks was first presented by Cruz [4,5]. Each flow between an $(I, O)$ pair has its own ideal service requirement which determines the slope of the Service-Lag curve for that flow. However, it becomes very difficult to plot the Service-Lags for thousands of flows each with different service requirements on the same plot. We overcome this difficulty by plotting *Normalized Service Lead/Lags*, as explained next.

Fig. 7 plots the Normalized Service Lead/Lag for many flows on the same graph. The $X$-axis denotes the normalized

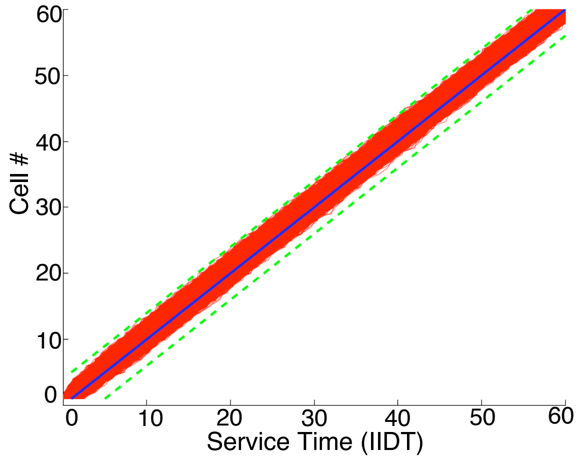


Fig. 7. Service lead-lag, N=16, F=1024.

time expressed in terms of the IIDT for every flow $f$, denoted $IIDT(f)$. The $Y$-axis denotes the cell number. The observed service time for cell $j$ in flow $f$, $c(j, f)$, is denoted $s(j, f)$. The normalized service time for $c(j, f)$ is given by $ns(j, f) = s(j, f)/IIDT(f)$. The Service Lag of $c(j, f)$ is defined as $s(j, f) - j \cdot IIDT$ units. If this service occurs sooner than the ideal service time, a Service Lead equal to $s(j, f) - j \cdot IIDT$ units has occurred (which is negative). In Fig. 7, each single line denotes the normalized service times observed for all cells in one particular flow. The individual service lines for individual flows are indistinguishable, due to the large number of flows plotted on the same graph. However, the minimum and maximum Service Lead/Lags are visible from this graph. According to Fig. 7, the observed Service Lead/Lags are very small, which is consistent with Table II. The mean standard deviation of the Service Lead/Lag is 0.536 IIDT, and the minimum and maximums are -2.85 IIDTs and +4.46 IIDTs respectively. We have performed extensive simulations for many switch and frame sizes, with consistent results.

## V. Bounds on the Jitter and Service Lead/Lag

In this section, bounds on the jitter and service lead/lag through one packet-switched IP switch / router are derived. One proposition and theorem formalize the operation of the recursive fair stochastic matrix decomposition algorithm for larger $N$.

**Definition:** Given an $NxN$ traffic rate matrix $R$ with rates $R(i, j)$ which has been mapped onto a permutation of size $NF$ using the $LM$ algorithm, define a *'Conforming Permutation'* as one in which all match requests belonging to a matrix element $R(i, j)$ appear on contiguous input pins and contiguous output pins. (More specifically, the mappings are $jF + \{R(i, j)|0 \leq i < N, 0 \leq j < N\} \rightarrow \{i(k, 1)|i \cdot F \leq k < jF + R(i, j) < (i + 1) \cdot F\}$ and $\{R(i, j)|0 \leq i < N, 0 \leq j < N\} \rightarrow \{o(k, 3)|j \cdot F \leq k < jF + R(i, j) < (j + 1) \cdot F\}$ .

**Definition:** A *'relatively fair'* partition of $2n$ elements is one where one half receives no less than $(n - 1)$ elements and the other half receives no more than $(n + 1)$ elements. An example of a worst-case relatively fair partitioning is shown

TABLE II
OBSERVED IDT AND SERVICE LEAD/LAG, N=16, F=1024

| Class | IIDT | Observed | Std. Dev. | Min. IDT | Max. IDT | Min. SLL | Max. SLL |
|---|---|---|---|---|---|---|---|
| 30 | 34.133 | 34.133 | 0.524 | 0.029 | 3.13 | -2.15 | 3.66 |
| 33 | 31.030 | 31.030 | 0.522 | 0.032 | 3.74 | -2.38 | 2.39 |
| 36 | 28.444 | 28.444 | 0.537 | 0.035 | 3.80 | -2.47 | 3.75 |
| 39 | 26.256 | 26.256 | 0.544 | 0.038 | 4.27 | -2.32 | 4.46 |
| 42 | 24.381 | 24.381 | 0.537 | 0.041 | 3.94 | -2.85 | 3.46 |
| 45 | 22.756 | 22.756 | 0.544 | 0.044 | 4.09 | -2.14 | 3.84 |
| 48 | 21.333 | 21.333 | 0.539 | 0.047 | 4.22 | -2.2 | 3.75 |
| 51 | 20.078 | 20.078 | 0.547 | 0.050 | 4.28 | -2.09 | 3.59 |
| 54 | 18.618 | 18.618 | 0.543 | 0.053 | 3.69 | -2.85 | 3.78 |
| 57 | 17.956 | 17.956 | 0.529 | 0.056 | 5.12 | -2.85 | 3.78 |
| 60 | 17.067 | 17.067 | 0.528 | 0.059 | 4.63 | -2.23 | 3.75 |

in the first stage of the Baseline network in Fig. 6, where 6 requests in the first stage are partitioned with 4 going to the upper half and 2 going to the lower half.

