Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2010, Article ID 169140, 14 pages
doi:10.1155/2010/169140

Review Article

Design of an IPTV Multicast System for Internet Backbone Networks

T. H. Szymanski[1,2] and D. Gilbert[1]

[1] *Department of Electrical and Computer Engineering, McMaster University, Hamilton, ON, Canada L8S 4K1*
[2] *Bell Canada Chair in Data Communications, McMaster University, Hamilton, ON, Canada L8S 4K1*

Correspondence should be addressed to T. H. Szymanski, teds@mcmaster.ca

Received 2 November 2009; Revised 3 March 2010; Accepted 23 March 2010

Academic Editor: Daniel Negru

The design of an IPTV multicast system for the Internet backbone network is presented and explored through extensive simulations. In the proposed system, a resource reservation algorithm such as RSVP, IntServ, or DiffServ is used to reserve resources (i.e., bandwidth and buffer space) in each router in an IP multicast tree. Each router uses an Input-Queued, Output-Queued, or Crosspoint-Queued switch architecture with unity speedup. A recently proposed *Recursive Fair Stochastic Matrix Decomposition* algorithm used to compute near-perfect transmission schedules for each IP router. The IPTV traffic is shaped at the sources using *Application-Specific Token Bucker Traffic Shapers*, to limit the burstiness of incoming network traffic. The IPTV traffic is shaped at the destinations using *Application-Specific Playback Queues*, to remove residual network jitter and reconstruct the original bursty IPTV video streams at each destination. All IPTV traffic flows are regenerated at the destinations with essentially zero delay jitter and essentially-perfect QoS. The destination nodes deliver the IPTV streams to the ultimate end users using the same IPTV multicast system over a regional Metropolitan Area Network. It is shown that all IPTV traffic is delivered with essentially-perfect end-to-end QoS, with deterministic bounds on the maximum delay and jitter on each video frame. Detailed simulations of an IPTV distribution system, multicasting several hundred high-definition IPTV video streams over several essentially saturated IP backbone networks are presented.

1. Introduction

Multimedia traffic such as IPTV and video-on-demand represent a rapidly growing segment of the total Internet traffic. According to Cisco [1, 2], global Internet traffic is nearly doubling every 2 years, and global capacity will have to increase 75 times over the decade 2002–2012 to keep up with the demand. Furthermore, video-based traffic such as IPTV [3] will represent 90% of global network loads in 2012. The US Federal Communication Commission (FCC) has required that all TV broadcasts occur in digital format in 2009, and the growing fraction of multimedia traffic threatens to overwhelm the current Internet infrastructure. According to [4], "*The United States will not be the first country to complete the transition to digital television… Luxembourg, the Netherlands, Finland, Andorra, Sweden, and Switzerland have all completed their transitions, utilizing the Digital Video Broadcasting—Terrestrial (DVB-T) standard. Transitions are now under way in more than 35 other countries*." According to Cisco [5]: "*With the deployment of these new IPTV services, existing network infrastructures will be pushed to their limits*." The congestion problems are already visible: "*Video is clogging the internet. How we choose to unclog it will have far-reaching implications*" [6]. Recognizing the problems, the US National Science Foundation initiated a major project called "*Global Initiative for Network Investigations*" (GENI), which is open to a complete "*clean slate*" redesign of Internet if necessary, in an attempt to address the problems [7, 8]. In summary, new technologies which support the efficient multicasting and broadcasting of multimedia services such as IPTV are essential.

In this paper, the design of an IPTV multicast system for Internet backbone networks is presented and explored

through extensive simulations. An IPTV multicast system was first proposed in [9] based upon a theoretical foundation established in [10, 11]. Extensive simulations where presented for an 8-node multicast tree in [9], however the design or simulation of a multicast system for a real IP network topology was not presented.

In this paper, the design of a realistic IPTV multicast system for several real backbone IP networks described in [12] is presented. Exhaustive simulations confirm that the system can deliver several hundred IPTV packet streams over the Internet backbone networks with essentially zero delay jitter, essentially zero packet loss rate, and essentially-perfect end-to-end QoS for every provisioned multicast tree. In the proposed system, a resource-reservation algorithm such as RSVP, IntServ, or DiffServ is used to reserve resources such as buffer space and transmission capacity in each multicast router in each multicast tree. Each IP router then uses a recently proposed *Recursive Fair Stochastic Matrix Decomposition* scheduling algorithm [9, 10] to schedule the IPTV traffic streams through the router while meeting rigorous QoS guarantees, under the constraint of unity speedup.

Internet routers can use three basic switch architectures, the *Input-Queued (IQ)* switch, the *Output-Queued (OQ)* switch, or the *Combined Input and Crosspoint Queued (CIXQ)* switch. OQ switches can achieve optimal throughput but they require an internal speedup of O(N), which renders them impractical for large sizes. *Combined Input and Output Queued* (CIOQ) switches have also been proposed. These switches can also achieve 100% throughput, but they also require a speedup typically by a factor of 2 or 4, which is difficult to realize and which increases costs. CIXQ switches can also achieve 100% throughput with simpler scheduling algorithms, but they require many crosspoint queues in the switching matrix which increase costs. To minimize costs, many high capacity routers exploit some form of Input Queueing. Figure 1 illustrates a packet-switched IP router using an $N \times N$ *Input-Queued* (IQ) switch architecture. Each input port has N "*Virtual Output Queues*" (*VOQs*), and the switch has a total of N^2 *VOQs*. Figure 2 illustrates a packet-switched IP multicast tree, which consists of a tree of packet-switched IP routers as shown in Figure 1.

Many IP routers exploit a fixed-sized cell switching architecture. Variable-sized IP packets containing video data arrive at the input ports. Each IP packet is disassembled into small fixed-sized cells, which are stored in the appropriate *VOQ* at the input side of the switch. Typical cells can be between 64 and 256 bytes in length. At each input port j, each *VOQ(j,k)* stores cells destined for output port k. The fixed-sized cells are scheduled for transmission across the switch in a series of time slots. The variable-sized IP packets are reconstructed at the output side of the router and are then transmitted to the next router in the multicast tree. In each time slot, a scheduling algorithm is used to compute a set of up to N cells to transfer across an IQ switch, subject to two constraints: (1) each input port transmits at most 1 cell from its N *VOQ*s, and (2) each output port receives at most 1 cell from any *VOQ*. The set of cells to transmit per time slot can be represented as a permutation.

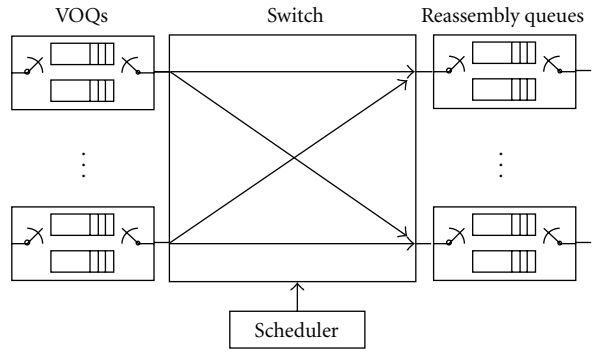

Figure 1: IQ switch.

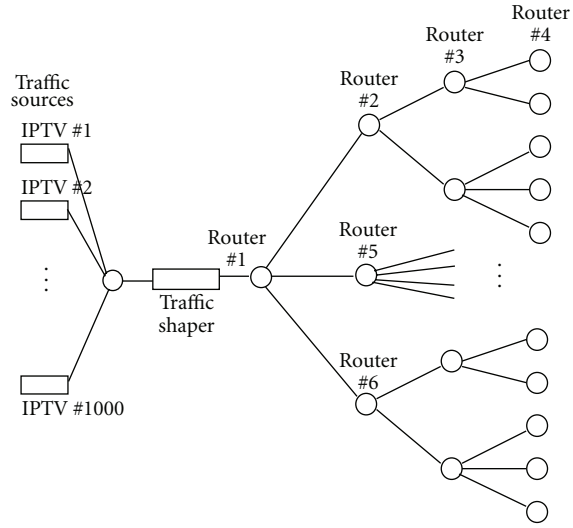

Figure 2: 2. IPTV Multicast tree.