**Proposition 4:** Given a traffic rate matrix element $R(i,j)$ such that $R(i,j) = 2n$, which has been mapped onto a *Conforming Permutation* of size $NF$ to be routed through the binary rearrangeable Perfect-Shuffle-based Benes topology $(2, N/2, 2)$ shown in Fig. 5 using the $LM$ algorithm, then using the $LBL$ algorithm, the $(2n)$ matches will be split *'relatively fairy'* onto the upper and lower middle stage $(N/2)x(N/2)$ switches, and these match requests will be mapped onto contiguous input and output pins of the middle stage switches, i.e., the permutations to be realized by each switch in the middle stage are also *Conforming Permutations*.

**Theorem 4:** Given the rearrangeably nonblocking network topology $(2, N/2, 2)$ in Fig. 5, and given an $NxN$ traffic rate matrix $r$ with rates $R(i,j)$ which have been mapped onto a conforming permutation of size $NF$, then after routing any $2n$ match requests in a Conforming Permutation which form one or more closed loops using the $LBL$ algorithm, then the remaining induced subgraph, i.e., network consisting of switches with unused pins only and wires between unused pins, is a rearrangeably nonblocking $(N-2n, (N-2n)/2, N-2n)$ network.

Theorem 4 implies that the service a traffic flow will receive will be *essentially equal* in each half of the frame schedule, and that the resulting smaller permutations to be routed in the middle stage of $(N/2)x(N/2)$ switches will be Conforming. Theorem 4 states that after routing one or more loops from a Conforming Permutation in the rearrangeable binary network, the remaining network is also a rearrangeable network. Together, propositions 4 and theorem 4 imply that after routing all the matches in a Conforming Permutation in the rearrangeable $(2, N/2, 2)$ network, the permutations in each middle stage switch form a smaller instance of the same problem, i.e., two smaller Conforming Permutations of size $(N/2)$ to be realized and two smaller rearrangeable networks of size $(2, N/4, 2)$, for which induction can be applied. Theorem 5 (to be presented) will exploit these propositions to bound the jitter and service lead/lag in the proposed resource reservation algorithm.

## A. Jitter Bound

Fig. 8 illustrates the methodology to determine bounds for the worst-case service lead-lag and the worst-case jitter (ie IDT). In Fig. 8, each node in the binary tree represents a scheduling problem $P(M,F)$, with an admissible traffic rate matrix $M$ to be realized within a frame of size $F$. Identify a single flow between IO pair $(i,j)$ given traffic rate $M(i,j)$. Consider the subproblem of scheduling this individual flow within a frame, denoted $P_f(M(i,j), F)$.

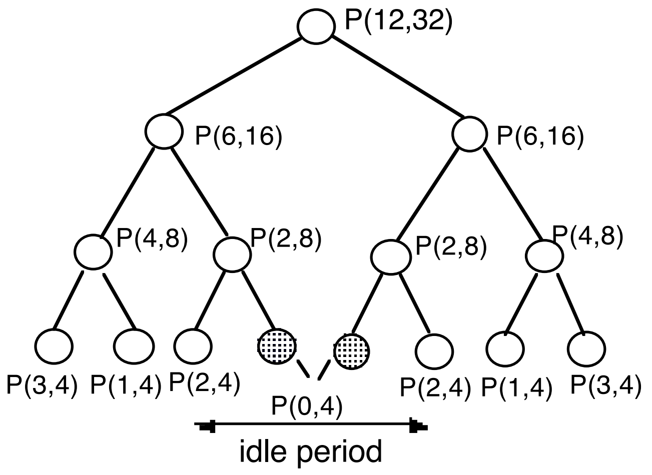


Fig. 8. Binary tree for the recursive scheduling problems.

We first give an informal description of the methodology. The use of the Perfect-Shuffle permutation results in a *'relatively fair'* but not perfectly fair partitioning of one GR traffic rate matrix into 2 GR traffic rate matrices. In Fig. 8, the root node represents a scheduling problem $P_f(12, 32)$, with 12 match requests to be realized in a frame of size $F = 32$ time-slots. In the first partitioning, these matches are split fairly with 6 matches going to each node in level 2 of the tree. In the 2nd level of the tree, the worst-case partitioning has 4 matches going to each of the outer nodes, and 2 matches going to each of the inner nodes. The worst-case idle period occurs at level 4 of the tree, when two adjacent innermost leaves each receive 0 match requests. The two leaves adjacent to the idle nodes must be non-empty due to the relatively fair partitioning, but may have as little as one match request each. Therefore, the worst-case idle period corresponds to a time duration equal to $< 4$ leaf nodes, ie $< 8$ IIDT.

Due to the use of the Perfect-Shuffle permutation, we cannot partition 2 match requests with any fairer resolution, and therefore there is no point to extend this binary tree to levels beyond 4. The maximum Service Lag occurs at the end of a worst-case idle period, and equals one half of the worst-case idle period plus any additional service deficits incurred due to the imperfect partitioning at any ancestors in the tree.

**Theorem 5:** Given a rearrangeably nonblocking network $(2, N/2, 2)$ and a quantized $NxN$ traffic rate matrix with rates $R(i, j)$ which has been mapped onto a Conforming Permutation of size $NF$ using the $LM$ algorithm, after routing all match requests in the Conforming Permutation using the $LBL$ algorithm and extracting the resulting frame schedule, then for each class of flows with $m$ match requests, $1 \leq m \leq F$, the expected IDT will equal the IIDT, and the maximum idle period is bounded, ie $max(IDT) \leq 8$ IIDT. The proof requires an identity and 3 lemmas.