Scheduling for IQ switches is known to be a difficult problem [13–27]. The difficulty is compounded when Quality-of-Service (QoS) constraints are added to the scheduling problem. It is has been shown that effective IPTV delivery requires low jitter [28–30]. The selection of a set of N cells to transfer per time-slot in an IQ switch is equivalent to finding a matching in a bipartite graph. Assuming link rates of 40 or 160 Gbps, the durations of a time-slot for a 64-byte cell are 12.8 and 3.2 nanoseconds, respectively. Therefore, schedulers for IQ switches must compute new bipartite graph matchings very quickly, typically at rates of 100–300 million matchings per second. Existing switch schedulers can be classified into two classes: (1) "*Dynamic schedulers*" which compute new bipartite matchings in every time-slot without any a priori knowledge of the long-term traffic demands on the switch, and (2) "*Guaranteed-Rate (GR) schedulers*" which periodically compute a sequence of F matchings to be used in F consecutive time slots called a "*scheduling frame*". The schedules can be reused repeatedly, and the schedule is recomputed when the long-term traffic demands of the switch are modified.

It is known that dynamic schedulers for IQ switches can achieve 100% throughput, if a *Maximum Weight Matching*

(*MWM*) algorithm is used to compute the matching for each time-slot, where the largest queues receive preferential service [13]. However, the *MWM* algorithm has complexity $O(N^3)$ work per time-slot and is considered intractable for use in real IP routers [13]. Therefore, existing dynamic schedulers typically use sub optimal heuristic schedulers, such as Parallel Iterative Matching or iSLIP [14]. However, due to the severe time constraints all heuristic schedulers have sub optimal throughput efficiencies and exhibit significant delay and jitter at high loads. The iSLIP algorithm used in the Cisco 1200 series routers is an iterative heuristic scheduler which can achieve throughput efficiencies as high as 80% for non uniform traffic patterns. However, the average queuing delay per cell can approach several thousand time slots at high loads, and the delay jitter can be equally high.

In this paper, a recently proposed *Recursive Fair Stochastic Matrix Decomposition* algorithm is used to schedule multicast IPTV traffic in each router in an IP multicast tree in a fully saturated IP network, and the performance is examined through extensive simulations. A resource-reservation protocol such as RSVP, IntServ, or DiffServ is used to maintain a traffic rate matrix for each IP router. (We note that while DiffServ does not use explicit resource reservation, DiffServ does use class-based weighted fair queueing, which effectively reserves bandwidth for each Diffserv traffic class.) Each traffic rate matrix is doubly sub stochastic or stochastic, and specifies the guaranteed traffic rates between every pair of *Input-Output* (IO) ports of the router. The traffic matrix in each router can then be mathematically decomposed to yield a sequence of bipartite graph matchings or permutations. Each matching configures the switch for one time-slot, and the sequence of matchings is guaranteed to deliver the IPTV stream through the IP router while providing rigorous QoS guarantees, under the constraint of unity speedup.

The *Recursive Fair Stochastic Matrix Decomposition (RFSMD)* algorithm proposed in [10] converts an admissible traffic rate matrix for a router into a quantized (integer) matrix with integer-valued elements, assuming a scheduling frame of length F time slots. The algorithm then recursively partitions the quantized matrix in a recursive and relatively fair manner, yielding a sequence of permutation matrices, also called permutations. The resulting sequence of permutations forms a "*frame transmission schedule*", for transmitting cells through the packet-switched IP router. The sequence of permutations in a frame transmission schedule can be repeatedly reused, as long as the traffic rate matrix remains unchanged. When the traffic rate matrix is updated by the RSVP, IntServ, or DiffServ algorithm, the frame transmission schedule can be recomputed. The RFSMD algorithm, along with appropriate traffic shaping at the sources, rigorously guarantees that every provisioned traffic flow achieves near-minimal delay and jitter and essentially-perfect end-to-end QoS, for all networks loads up to 100%. By "essentially-perfect QoS", we mean that every provisioned traffic flow can be delivered with zero packet loss rate, nearminimal end-to-end delay and zero network-introduced delay jitter. Theoretical bounds on the delay, jitter and QoS are presented in [10, 11] and are summarized in Section 4 of this paper. In this paper, we apply the RFSMD scheduling algorithm to the problem of multicasting real IPTV traffic through multicast trees in several real IP backbone networks, to explore the feasibility of large-scale IPTV multicasting in IP networks.

Section 2 describes video traffic model. Section 3 describes some prior guaranteed-rate scheduling algorithms. Section 4 describes the low-jitter RFSMD scheduling algorithm in more depth. Section 5 describes the IP backbone networks and the IPTV multicast trees, and presents detailed simulation statistics. Section 6 contains concluding remarks.

2. Video Traffic Model

Cisco Systems estimates that several hundred video channels requiring up to 1 Gbps bandwidth may be distributed over the IP backbone to support emerging IPTV applications [5]. To gather realistic data for our simulations, a high-definition video stream entitled "*BBC Blue Planet*" available at the University of Arizona [31] website was processed. The video stream includes 61 K video frames which arrive at the rate of 24 video frames/sec. The minimum, mean, and maximum video frame sizes are 81, 21 K, 495 K bytes respectively, illustrating a very bursty behaviour.

We assume these video frames are disassembled into fixed-sized 64-byte cells before transmission into the IP multicast tree. These video frame sizes correspond to a mean of 328 cells per video frame, with a minimum and maximum of 2 and 7,735 cells per video frame, respectively. The single video stream has a compression ratio of 151, with a mean bit rate of about 4 Mbps, and a peak bit rate of 95 Mbps. To simplify the terminology, define this data to represent a single "*video channel*". A "*video stream*" consists of the aggregation of 1 or more video channels.

The Arizona website [31] provides video frame size statistics for a few high-definition video channels. In this paper, we assume 100 high-definition video channels are to be multicast in each multicast tree, each with a 4 Mbps average rate, for an aggregate stream traffic rate of approximately. 404 Mbps. To achieve the statistics for the 100 video channels used in our network simulations, the data for the video "*BBC Blue Planet*" was reused in a circular manner, with a randomly selected starting video frame for each channel.

Table 1 lists some properties of the single video channel and several aggregated video traffic streams. In Table 1, the ratio of the peak-to-mean rates is an indication of the burstiness of the traffic. The single channel has a peak-to-mean ratio of 23.5, indicating a high degree of burstiness. Referring to Table 1, the aggregated stream of 100 channels has an aggregate data rate of 404 Mbps, with a peak rate of 700 Mbps. The ratio of peak-to-mean rates is 1.73, indicating a considerable reduction in burstiness.

Figure 3 illustrates visually the effect of aggregation of multiple video channels on the burstiness of the aggregated video stream. Figure 3 illustrates the instantaneous normalized bandwidth versus time for the same aggregated streams. The mean rate of each stream has been normalized to 1, and the reduction in burstiness when many video channels are aggregated is evident.

Table 1: Statistics on aggregated traffic.

Channels	Mean Rate(Mbps)	Max Rate	Max/Mean	Standard Deviation	10% extra cap delay (sec)
1	4.04	95	23.5	6.68	97.1
10	40.4	165	4.08	21	45.4
100	404	700	1.73	69	4.35

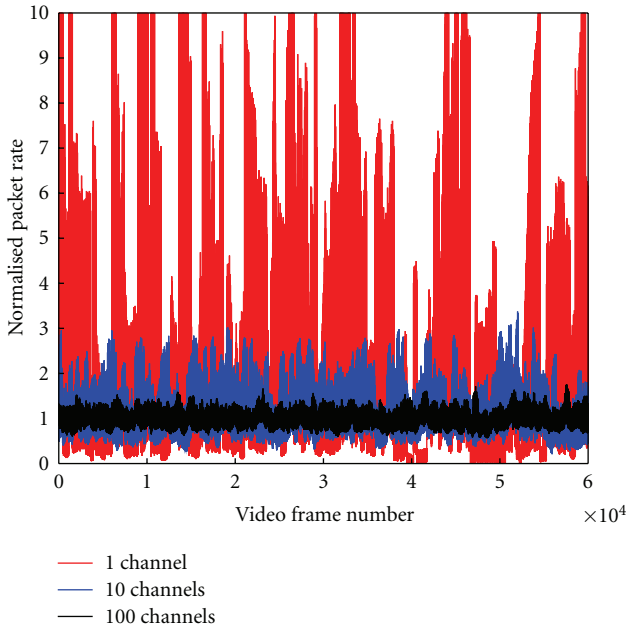

Figure 3: Burstiness of aggregated traffic.

Assume that video frames for any one video channel arrive at the root of a multicast tree at the fixed rate of 24 video frames per second. The arrival rate of video frames for the aggregated stream of 100 channels is therefore 2,400 video frames per second. The arriving traffic is quite bursty, as shown in Figure 3.