**Identity:** Floor$(a + b)$ = floor $(a)$ + $b$, for all integers $b$.

**Lemma 1:** Given $n$ match requests to be realized in a scheduling problem $P_f(n, F)$, after one *'relatively fair'* partitioning, the number of match requests allocated to each subproblem, denoted $n'$, is equal to at least $n1$ and at most $n2$, where $n1 = \lceil n/2 \rceil - 1$ and $n2 = \lfloor n/2 \rfloor + 1$ :

$$\lceil n/2 \rceil - 1 \leq n' \leq \lfloor n/2 \rfloor + 1 (10)$$

**Proof:** There are 2 cases to consider, $n$ even and $n$ odd. Referring to Fig. 6, for each case there are 2 subcases to consider, wherein the $n$ contiguous match requests start on an even (or odd) pin of the Baseline network. By exhaustively examining all 4 cases, the lemma is seen to be true. •

**Lemma 2:** Given $R$ match requests to be realized in a scheduling problem $P_f(R, F)$, where $R = 2^j a + \delta$ where $\delta = 0, 1, , 2^j - 1$, after $j$ relatively fair partitions, the number of match requests $n$ allocated to each scheduling subproblem (or partition) is equal to at least $a - 1$ and at most $a + 2$, ie $(a - 1) \leq n \leq (a + 2)$, where $a = \lfloor R/2^j \rfloor$ .

**Proof:** The proof is by induction: Assume that lemma 2 is true for $j$ partitions. Consider a leaf node after $j$ partitions, with $n$ match requests. There are 2 extremal cases to consider for the $(j+1)-th$ partition: the *'deficit'* case where $n = a-1$ and the *'surplus'* case where $n = a+2$. For each case there are 2 subcases to consider, the case where $a$ is even and $a = 2a'$, and the case where $a$ is odd and $a = 2a' + 1$.

**Case (1)** Deficit with even $a$ ($n = a - 1$ and $a = 2a'$): By lemma 1, $n' \geq \lceil n/2 \rceil - 1 = a' - 1$ and $n' \leq \lfloor n/2 \rfloor + 1 = a' + 2$. Observe that $a = \lfloor R/2^j \rfloor$ and $a' = \lfloor R/2^{j+1} \rfloor$ and the lemma is true.

**Case (2)** Surplus with even $a$ ($n = a + 2$ and $a = 2a'$): By lemma 1, $n' \geq \lceil n/2 \rceil - 1 = a'$ and $n' \leq \lfloor n/2 \rfloor + 1 = a' + 2$. Observe that $a' = \lfloor R/2^{j+1} \rfloor$ and the lemma is true.

**Case (3)** Deficit with odd $a$ ($n = a-1$ and $a = 2a'+1$): By lemma 1, $n' \geq \lceil n/2 \rceil - 1 = a' - 1$ and $n' \leq \lfloor n/2 \rfloor + 1 = a' + 1$. Observe that $a' = \lfloor R/2^{j+1} \rfloor$ and the lemma is true.

**Case (4)** Surplus with odd $a$ ($n = a+2$ and $a = 2a'+1$): By lemma 1, $n' \geq \lceil n/2 \rceil - 1 = a' + 1$ and $n' \leq \lfloor n/2 \rfloor + 1 = a' + 2$. Observe that $a' = \lfloor R/2^{j+1} \rfloor$ and the lemma is true. In all cases, the Lemma holds true •

**Lemma 3:** Given $R$ matches to be realized in a scheduling problem $P_f(R, F)$, after $j = \lfloor log_2 R \rfloor$ *'relatively fair'* partitions, the number of match requests $n$ allocated to each scheduling sub-problem is equal to at most 4 and at least 0, ie $0 \leq n \leq 4$ .

**Proof.** The proof follows from lemma 2. By lemma 2, $(a - 1) \leq n \leq (a + 2)$. Observe that $a = \lfloor R/2^j \rfloor$ and since $j = \lfloor log_2 R \rfloor$ then $1 \leq a \leq 2$ . Therefore, $0 \leq n \leq 4$. •

Lemma 3 establishes upper and lower bounds on the number of match requests allocated to a scheduling subproblem, equivalently a node in a binary tree, after $j$ partitions. Observe that the expected number of requests allocated per leaf node is $R/2^j$.

**Proof of Theorem 5:** To establish Theorem 5 consider the problem of scheduling any individual flow with $R$ match requests in a frame of size $F$. By lemma 3, after $j = \lfloor log_2 R \rfloor$ partitions the number of match requests allocated to each leaf of the binary tree is between 0 and 4. Furthermore, observe that any node one level up the tree must have at least 1 match request. Therefore, a leaf node with 0 match requests must have a neighboring sibling node with $> 0$ match requests. Therefore, the worst-case idle period must have 2 consecutive leaf nodes containing 0 match requests each. In the worst-case, each neighboring node may have only 1 match request, and this request may occur in such a way to prolong the idle period. By lemma 3, in the worst-case each leaf node corresponds to time duration equal to 2 IIDTs. Therefore, the worst-case idle period is bounded by 8 IIDT time-slots. •

**Theorem 6:** The worst-case service lag is $O(logNF)$ IIDT time-slots.

**Proof:** The worst-case service lag occurs when 2 consecutive leaf nodes have zero requests creating the worst-case idle period. By lemma 3, a service lag in the first half of the idle period must be preceded by a service lead which will negate much of the service lag. Therefore, the worst-case service lag occurs in the last half of the worst-case idle period. The worst-case cumulative service deficit prior to the worst-case idle period is $O(logNF)$ cells, and this occurs when the maximum number of ancestor nodes receive 2 less match requests than in the ideal partition, as established in lemma 3. The worst-case service lag is therefore $O(logNF)$ IIDT time-slots •

## *B. Extensions*

**Theorem 7:** The idle period bound of $K$ IIDT and service lag bound of $O(logNF)$ IIDT also apply to any individual flow which *shares* a VOQ or a transmission link with other flows.

**Proof.** Observe that the match requests associated with any one flow traversing a VOQ will be mapped onto contiguous input pins in the rearrangeable network by the LM algorithm. Therefore, in any relatively fair partition in the RFSMD algorithm, the match requests associated with any one flow will be split relatively evenly over both halves of the BRN. Therefore, lemmas 1-3 and theorems 4 and 5 apply, when the IIDT for the individual flow is used. •

Observe that given the LM algorithm, the match requests associated with any one flow can start on even or odd pins in the rearrangable network. The IDT bound for individual flows can be reduced by using a more complex mapping algorithm. Consider a variation of the LM algorithm called the *'Even-Odd-Mapping'* (EOM) algorithm, which processes all flows in one VOQ with an even number of match requests first, followed by the flows with an odd number of match requests.

**Theorem 8:** The IDT bound for an individual flow which shares a VOQ or a transmission link with other flows is bounded by 4 IIDT, when the EOM mapping algorithm is used.

**Proof:** After any partitioning step, all the match requests associated with any one flow must start their mapping onto an even pin in the rearrangeable network. It follows that the partitioning through the baseline network and lemma 1 are changed so that $\lfloor n/2 \rfloor \leq n1 \leq \lfloor n/2 \rfloor + 1$, lemmas 2 and 3 change accordingly, and the bound in Theorem 5 is reduced to 4 IIDT. •

## C. *Time-Complexity*

The traffic rate matrix for a backbone IP router is incrementally updated by the RSVP, IntServ or DiffServ protocol as GR connections are established, removed or modified in a switch. Relatively small increases to the GR for an IO pair $(i, j)$ can be made by a search of the existing schedule for idle slots. The time complexity of such an incremental change is $O(F)$. A complete recomputation of the schedule requires that the traffic rate matrix be mapped to a permutation, which incurs $max(O(N^2), O(NF))$ time. This mapping can be accomplished incrementally as the traffic rates change, so that this term has negligible effect. The RFSMD algorithm incurs a serial complexity of $O(NFlog(NF))$ time to compute $F$ permutations, or equivalently $O(NlogNF)$ time to compute 1 permutation. This time complexity per permutation is considerably better than those for all existing GR schedulers and all suboptimal maximal matching schedulers. Assuming an 8x8 switch, a link rate of 40Gbps and a frame size of $F = 1K$ time-slots, the scheduling frame rate is 76.3 KHz. However, in an IP router the provisioned traffic rates will change relatively slowly, and the rate at which new schedules are recomputed is expected to be much lower, ie 100 to 1000 times per second. At this rate, bandwidth can even be provisioned for individual video frames in an IPTV stream (at 33 frames per second) or individual talk-spurts in a VOIP stream. The RFSMD algorithm has the same structure as the well-known FFT algorithm. Intel projects the performance of its multi-core processors to be 100,000 MIPS by 2010 (www.intel.com). With this performance, we estimate that a standard laptop multi-core CPU chip can compute new schedules at the rate of 100 KHz, which is considerably faster than the maximum estimated recomputation rate of 100 to 1000 Hz. A multiple processor- chip or FPGA implementation will be considerably faster.

## VI. Experimental Results

A network model was developed and simulated to obtain experimental results, to demonstrate Theorems 4-8. A linear chain of 16 IP routers employing IQ switches was established, as shown in Fig. 9. Each IQ switch has size 8x8. All 8 output links from router $j$ lead to router $j + 1$. The line rate is fixed at 40 Gbps, and the frame length $F$ is fixed at 1K time-slots, such that each time-slot reservation results in a guaranteed bandwidth of 40 Mbps. A traffic model which essentially saturates every IQ switch in the chain was developed. At router 1, 193 traffic flows arrive at the incoming links with an average of 24.125 traffic flows arriving on each input link.

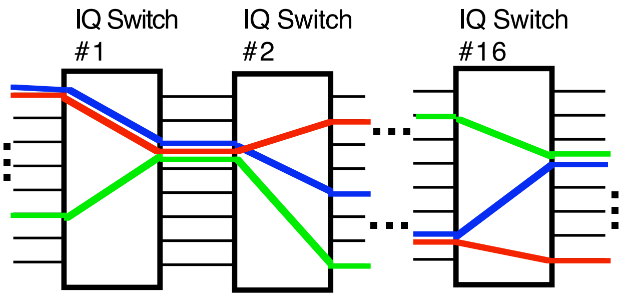
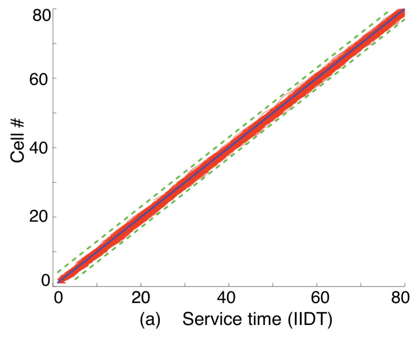


Fig. 9. Linear chain of 16 IQ switches with 193 competing traffic flows.