In this paper, we assume an *Application-Specific Token Bucket Traffic Shaper (ASTS)* module [9, 11] is used at the source to smoothen incoming bursty IPTV traffic to conform to an appropriate mean traffic rate with bounded burstiness. Cells are allowed to depart the ASTS at the maximum rate of 440 Mbps, when cells are available. The ASTS will introduce an *application-specific* delay at the source which is independent of the network. Referring to Figure 2, assume that the 100 video channels are available at the root of each IP multicast tree for distribution. The aggregated stream of 100 channels has an average data rate of 404 Mbps, and each IP multicast tree must be provisioned to support this traffic. In this paper, we assume that each IP multicast tree is provisioned such that the average data rate of the aggregated video stream consumes 90% of the provisioned tree link capacity, thereby providing 10% excess bandwidth for bursts. Therefore, each IP multicast tree must be provisioned to support about 440 Mbps of guaranteed rate traffic on every link in the tree. Given the line rate of 40 Gbps, the use of 64-byte cells, and a scheduling frame of length $F = 1$ K, then each time-slot reservation represents a bandwidth of approximately 40 Mbps. Therefore, a guaranteed rate of 440 Mbps requires the reservation of 11 cells per scheduling frame, or about 1% of the line rate. However, each backbone network has multiple IPTV multicast trees, between 9 and 16 trees in our simulations, so the fraction of multicast traffic on each link will be typically between 4% and 7% of the link capacity in our models. When bursts of cells arrive at the ASTS module, these cells will be temporarily stored, and will be released into the network at an average rate of 400 Mbps and a maximum data rate of 440 Mbps.

Referring to Figure 2, the router no.1 is the root of a multicast tree and implements a 1-to-3 multicasting of the cells. There are many nodes in an IP multicast tree, distributing content to potentially millions of end-users (i.e., households). Each destination node of the multicast tree has an *Application-Specific Playback Queue (ASPQ)* [9, 11] which receives the fixed-sized cells corresponding to the aggregated video stream. The purpose of the ASPQ is to filter out residual network-introduced jitter and reconstruct the original bursty video frames. The destination nodes in an Internet backbone network may represent a Central Office in a city. The Central Office node must deliver the IPTV streams to the ultimate end-users, which are the residential homes or mobile phones, and so forth, over a regional Metropolitan Area Network. The delivery of the IPTV traffic over the regional Metropolitan Area Network can use the same IPTV multicast design described in this paper. Therefore, bursty IPTV traffic flows can be delivered to the ultimate end-users in a hierarchical manner, with essentially zero delay jitter, with essentially zero packet loss rates and with essentially-perfect end-to-end QoS.

IP networks typically transmit variable-sized IP packets. Packets are typically disassembled into fixed-sized cells at the input size of each IP router, and IP packets are reassembled at the output size of the IP router, before they are transmitted to the next IP router. The use of variable-sized IP packets typically leads to delays associated with disassembling and reassembling IP packets in each IP router. In this paper, we assume an IP/MPLS technology, where all IP packets carrying video data have a fixed size, for example 64 bytes, 1024 bytes, or 1500 bytes. IP packets are disassembled once at the ingress router and reassembled once at the egress router of an MPLS domain. This assumption eliminates the need to repeatedly disassemble and re-assemble variable-sized IP packets at each IP router within one domain, and removes the packet reassembly delay in each IP router. However, the main results hold even if large fixed-sized packets are used

(i.e., 1500 bytes), or if variable size packets are used. In this case, the packet reassembly delay must be added to each router. The important point is that all variable queueing delays and jitter have been removed.

3. Prior Guaranteed-Rate Scheduling Algorithms

Several schemes have been proposed for scheduling guaranteed-rate (GR) traffic through an IQ packet switch, which are briefly reviewed. All of the schemes discussed in this section assume that every router maintains a traffic rate matrix, which specifies the requested traffic rates between all IO pairs in the router.

In the Birkoff von-Neuman (BVN) scheme proposed in [17, 18], a doubly stochastic traffic rate matrix is decomposed into a sequence of permutation matrices and associated weights. Each matrix represents a switch configuration, that is, a matching or permutation of input ports onto output ports. These matrices are then scheduled to appear in proportion to their weights using the GPS or WFQ algorithm, to determine the sequence of switch configurations which meet the GR traffic requirements. Each switch configuration is used to configure the packet switch for one time-slot. Best-Effort IP traffic can then use any switching capacity not used by GR traffic. The BVN decomposition can achieve 100% throughput through a router, that is, it does not introduce "*speedup*" at any router, However, the BVN decomposition has a time complexity of $O(N^{4.5})$ in a serial processor and is generally considered too slow for use in real-time packet-switched IP routers.

An algorithm to schedule traffic through an IQ packet-switched IP router, while attempting to minimize the "*service lag*" amongst multiple competing IP flows and to minimize the speedup, was developed at MIT [16]. A doubly stochastic traffic rate matrix is first quantized to contain integer values and then is decomposed into a series of permutation matrices and associated weights, which then must be scheduled. With speedup $S = 1 + sN$ between 1 and 2, the maximum service lag (defined ahead) over all IO pairs is bounded by $O((N/4)(S/(S-1)))$ time slots. According to [16]; "*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 IO pairs simultaneously*."

A greedy scheduling algorithm which attempts to minimize the delay jitter amongst simultaneous competing IP flows through an IQ packet switch was developed at Bell Labs [19, 20]. The traffic demand is specified in an $N \times N$ traffic rate matrix. The low-jitter GR traffic is constrained to be a relatively small fraction of the total traffic. The delay and jitter minimization problem is first formulated as an integer programming problem. The traffic rate matrix must be decomposed into a set of permutation matrices and associated weights, such that each traffic flow is supported by exactly one permutation in the set. This matrix decomposition problem was shown to be NP-HARD. They then formulate a greedy low-jitter decomposition with complexity $O(N^3)$ time. The resulting schedule requires a worst-case speedup of $O(\log N)$, which renders it costly in practice. Hard analytic bounds on the jitter were not available.

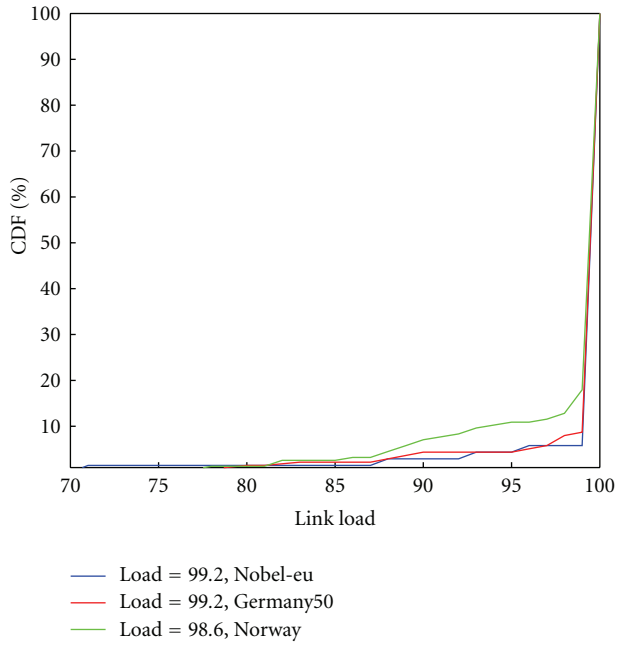

Figure 4: Average link loads.

A heuristic scheduling algorithm for scheduling packets in an IQ switch was developed at the UCR [21]. The algorithm attempts to decompose an admissible integer traffic rate matrix into the sum of integer matrices, where the maximum row or column sums of the matrices represent a decreasing geometric sequence. The authors establish a jitter bound which grows as the switch size N increases, and identify an open problem "*to determine the minimum speedup required to provide hard guarantees, and whether such guarantees are possible at all*" [21].

In summary, there is a considerable body of recent research into mathematical scheduling algorithms based upon the decomposition of traffic rate matrices. Unfortunately, tractable scheduling algorithms which achieve stability within the capacity region or which achieve bounded delays or jitter under the constraint of unity speedup, in one IP/MPLS router or a network of IP/MPLS routers, are unknown.