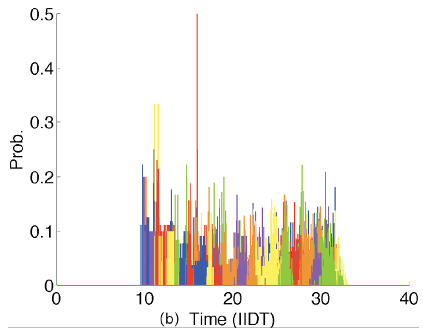
Each traffic flow represents a multimedia stream, with a randomly selected guaranteed traffic-rate in increments of 40 Mbps. The traffic rates on all 193 arriving traffic flows were selected to essentially saturate the input links of the first 8x8 switch; each link has an average load of 99.6 %. To saturate the network, all 193 flows are routed through the linear chain of 16 IQ switches, such that all 193 traffic flows exit the output side of the last switch 16. A backtracking routing algorithm was created to route the traffic flows. Once a path for a flow through the linear chain was found, the path was fixed and a new flow would be routed. Observe that many individual flows are statistically multiplexed onto each link. The number of competing traffic flows traversing each link varies from a low of 20 to a high of 28 flows. Competing flows may share several links, they may diverge, and they may converge on other downstream link(s) again. Each IQ router reserves on average 99.6% of available bandwidth on every IO link, an extremely high loading, while operating at unity speedup. Several network models were developed and simulated with various switch sizes (4x4, . . ., 1Kx1K), various frame sizes $(F = 32, \ldots, 4K)$, various lengths (up to 64 switches), various numbers of flows (up to 4K) and various topologies, all operating at essentially 100% throughput, and all yielded essentially identical results to be described next.

Fig. 10(a) illustrates the observed service lead/lag for each individual traffic flow, as described in Theorem 6. The lead/lag curves for all 193 flows in all 16 switches (ie 3,088 curves) are plotted on the same graph. These curves correspond to the service lead/lag plot of Fig. 7 (in Fig. 7 each flow represents the aggregated traffic flowing through one VOQ). The ideal service curve is the solid 45-degree diagonal. In Fig. 10(a), the dashed lines above and below the main diagonal correspond to service leads/lags of 4 IIDT. According to Fig. 10(a), the observed Service Lead/Lags are within 4 IIDT, consistent with the upper bound in Theorem 6.

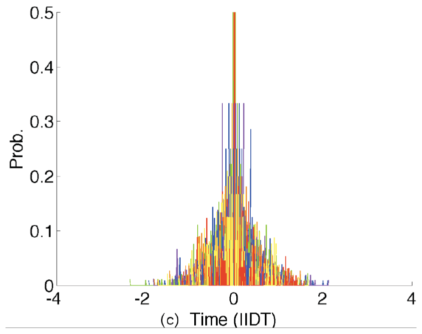
Fig. 10(b) plots the observed end-to-end delay PDF for all 193 flows when leaving switch 16. The end-to-end delay has an average value of 16 IIDT, with a minimum of 10 IIDT and a maximum of 34 IIDT, a relatively wide spread. Fig. 10(c) plots the deviation from the mean end-to-end delay for each flow, for all 193 flows. The mean delays were normalized to 0 on the x-axis, and the deviations are expressed in IIDTs. Observe that every flow exhibits a very small delay deviation about its mean, within 2 IIDT. This graph indicates that every



(a) Service lead and lag curves.



(b) End-to-end delay PDF.



(c) Deviation of end-to-end delay from mean.

Fig. 10.

flow receives a very low end-to-end network-introduced delay jitter, which can be easily filtered out using a playback buffer. Fig. 11(a) plots the observed IDT PDF for all 193 flows when leaving switch 1. Fig. 11(b) plots the experimentally observed IDT PDF for all 193 flows when leaving switch 16. All flows leaving any IQ switch exhibit a bounded IDT jitter which is +/- 4 IIDT time-slots, consistent with theorem 8. Fig. 11(c) plots one cumulative IDT PDF for all flows, which indicates that every cell waits on average 1 perfect IIDT for service, the ideal waiting time.

Fig. 11(d) illustrates the PDF for the number of cells per flow which are queued in any VOQ in any switch. The results for all 193 flows and all 16 IQ switches are presented on one plot, ie 3,088 PDFs are plotted. The average number of buffered cells per flow per router is 1.45. The maximum number of cells buffered per flow was 7 cells, although this is quite rare. We emphasize that these results are based upon a linear chain of IQ switches operating at essentially 100% load, with 193 traffic flows competing for service over 8 incoming links in each of 16 switches. Even at this high load, every end-to-end flow receives essentially-perfect QoS, ie it receives its guaranteed-rate with near-minimal average queueing delays and near-minimal network-introduced delay jitter.

### A. *Buffer Sizing in IP Routers*

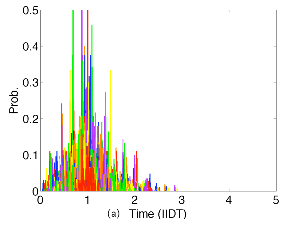
Existing IP routers must use large buffers to keep the transmission pipelines active. Commercial IP routers currently follow a well-established design rule called the *'classical buffer rule'*, where each link requires a buffer of capacity $B = O(C \cdot T)$ bits, where $C$ is the link capacity and $T$ is the round-trip time of the flows traversing the link [39]. This buffer size will generally avoid depleting the queue in an IP router and will keep the transmission pipeline active. Given a 40 Gbps link transporting TCP flows with a 250 millisec round-trip time, then $B$ is roughly five million IP packets [39], or equivalently several tens of millions of fixed-sized cells.

A *'small buffer rule'* was proposed in [39], where $B = O(C \cdot T/N^{1/2})$ , and where $N$ is the number of long-lived TCP flows traversing the router. Using the same parameters, the small buffer size $B$ is roughly fifty thousand IP packets [39] or several hundred thousand cells. More recently, [42] proposed a *'tiny buffer rule'* where $B = O(logW)$, where $W$ is the maximum TCP congestion window size. Using the same parameters, average buffer sizes of between 20-50 IP packets or between several hundred and one thousand cells may suffice if (a) the jitter of incoming traffic at the source node is sufficiently small and bounded, (b) the IP routers introduce a sufficiently small jitter, and (c) 10-15% of the throughput is sacrificed. However, [43][44] have argued that small buffers may cause significant losses, instability or performance degradation at the application layer. Furthermore, the design of a scheduling algorithm for IQ routers with low jitter and unity speedup is in itself a significant unsolved problem [12-38]. In summary, there are significant problems when using existing schedulers and the inherently high jitter TCP flow control protocol in IP networks.

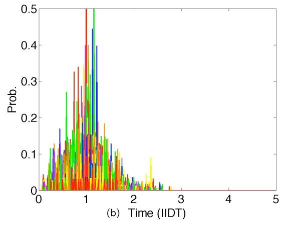
Fig. 11(d) illustrates that the proposed low-jitter RFSMD scheduling algorithm results in extremely low amounts of buffering. Each flow buffers 1.445 cells per IP router on average, several orders of magnitude less buffering than existing IP routers. Rigorous bounds on the buffer requirements are presented in [51].

### B. *Applications*

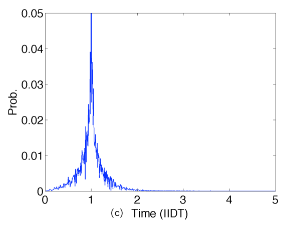
Consider an IP/MPLS domain where variable-size IP packets are fragmented into fixed-size cells upon entry and reconstructed upon exit of the domain, and where the RFSMD algorithm is used to schedule traffic through each router. Each router will only need to buffer a small number of cells in its VOQs, to guarantee that the transmission pipeline never becomes idle. Cells will be transported along the end-to-end path with near minimal queueing delays. If a playback buffer is used at the destination to filter out residual network-introduced jitter, traffic can be delivered with essentially-zero network-introduced delay jitter. In [49], it is shown that supercomputer traffic can be delivered across a Fat-Tree network with essentially-zero network-introduced delay jitter, with near-optimal queueing delays, with 100% load and no speedup. In [50], it is shown that bursty leaky-bucket constrained IPTV traffic can be multicast across the Internet with essentially-zero network-introduced delay jitter, with near-optimal queueing delays, with 100% load and no speedup. In [52][53] it is shown that traffic can be scheduled between Base-Stations in infra-structure based multihop Wireless Mesh Networks with essentially-zero network-introduced delay jitter, with near-optimal queueing delays, with 100% throughput and no speedup. Finally, the RFSMD algorithm can also be used



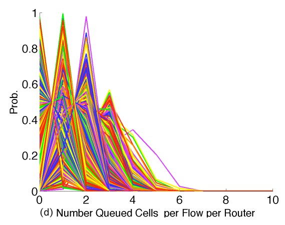
(a) IDT PDF leaving switch #1.



(b) IDT PDF leaving switch #16.



(c) Cumulative IDT PDF, all 193 flows, all 16 switches.



(d) PDF, number queued cells per flow per router.

Fig. 11.

to schedule traffic through a crosspoint buffered crossbar switch with zero or one cell buffer at each crosspoint. Each permutation from the RFSMD algorithm can be processed so that the writes to the crosspoint buffers occur at time-slot t, and the reads from the crosspoint buffers occur in the time-slots t or t+1. In this case, the maximum buffer size at any crosspoint to ensure 100% throughput is 0 or 1 cell, and the same IDT and service lag bounds developed herein apply.

## VII. Conclusions

A low-jitter Guaranteed-Rate scheduling algorithm for packet-switched IP routers has been presented. An IntServ, DiffServ or RSVP protocol can be used to reserve bandwidth and buffer-space for guaranteed-rate traffic in each router along an end-to-end path of IP/MPLS-routers. The proposed RFSMD algorithm can be used to schedule each router to meet rate and delay guarantees for all admissible traffic patterns and all loads up to 100 %. The RFSMD scheduling algorithm naturally supports multicast traffic, which can be specified along with non-multicast traffic in the traffic rate matrix. The algorithm applies to IQ switches without speedup, OQ switches without speedup, combined CIOQ switches without speedup, and to switches with combined input and crosspoint queueing without speedup. Each packet-switched router acts as an essentially-transparent programmable circuit switch for guaranteed-rate traffic, where the average queueing delay per router is bounded by a small number of ideal cell delays (typically several IIDT). The jitter and service lag per router are also bounded by a small number of IIDTs. In a large IP/MPLS network, bounds on the end-to-end delay, jitter and service lag are presented in [51]. When an appropriately-sized playback buffer is employed, essentially-zero delay jitter can be achieved on an end-to-end path in any IP/MPLS network for all shaped traffic flows [51]. The algorithm can be used to schedule low-jitter traffic in crossbar-based packet-switches, satellite packet systems, all-optical packet switches, frame-relay systems, wireless networks and other related systems, to provide near-perfect end-to-end QoS with rate and delay guarantees and essentially-zero jitter. Applications of the algorithm to provision time-sensitive telerobotic control systems such as telerobotically-controlled surgery [1] over a saturated Internet backbone network are presented in [54]. Optimized software and hardware implementations are being developed.

## Acknowledgement

The thoughtful questions of the reviewers are acknowledged. Financial support is acknowledged from the Ontario Centers of Excellence and the Red Wilson/Bell Canada Chair in Data Communications for the testing of the RFSMD algorithm at McMaster University. The algorithm was developed by THS Technologies.