4. The Recursive Fair Stochastic Matrix Decomposition Algorithm

An $N \times M$ packet switch has N input and M output ports, and an associated traffic rate matrix. Each input port j for $0 \le j < N$ has M Virtual Output Queues, one for each output port k, $0 \le k < M$. The guaranteed-rate traffic requirements

for an $N \times N$ packet switch can specified in a doubly sub stochastic or stochastic traffic rate matrix Λ:

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

Each element $_{j,k}$ represents the fraction of the transmission line rate reserved for guaranteed-rate traffic between IO pair (j,k). The transmission of cells through the switch is governed by the frame 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 $1/F = 0.78\%$ of the line rate, which reserves one time-slot per frame on a recurring basis. Define a new quantized traffic rate matrix R where each traffic rate is expressed as an integer number times the minimum quota of reservable bandwidth:

$$R = \begin{pmatrix} R_{0,0} & R_{0,1} & \cdots & R_{0,N-1} \\ R_{1,0} & R_{1,1} & \cdots & R_{1,N-1} \\ \cdots & & & \cdots \\ R_{N-1,0} & R_{N-1,1} & \cdots & R_{N-1,N-1} \end{pmatrix}, \quad \sum_{i=0}^{N-1} R_{i,j} \leq F, \quad \sum_{j=o}^{N-1} R_{i,j} \leq F. \tag{2}$$

Consider two classic underlying theories from the field of graph theory and combinatorial mathematics, summarized in [10, 16–21]. Theorem 1 states that any integer matrix with a maximum row or column sum of F can be expressed as the sum of F permutation matrices. Theorem 2 states that any doubly sub stochastic or stochastic matrix can be decomposed into a convex set of permutations matrices and weights. All of the matrix decomposition algorithms reviewed in Section 3 attempt to exploit the above 2 theorems. They all are based upon the decomposition of a given traffic rate matrix into a convex set of constituent permutation matrices and associated weights, such that the weighted sum of the permutation matrices equals the original traffic rate matrix. Unfortunately, problems in combinatorial mathematics are difficult to solve efficiently, due to the large number of combinations that must be considered in any solution. As shown in Section 3, the problem of scheduling traffic to achieve zero jitter in minimum time is NP-HARD. All the algorithms reviewed in Section 3 (except for the BVN algorithm) require the introduction of "*speedup*" to the switches to achieve a decomposition, which limits their practical applicability. The BVN decomposition does not require a speedup, but its complexity is $O(N^{4.5})$ on a serial processor, which is considered intractable given that current Internet routers must compute decompositions at the rates of 100–300 million permutation matrices per second.

The *Recursive Fair Stochastic Matrix Decomposition* algorithm presented in [10] is an efficient deterministic algorithm to decompose an admissible traffic rate matrix into a sum of permutation matrices under the constraint of unity speedup. Let $P(M,F)$ denote the problem of scheduling an admissible quantized traffic rate matrix M into a scheduling frame of length F time slots, as shown in Figure 5. In particular, an integer matrix M with a maximum row or column sum of F must be partitioned into two integer matrices $M1$ and $M2$, each with a maximum row or column sum of $F/2$. More specifically, the scheduling problem $P(M,F)$ must be recursively decomposed into 2 smaller scheduling problems $P(M1,F/2)$ and $P(M2,F/2)$, such that the integer matrices $M1 + M2 = M$, where $M1$ and $M2$ are admissible integer traffic rate matrices, and for all j and k where $0 \leq j \leq N$ and $0 \leq k \leq N$, then $M1(j,k) \leq M2(j,k) + c$ and $M2(j,k) \leq M1(j,k) + c$ for $c = 1$. This step of partitioning an integer matrix into 2 integer matrices is a problem in combinatorial mathematics. The RFSMD algorithm achieves the decomposition by transforming one combinatorial problem into another combinatorial problem with a recently proposed solution. In particular, the problem of decomposing an integer matrix relatively fairly into 2 integer matrices is transformed into the problem of routing permutations in a rearrangeably nonblocking switching network [10]. An efficient algorithm for the combinatorial problem of routing permutations in rearrangeable networks was proposed in [10].

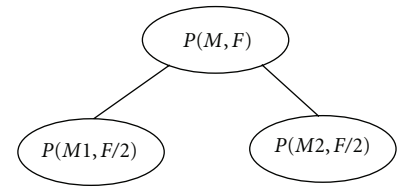

Figure 5: Recursive problem P(M,F) of scheduling of matrix M into a scheduling frame with F time slots.

The recursive nature of the RFSMD algorithm yields a very efficient decomposition. The decomposition of an $N \times N$ integer matrix with a maximum row or column sum of F can be accomplished in $O(NF \log (NF))$ time. The decomposition yields F permutations or bipartite graph matchings, which can configure the switch for F time slots. The computational complexity for each computed permutation of N elements is therefore $O(N \log (NF))$ time on a serial processor, which is near optimal.

The RFSMD algorithm also partitions each matrix M relatively fairly, so that the two smaller matrices $M1$ and $M2$ differ by at most 1 in any position. As a result, the traffic in the original scheduling problem is scheduled relatively fairly over the 2 smaller scheduling problems. This relatively fair recursive partitioning leads to a bound on the maximum jitter for any traffic flow, and a bound for the *maximum normalized service lead* or *maximum normalized service lag* for any scheduled traffic flow (these terms are formally defined in the next paragraphs). The bounded normalize service lead/lag can be used to create bounds on the end-to-end QoS for every scheduled traffic flow, as will be shown ahead.

One step in the decomposition for a 4×4 matrix operating at 99.2% load with unity speedup is shown in (3). Given an $N \times N$ switch and a fixed scheduling frame length F, the *RFSMD* matrix decomposition algorithm [10] bounds the *normalized service lead* and *service lag* for the aggregated traffic leaving any *node* to $\leq k \cdot IIDT$ time slots for constant K, where $IIDT$ represents the "*Ideal Interdeparture Time*" for cells belonging to the aggregated traffic leaving an edge. Furthermore, the bound applies to all individual competing traffic flows traversing each edge, provided that cells are selected for service within each VOQ according to a GPS scheduling algorithm

$$\begin{bmatrix} 106 & 222 & 326 & 345 \\ 177 & 216 & 303 & 326 \\ 459 & 232 & 183 & 147 \\ 282 & 352 & 211 & 178 \end{bmatrix} = \begin{bmatrix} 53 & 111 & 163 & 172 \\ 88 & 108 & 152 & 163 \\ 230 & 116 & 91 & 74 \\ 141 & 176 & 105 & 89 \end{bmatrix} + \begin{bmatrix} 53 & 111 & 163 & 173 \\ 89 & 108 & 151 & 163 \\ 229 & 116 & 92 & 73 \\ 141 & 176 & 106 & 89 \end{bmatrix}. \quad (3)$$

Each smaller scheduling problem contains approximately one half of the original time-slot reservation requests, and has as smaller scheduling frame of length $F/2$ time slots to realize these reservation requests, as shown in Figure 5. Repeated application of the relatively fair recursive partitioning results in a sequence of partial or full permutation matrices which determine the IQ switch configurations, called the frame transmission schedule. Due to the relatively fair recursive partitioning, the service a traffic flow receives in each half of the frame schedule will be relatively fair, and the delay jitter will be relatively small.

Several of the following definitions from [10] will be useful to interpret the simulation results to be presented in Section 6.

Definition 1. A "*Frame transmission 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 scheduling frame. Given a line rate L, the frame length F is determined by the desired minimum quota of reservable bandwidth $= L/F$. To set the minimum quota of reservable bandwidth to $\leq$ 1% of L, set $F \geq 100$, that is, $F = 128$.

Definition 2. The "*Ideal Interdeparture Time*" denoted $IIDT$ (i, j) of cells in a GR flow between IO pair (i, j) with quantized rate $R(i, j)$ time-slot reservations in a frame of length F, given a line rate L in bytes/sec and fixed-sized cells of C bytes, is given by: $IIDT(i, j) = F/R(i, j)$ time slots, each of duration (C/L) sec. (The subscripts will be suppressed when possible.)

Definition 3. The "*Ideal Service Time*" (ST) of cell $0 \leq c \leq R(i, j)$ in a GR flow between IO pair (i, j) with an *Ideal Interdeparture Time* of $IIDT(i, j)$ is given by $ST = j^* IIDT(i, j)$ time slots.

Definition 4. The "*Received Service*" of a flow with quantized guaranteed rate $R(i, j)$ at time-slot t within a frame of length F, denoted $Sij(t)$, equals the number of permutation matrices in time slots $1 \ldots t$, where $1 \leq t \leq F$, in which input port i is matched to output port j.