## References

[1] Cisco News Release, "Bell Canada uses CISCO technology to help deliver surgical grade network to power historic telerobotics assisted surgery," Mar. 2003. [Online]. Available: www.cisco.com

[2] Cisco Systems White Paper, "Optimizing video transport in your IP triple play network," 2006. [Online]. Available: www.cisco.com

[3] M. Hluchyi, M. Karol, and S. Morgan, "Input versus output queueing on a space division switch," *IEEE Trans. Commun.*, vol. 35, 1987.

[4] R. Cruz, "A calculus for network delays—part 1: network elements in isolation," *IEEE Trans. Inf. Theory*, vol. 37, no. 1, pp. 114-131, Jan. 1991.

[5] R. Cruz, "A calculus for network delays—part 2: network delay," *IEEE Trans. Inf. Theory*, vol. 37, no. 1, pp. 132-141, Jan. 1991.

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

[7] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated service networks: the multiple node case," *IEEE/ACM Trans. Networking*, vol. 2, no. 2, pp. 137-150, 1994.

[8] V. Anantharam, N. McKeown, A. Mekittikul, and J. Walrand, "Achieving 100% throughput in an input queued switch," *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1260-1267, 1999.

[9] E. Leonardi, M. Mellia, F. Neri, and M. A. Marsan, "Bounds on the delay and queue lengths in input-queued cell switches," *J. ACM*, vol. 50, no. 4, pp. 520-550, July 2003.

[10] S. T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching output queueing with a combined input/output-queued switch," *IEEE J. Sel. Areas Commun.*, vol. 17, no. 6, pp. 1030-1039, June 1999.

[11] N. Nabeshima, "Performance evaluation of a combined input and crosspoint queued switch," *IEICE Trans. Commun.*, vol. E83-B, no. 3, pp. 737-741, Mar. 2000.

[12] C. E Koksal, R. G. Gallager, and C. E. Rohrs, "Rate quantization and service quality over single crossbar switches," in *Proc. IEEE Infocom.*, 2004.

[13] P. Gopya and H. M. Vin, "Generalized guaranteed rate scheduling algorithms: a framework," *IEE/ACM Trans. Networking*, vol. 5, no. 4, pp. 561-571, Aug. 1977.

[14] I. Keslassy, M. Kodialam, T. V. Lakshamn, and D. Stiliadis, "On guaranteed smooth scheduling for input-queued switches," *IEEE/ACM Trans. Networking*, vol. 13, no. 6, Dec. 2005.

[15] M. S. Kodialam, T. V. Lakshman, and D. Stilladis, "Scheduling of guaranteed-bandwidth low-jitter traffic in input-buffered switches," US Patent Application 20030227901, 2003.

[16] M. Hall, *Combinatorial Theory*. Waltham, MA: Blaisdell.

[17] V. E. Benes, *Mathematical Theory of Connecting Networks and Telephone Traffic*. Academic Press, 1965.

[18] F. K. Hwang, *The Mathematical Theory of Nonblocking Switching Networks*," World Scientific Series on Applied Mathematics. New Jersey: 1998.

[19] A. Jajszczyk, "Nonblocking, repackable and rearrangeable Clos networks: fifty years of the theory evolution," *IEEE Commun. Mag.*, pp. 28-33, Oct. 2003.

[20] J. S. Turner and R. Melen, "Multirate Clos networks: 50Th anniversary of Clos networks," *IEEE Commun. Mag.*, pp. 38-44, 2003.

[21] T. Inukai, "An efficient SS/TDMA time slot assignment algorithm," *IEEE Trans. Commun.*, no. 10, pp. 1449-1455, 1979.

[22] T. Weller and B. Hajek, "Scheduling nonuniform traffic in a packet switching system with small propagation delay," *IEEE/ACM Trans. Networking*, vol. 5, no. 6, pp. 813-823, 1997.

[23] A. Hung, G. Kesidis, and N. McKeown, "ATM input buffered switches with guaranteed rate property," in *Proc. IEEE ISCC*, 1998.

[24] B. Towles and W. J. Dally, "Guaranteed scheduling for switches with configuration overhead," *IEEE Trans. Networking*, vol. 11, no. 5, pp. 835-847, Oct. 2003.

[25] W. J. Chen, C.-S. Chang, and H.-Y. Huang, "Birkhoff-von Neumann input buffered crossbar switches," in *Proc. IEEE Infocom*, 2000.

[26] C.-S. Chang, W. J. Chen, and H.-Y. Huang, "On service guarantees for input buffered crossbar wwitches: a capacity decomposition approach by Birkhoff and von Neuman," in *Proc. IEEE iWQoS*, pp. 79-86, 1999.

[27] S. R. Mohanty and L. N. Bhuyan, "Guaranteed smooth switch scheduling with low complexity," in *Proc. IEEE Globecom*, 2005, pp. 626-630.

[28] M. C. Paull, "Reswitching of connection networks," *Bell Syst. Tech. J.*, vol. 41, pp. 833-855, 1962.

[29] D. C. Opferman and N. T. Tsao-Wu, "On a class of rearrangeable switching networks—part I: control algorithm," *Bell Syst. Tech. J.*, vol. 5O, no. 5, pp. 1579-1600, May-June 1971.

[30] S. Andresen, "The looping algorithm extended to base 2 rearrangeable switching networks," *IEEE Trans. Commun.*, vol. vol. 25, no. 10, pp. 1057-1063, Oct. 1977.

[31] F. Hwang, "Control algorithms for rearrangeable Clos networks," *IEEE Trans. Commun.*, vol. 31, pp. 952-954, Aug. 1983.

[32] J. Gordon and S. Srikanthan, "Novel algorithm for Clos-type networks," *IEEE Electron. Lett.*, vol. 26, no. 21, pp. 1772-1774, Oct. 1990.