Definition 5. The "*Service Lag*" of a flow between input port i and output port j, at time-slot t within a frame of length F, denoted $Lij(t)$, equals the difference between the requested quantized GR prorated by t/F, and the received service at time-slot t, that is, $Lij(t) = Sij(t) - (t/F)Rij(t)$. A positive Service Lag denotes the case where the received service is less than the requested service, that is, a cell arrives later than its ideal service time. A negative Service Lag is a Service Lead, where the received service exceeds the requested service, that is, a cell arrive sooner than its ideal service time. The *normalized service lead/lag* for a flow f equals the service lead/lag for flow f divided by the IIDT for flow f.

Consider a discrete time queueing model, where time is normalized for all flows and is expressed in terms of the IIDT for each flow. The following notations presented in [11] are used. The cumulative arrival curve of a traffic flow f is said to conform to $T(\lambda, \beta, \delta)$, denoted, $A_f : T(\lambda, \beta, \delta)$ if the average cell arrival rate is λ cells/sec, the burst arrival rate is $\leq \beta\lambda$ cells/sec, and the maximum normalized service lead/lag is δ. A similar notation is used for cumulative departures and cumulative service. In any IP router, the cumulative departure curve for f is said to "*track*" the cumulative service curve for f when cell departures are constrained by the scheduled service opportunities. This situation occurs when flow f has queued cells at *VOQ(j,k)*.

The following four theorems were established in [11]. Assume each traffic flow is admitted to an IP/MPLS network subject to an *Application-Specific Token-Bucket Traffic Shaper Queue (ASSQ),* and has a maximum normalized service lead/lag of K cells. The traffic rate matrix for each router is updated by a resource reservation protocol such as RSVP, IntServ, or DiffServ. Each *IP router* is scheduled using the proposed *RFSMD* algorithm with a maximum normalized service lead/lag of K cells. We assume fixed size cells, with any reasonable cell size, however similar bounds apply for the case of variable-sized IP packet.

Theorem 1. *Given a flow f traversing VOQ(j,k) over an interval $t \in [0, \tau]$, with arrivals $A_f : T(\phi(f), \beta, K)$, with service $S_f : T(\phi(f)\beta, K)$ and $Q(0) \leq O(K)$, then $Q(t) \leq O(K)$.*

Theorem 2. *When all queues in all intermediate nodes have reached steady state, the maximum end-to-end queueing delay of a GR flow traversing H routers is $O(KH) \cdot IIDT$ time slots.*

Theorem 3. *In the steady state, the departures of traffic flow f at any IQ router along an end-to-end path of H routers are constrained by the scheduling opportunities, and will exhibit a maximum normalized service lead/lag of K, that is, $S_f : T(\phi(f)\beta, K)$. The normalized service lead/lag of a flow is not cumulative when traversing multiple routers.*

Theorem 4. *A traffic flow which traverses H IQ routers along an end-to-end path can be delivered to the end-user with zero network-introduced delay jitter, when a playback buffer of size 4 K cells is employed, that is,* $S_f : T(\phi(f)\beta, K)$.

According to Theorem 4, a bursty IPTV traffic flow which traverses H IQ routers along an end-to-end path can be delivered to the end-user with *zero network-introduced delay jitter*, when an appropriately-sized Application-Specific Token-Bucket Traffic Shaper Queue and Application-Specific Playback Queue are used, as described in Section 3.

Theorem 1 states that the number of cells buffered for any flow in any router in any network is limited to a small number for all loads up to 100%. Theorem 2 states that all end-to-end traffic flows will never experience any congestion, excessive delay or throughput degradation. Every end-to-end flow experiences a small but effectively negligible queueing delay at each router compared to current router technologies. Theorem 3 states that the normalized service lead/lag is not cumulative when traversing multiple routers in any network. Theorem 4 states that every end-to-end traffic flow can be delivered at every destination router with essentially-perfect end-to-end Quality of Service. In particular, a bursty IPTV traffic flow transmitted through an IP network using the RFSMD scheduling algorithm can be perfectly regenerated at every destination route, with zero packet loss rate and zero delay jitter. These 4 theorems demonstrate that traffic flows can be scheduled to achieve essentially-perfect end-to-end QoS in any Internet topology for all loads up to 100%. The simulation results reported in Section 6 will demonstrate these 4 theorems.

5. Backbone Topologies

The *Survivable Network Design Library* (*SNDlib*) is a library of telecommunication network designs [12]. The library contains data for 22 IP backbone networks and several optimization problems for each network. The SNDlib provides a series of real IP network topologies appropriate for reproducible case studies. In this paper, simulation results for three of the SDNlib networks are presented. We have followed the node placement and link assignments described by SNDlib, although we have made our own assumptions about the IPTV broadcast tree construction, session demands, and link capacities.

We selected the NOBEL-EU, GERMANY50, and the NORWAY network models from [12], as shown in Figure 6. The NOBEL-EU network covers Europe and has 28 nodes and 41 edges. It is a reference network originating from the European project NOBEL, where a detailed cost model was developed for various kinds of SDH and WDM equipment [12]. The GERMANY50 has 50 nodes and 88 edges provided by T-Systems International AG, and is an extension of the NOBEL-GERMANY network. The NORWAY network has 27 nodes and 51 edges and represents a backbone network from Norway.

We developed several heavily loaded IPTV *traffic specifications* for each network. In each traffic specification, one-third of the highest degree nodes are selected as roots of the IPTV multicast trees. For example, in the NOBEL-EU network with 28 nodes there are 9 multicast trees, each broadcasting 100 IPTV traffic streams. To generate each traffic specification, a 2-step process was used. In the first step, the IPTV multicast trees where routed into the network. N/3 nodes with the highest degrees were selected as roots for the N/3 trees. Each multicast tree was then routed to every other node in the network, using a shortest distance spanning tree computed using Dijkstra's algorithm. Many links in the network, especially those in the center of the topology, carry traffic from each of the all N/3 multicast trees. In the second step, background traffic was added in an iterative manner between randomly selected pairs of nodes, to essentially saturate the network. In our traffic specifications, we achieve link loads of approx. 99%. In each iteration, a random pair of nodes was selected, and a shortest delay path between the nodes was computed using an OSPF routing algorithm. A traffic rate was selected for the flow to nearly saturate the end-to-end path, and the flow would be confirmed. As the network load approaches 100%, it becomes increasingly difficult to route any additional flows, as all links approach saturation. Therefore, in the last iteration, multiple 1-hop traffic flows are added between neighboring nodes to effectively saturate the network, so that every link has a load of approximately 99%. Referring to Figure 4, the link loads used in our 3 network simulations are 99.2%, 99.2%, and 98.6%.

Each traffic specification represents an extremely heavy load, not likely to be encountered in practice. A resource reservation protocol such as RSVP, IntServ, or DiffServ was then used to reserve buffer space and bandwidth along each router and link in the network. These reservations result in the creation of an admissible traffic rate matrix for every router in the network. The traffic rate matrices for each router where then scheduled using the RFSMD algorithm. The network was then simulated, to gather statistics on the end-to-end QoS for every traffic flow, including IPTV multicast traffic and background traffic.

Numerous traffic specifications were generated, routed and simulated for the three selected network topologies. Figure 6 illustrates typical multicast trees in the 3 topologies examined in this paper, that is, the NOBEL-EU, NORWAY, and the GERMANY50 topologies. All of network simulations indicated essentially identical results, so we present the detailed simulations of one traffic specification over two networks, the NOBEL-EU and GERMANY50 networks. The NOBEL-EU network has 28 nodes, 41 links, and 9 IPTV multicast trees, each multicasting 100 IPTV channels to all other nodes. The selected traffic specification results in 321 traffic flows in the NOBEL-EU network, with an average distance of 5 hops and a maximum distance of 14 hops. The average link load is about 99%. The GERMANY50 network has 50 nodes, 88 links, and 16 IPTV multicast trees, each multicasting 100 IPTV channels to all other nodes. The selected traffic specification results in 693 traffic flows in the GERMANY50 topology, with an average distance of 5 hops and a maximum distance of 18 hops.

The roots of the IPTV multicast trees are illustrated as a bold circle in Figure 6. The IPTV multicast trees are

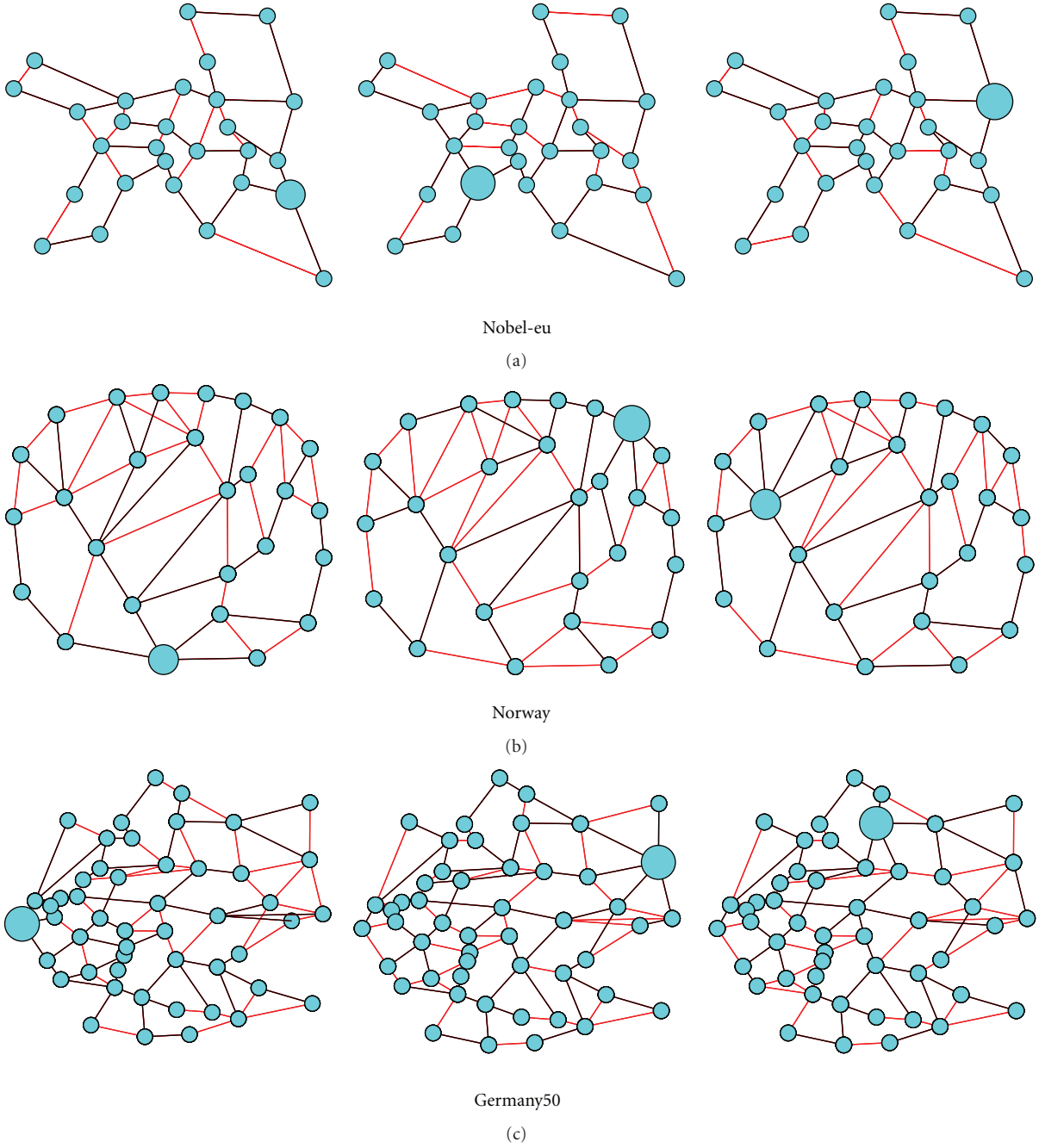

Figure 6: Three network topologies, each showing three typical multicast trees. Bold nodes are the roots of IPTV multicast trees.

routed according to Dijkstra's shortest path algorithm. As a result, links near the center of the network will tend to carry more IPTV traffic than those near the edge. Each multicast tree for a given source node is not unique, since in many cases each tree has several equal length shortest paths from which to select an edge. In our traffic specifications, a typical link will devote between 4% and 7% of its allocated bandwidth of 40 Gbps to IPTV multicast data, with a maximum of 20%. The remaining bandwidth is used for background traffic between randomly selected pairs of nodes.

6. Results of IPTV Multicasting

Assume an IP/MPLS backbone network with 40 Gbps links, with a scheduling frame length $F = 1.024$ (where 1 Gbps denotes 2^{30} bps and 1 Mbps denotes 2^{20} bps). The minimum quota of reservable bandwidth is one time-slot reservation per scheduling frame, or equivalently 4 Mbps. The provisioned GR rate of 440 Mbps per multicast tree established in Section 2 requires 11 time-slot reservations per scheduling frame. Each IP router in Figure 6 must reserve and schedule 11 cell transmissions per scheduling frame between the

appropriate IO pairs, per IPTV multicast tree. In our traffic specifications, every incoming and outgoing link in each IP router is essentially saturated with additional background traffic. Given this worst-case load, the IP routers should find it challenging to schedule the traffic to meet QoS guarantees.

The performance of the IP multicast trees in each IP backbone network was evaluated using a discrete-event simulator written in the C programming language, with over 20,000 lines of code. The simulator was written with the help of four graduate students, with funding provided by the Ontario Centers of Excellence. (Several independent simulators were actually written, to verify the experimental results.) The simulations were run on a large cluster-based supercomputing system in the Dept. of ECE at McMaster University, with 160 dual processing nodes running the LINUX operating system. Each dual-processor node has a clock rate of 1-2 GHz and between 1-2 GB of memory. A central dispatcher assigns tasks to processors to exploit parallelism.

Figure 7 illustrates the end-to-end delay (lifetime) PDF for (a) background traffic and (b) for multicast traffic in the GERMANY50 topology. Recall that there are 693 end-to-end paths and 16 IPTV multicast trees in this traffic specification. The multicast trees were routed first in the network using Dijkstra's shortest path algorithm, and therefore they have relatively low path lengths. The end-to-end delay for multicast traffic shown in Figure 7(b) varies from 1 IIDT up to about 9 IIDT. The background traffic is point-to-point, and was routed along shortest-delay paths using the OSPF routing algorithm. The shortest-delay paths generally do not equal the shortest-hop paths, and as a result the background traffic tends to be routed along longer paths. The end-to-end delays for background traffic shown in Figure 7(a) varies from about 1 IIDT up to 40 IIDT, indicating that some paths are very long.

Figure 8 illustrates the jitter of the traffic along each end-to-end path, for both background traffic and multicast traffic. The jitter along one path is defined as the deviation of each cell's delay along that path, relative to the mean value of the delay along a path. Figures 8(a) and 8(b) apply to the NOBEL-EU topology with 321 end-to-end paths. Figures 8(c) and 8(d) apply to the GERMANY50 topology with 693 end-to-end paths. Observe that 99% of all traffic flows arrive with less than 3 IIDTs of jitter, and 100% of all traffic flows arrive with less than 4 IIDTs of jitter. Observe from Figure 8 that the jitter for all point-to-point traffic and all multicast traffic is small and bounded by 4 IIDT, and the bound is network independent. These experimental results are consistent with the 4 theorems summarized in this paper. In [10], Theorems 5 and 8 establish that all jitter is bounded by 4 IIDT. Therefore, the experimental results in Figure 8 are perfectly consistent with the theorems in [10, 11].