[33] Y. K. Chiu and W. C. Siu, "Comment: novel algorithm for Clos-type networks," *IEEE Electron. Lett.*, vol. 27, no. 6, pp. 524-526, Mar. 1991.

[34] H. R. Ramanujam, "Decomposition of permutation networks," *IEEE Trans. Comput.*, vol. C-22, pp. 639-643, July 1973.

[35] M. Kubale, "Comments on decomposition of permutation networks," *IEEE Trans. Comput.*, vol. C-31, pp. 265, Mar. 1982.

[36] A. Jajszczyk, "A simple algorithm for the control of rearrangeable switching networks," *IEEE Trans. Commun.*, vol. 33, pp. 169-171, Feb. 1985.

[37] C. Cardot, "Comments on a simple control algorithm for the control of rearrangeable switching networks," *IEEE Trans. Commun.*, vol. 34, p. 395, Apr. 1986.

[38] J. D. Carpinelli and A. Y. Oruc, "A nonbacktracking matrix decomposition algorithm for routing on Clos networks," *IEEE Trans. Commun.*, vol. 41, no. 8, pp. 1245-1251, Aug. 1993.

[39] Y. Ganjali and N. McKeown, "Update on buffer sizing in Internet routers," *ACM Sigcomm Comp. Commun. Rev.*, vol. 36, no. 5, pp. 67-70, Oct. 2006.

[40] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," in *Proc. Sigcomm*, pp. 281-292, ACM Press, 2004.

[41] G. Raina and D. Wishick, "Buffer sizes for large multiplexers: TCP queueing theory and instability analysis," in *Proc. EuroNGI*, Apr. 2005.

[42] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, "Routers with very small buffers," in *Proc. IEEE Infocom.*, Apr. 2006

[43] A. Dhamdhere and C. Dovrolis, "Open issues in router buffer sizing," *ACM/SIGCOMM Comp. Commun. Rev.*, vol. 36, no. 1, pp. 87-92, Jan. 2006.

[44] G. Vu-Brugier, R. S. Stanojevic, D. J. Leith, and R. N. Shorten, "A critique of recently proposed buffer sizing strategies," *ACM/SIGCOMM Comp. Commun. Rev.*, vol. 37, no. 1, pp. 43-47, May 2007.

[45] T. H. Szymanski, "QoS switch scheduling using recursive fair stochastic matrix decomposition," in *Proc. IEEE High Performance Switching and Routing (HPSR) Conf.*, 2006, pp. 417-424.

[46] T. H. Szymanski, "Method and apparatus to schedule packets through a crossbar switch with delay guarantees," US Patent Application, submitted.

[47] T. H. Szymanski and C. Feng, "Randomized routing of virtual connections in an essentially nonblocking log N-depth network," *IEEE Trans. Commun.*, vol. 43, no. 9, pp. 2521-2531, Sep. 1995.

[48] T. H. Szymanski and D. Gilbert, "Delivery of guaranteed rate internet traffic with very low delay jitter," in *Proc. IEEE Pacific Rim Conf. Control, Commun. Signal Processing*, Aug. 2007, pp. 450-455.

[49] T. H. Szymanski and D. Gilbert, "Low-jitter guaranteed-rate communications for cluster computing systems" (invited), *Pacific Rim Special Issue, Int. J. Computer Networks Distributed Syst.*, vol. 2. no. 1, pp. 140-160, 2008.

[50] T. H. Szymanski and D. Gilbert, "Internet multicasting of IPTV with essentially zero delay fitter," *IEEE Trans. Broadcast.*, vol. 55, no. 1, pp. 20-30, Mar. 2009.

[51] T. H. Szymanski, "Bounds on the end-to-end delay and jitter in input-buffered and internally buffered IP networks," in *Proc. IEEE Sarnoff Symposium*, Apr. 2009.

[52] T. H. Szymanski, "A conflict-free low-jitter guaranteed rate MAC protocol for base-station communications in a wireless mesh network." in *Proc. Int. Conf. Accessnets*, Oct 2008.

[53] T. H. Szymanski, "Throughput and QoS optimization in nonuniform wireless mesh networks," in *Proc. ACM Int. Workshop Q2SWINET*, Nov. 2008, pp. 9-19.

[54] T. H. Szymanski and D. Gilbert, "Provisioning mission-critical telerobotic control systems over Internet backbone networks with essentially-perfect QoS," submitted.

**Ted H. Szymanski** completed a BaSc. in Engineering Science and the MaSc. and Ph.D. degrees in Electrical Engineering at the University of Toronto. He has held faculty positions at Columbia University in New York, where he was affiliated with the Center for Telecommunications Research (CTR), and McGill University in Montreal, where he was affiliated with the Canadian Institute for Telecommunications Research (CITR). From 1993 to 2003, he was a principle architect in a national research program on Photonic Systems funded by the Canadian Networks of Centers of Excellence (NCE) program. The program brought together significant industrial and academic collaborators, including Nortel Networks, Newbridge Networks (now Alcatel), Lockheed-Martin/Sanders, Lucent Technologies, and McGill, McMaster, Toronto, and Heriot-Watt Universities. The program demonstrated a free-space "intelligent optical backplane" exploiting emerging optoelectronic technologies, with 1,024 micro-optic laser channels per square centimeter of bisection area, for which he holds two patents. He currently holds the Red Wilson / Bell Canada Chair in Data Communications at McMaster University. His current interests include switching, scheduling, and network QoS for emerging telemedicine and telerobotic control systems. He has consulted for several companies and has served as the Associate Chair (undergraduate) and the undergraduate student advisor in the ECE Department.