Figure 9 illustrates the PDF for the distribution of the number of cells queued in each IP router in the GERMANY50 topology, for background traffic and multicast traffic. There are 3,465 individual plots in Figure 9, equal to 693 end-to-end paths passing through 5 routers on average. The number of cells queued per traffic flow in each router is small, less than 2 cells on average per multicast output

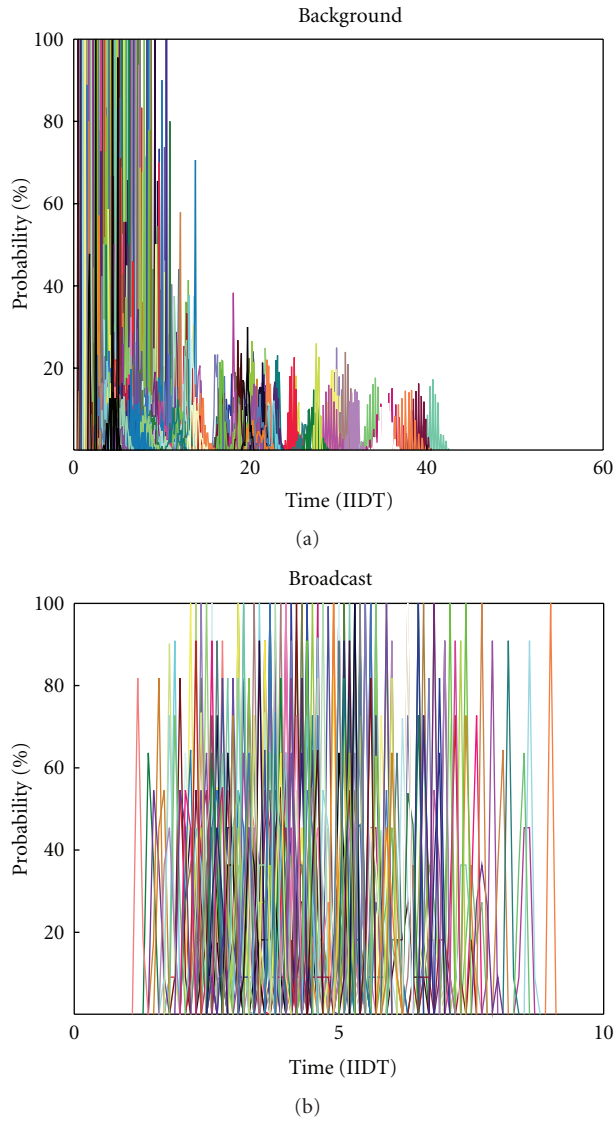

Figure 7: PDF of End-to-End Delay for Germany50 topology.

port. No router buffers more than 7 cells per flow per unicast output, as shown in Figure 9(a). No router buffers more than 3 cells per aggregated flow per multicast output port, as shown in Figure 9(b). The multicast cells are stored in a multicast queue, which is separate from the VOQs in each router. Figure 9 indicates that the cells move at a relatively consistent rate through the IP network. These experimental results are consistent with the 4 theorems summarized in this paper. In particular, Theorem 1 in [11] establishes that all queue sizes are bounded by a small number of cells (16 cells for these networks). Therefore, the experimental results in Figure 9 are perfectly consistent with the theorems in [10, 11]. The amount of memory required for buffering is several orders of magnitude lower when compared to traditional IP routers, as the next section will demonstrate.

Figure 10 illustrates the end-to-end delay for the IPTV multicast traffic, as a function of the excess bandwidth selected for each multicast tree. Recall that in our traffic

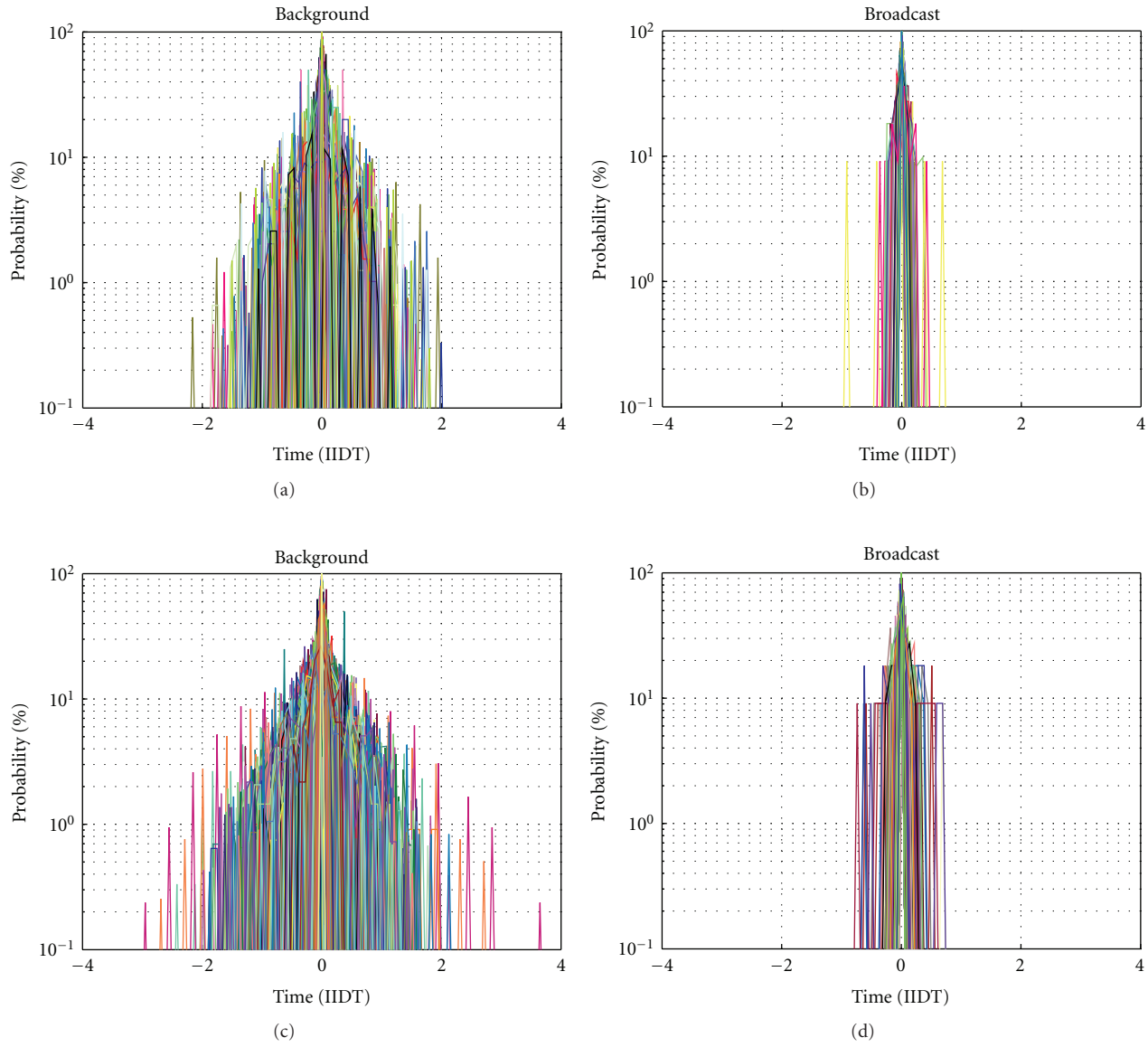

Figure 8: PDF of Delay Jitter. (a) and (b): Nobel-EU topology. (c) and (d): Germany50 topology.

specifications, an excess bandwidth of 10% was provisioned for every IPTV multicast tree. The excess bandwidth determines the end-to-end delay. According to Figure 10, an excess bandwidth of 10% yields an end-to-end queueing delay of about 3 seconds on each IPTV multicast tree. The *Application Specific Playback Queue* at each destination node has reconstructed the original bursty IPTV video frames, filtered out the residual network jitter and made all 100 video streams available at each destination IP router in the multicast tree with essentially zero delay jitter and essentially-perfect end-to-end QoS.

6.1. Buffer Sizing in IP Routers. In this section, we explore the issue of memory requirements in Internet routers. One well-established design rule for buffer sizing in IP networks using TCP flow-control is called the "*classical buffer rule*", also called the "*bandwidth-delay-product*" buffer rule [31]. This design rule states that *each link in each IP router* requires a buffer of $B = O(C \cdot T)$ bits, where C is the link capacity and T is the round-trip time of the flows traversing the link [31]. According to data in [31], a 40 Gbps link handling TCP flows with a round-trip time of 250 millisec requires a buffer size B about five million IP packets per link. Each IP packet may contain up to 1500 bytes (or equivalently 24 fixed-sized cells). Therefore, each link buffer may require up to 7.5 Gigabytes of memory per link. A 16×16 router may require about 120 Gigabytes of high-speed memory. This large buffer size is partially due to the use of the traditional TCP flow-control protocol, which introduces a large jitter into the traffic flows, which in turn necessitates the use of large buffers. The large buffer size is also partially due to the use of sub optimal heuristic schedulers, which introduce a large and potentially unbounded jitter at every router, which in turn necessitates the use of large buffers.

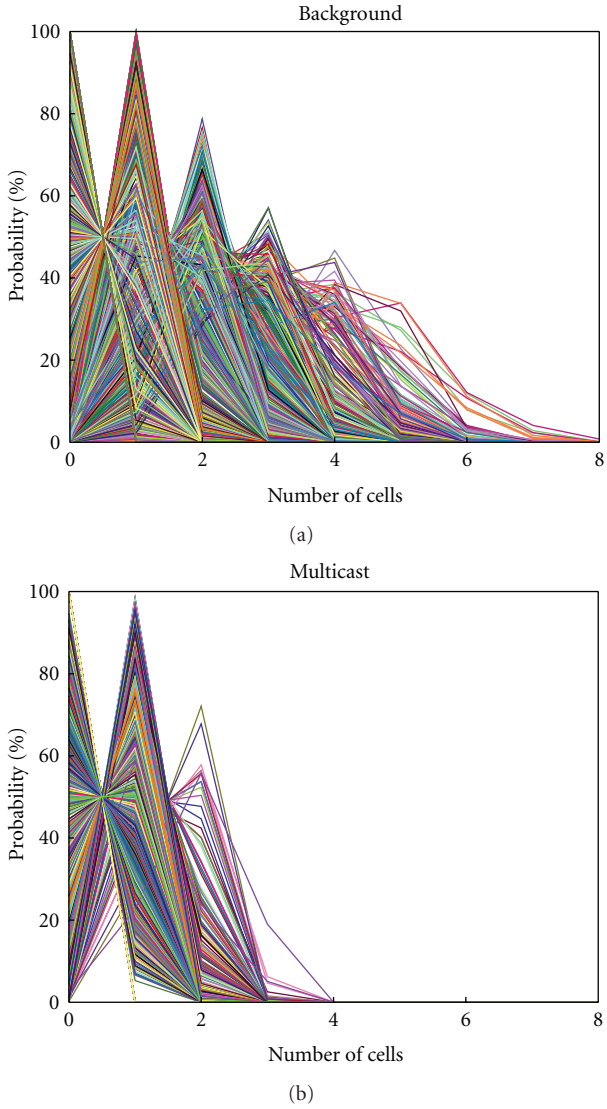

Figure 9: PDF for number of queued cells per traffic flow per router for Germany50 topology. (a) for background traffic. (b) for multicast traffic.

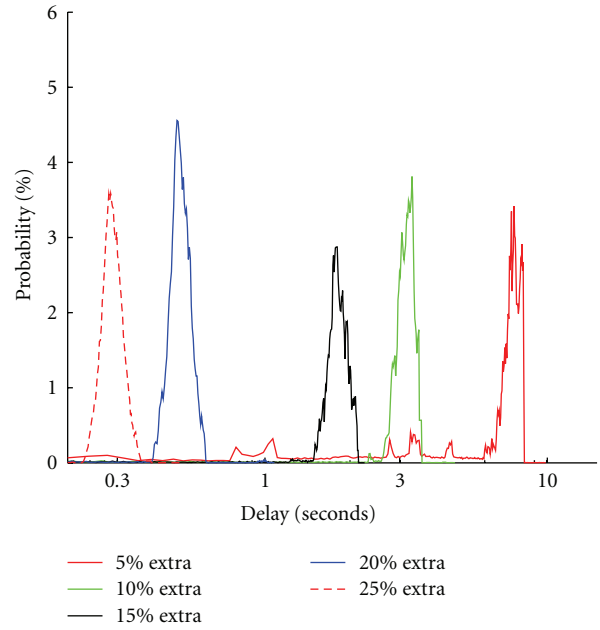

Figure 10: Plot of end-to-end delay versus excess bandwidth for multicast traffic.

A "*small buffer rule*" was proposed in [32], where $B = O(C \cdot T/\sqrt{N})$, and where N is the number of long-lived TCP flows traversing the router. With the same parameters reported above, the buffer size B can be reduced to about fifty thousand IP packets [32]. Therefore, each link buffer may require up to 75 Megabytes of memory per link. A 16×16 router may require about 1.2 Gigabytes of high-speed memory.

More recently, [33] proposed a "*tiny buffer rule*" where $B = O(\log W)$, where W is the maximum TCP congestion window size. With the same parameters, [33] suggests that average buffer sizes of between 20–50 IP packets or equivalently about 30,000–75,000 bytes of memory may suffice if 3 important conditions can be met; (a) the jitter of incoming traffic at the source node is sufficiently small, (b) the IP routers introduce a sufficiently small jitter, and (c) 10–15% of the throughput is sacrificed. The paper in [33] however did not propose a low-jitter scheduling algorithm, which is in itself a major theoretical unsolved problem as Section 3 illustrates. Furthermore, [32, 34] have argued that small buffers may cause significant losses, instability or performance degradation at the application layer.

Our theorems in [10, 11] and our exhaustive simulations indicate that network memory requirements can be several orders of magnitude smaller than those in current IP routers, when a low-jitter scheduling algorithm with a bounded normalized service lead/lag is used, as established in Theorems 1–4. In our IP multicast system, each IP router needs to buffer on average about 2 cells (128 bytes) per flow per router to guarantee 100% throughput and essentially-perfect end-to-end QoS for all provisioned traffic flows, even at 100% network loads. In comparison, existing IP routers using the combination of heuristic schedulers, TCP flow control, and the classic bandwidth-delay-product buffer-sizing rule require buffers of about 5 million IP packets per link at 40 Gbps to achieve high throughput, without providing any rigorous QoS guarantees. The RFSMD algorithm and Theorems 1–4 allow for reductions in buffer sizes by several orders of magnitude compared to the existing IP multicast technology, while simultaneously meeting rigorous QoS constraints.

7. Conclusions

The design of an IPTV multicast system for packet-switched Internet backbone networks has been presented. Three real IP backbone network topologies, the NOBEL-EU European topology, and the GERMANY50 and the NORWAY topologies, were used to illustrate the design. Multiple IPTV multicast tees were provisioned into each network, each

carrying 100 IPTV channels. Additional background traffic was added between nodes to essentially fully saturate each network. Every link operates at loads of approximately 99%. A resource reservation algorithm was used to reserve resources for each ITPV multicast tree in the network. A recently proposed *Recursive Fair Stochastic Matrix Decomposition a*lgorithm was used to compute near-perfect low-jitter transmission schedules for each packet-switched IP router in each multicast tree. The low-jitter schedules remove most of the cell delay jitter associated with the sub optimal dynamic scheduling algorithms used in existing IP routers and minimize the amount of buffering required in the IP routers. Extensive simulations indicate that large-scale IPTV multicasting can be supported. The IPTV traffic is delivered to every destination node with near-minimal delays, near-minimal jitter, zero packet loss rates, and essentially-perfect end-to-end QoS, as confirmed by theory. Our extensive simulations indicate that each IP router typically buffers less than 2 cells (about 128 bytes) of video data per traffic flow per router, several orders of magnitude less buffering than current IP routers require. The multicast technology is also applicable to other multimedia streams over the Internet, including VOIP, Video-on-Demand, Telemedicine, and Telerobotic control over IP. The application of the technology to deliver telerobotic control systems over saturated Internet backbone networks is described in [35].

References

[1] Cisco Systems White Paper, "Approaching the Zettabyte Era," 2008, http://www.cisco.com.

[2] R. Lawler, "Cisco Sees a Zettaflood of IP Traffic-Driven by Video," Contentinople—Networking the Digital Media Industry, 2008, http://www.contentinople.com.

[3] Y. Xiao, X. Du, J. Zhang, F. Hu, and S. Guizani, "Internet protocol television (IPTV): the killer application for the next-generation internet," *IEEE Communications Magazine*, vol. 45, no. 11, pp. 126–134, 2007.

[4] J. M. Boyce, "The US digital television broadcasting transition," *IEEE Signal Processing Magazine*, vol. 26, no. 3, pp. 110–112, 2009.

[5] Cisco Systems White Paper, "Optimizing Video Transport in your IP Triple Play Network," 2006, http://www.cisco.com.

[6] L. Hardesty, "Internet gridlock: video is clogging the internet. How we choose to unclog it will have far-reaching implications," *MIT Technology Review*, 2008.

[7] M. Baard, "NSF Preps New Improved Internet," Wired, August 2005.

[8] BBN Technologies—GENI Project Office, "Global Environment for Network Innovations—GENI System Overview," December 2007.

[9] T. H. Szymanski and D. Gilbert, "Internet multicasting of IPTV with essentially-zero delay jitter," *IEEE Transactions on Broadcasting*, vol. 55, no. 1, pp. 20–30, 2009.

[10] T. H. Szymanski, "A low-jitter guaranteed-rate scheduling algorithm for packet-switched IP routers," *IEEE Transactions on Communications*, vol. 57, no. 11, 2009.

[11] T. H. Szymanski, "Bounds on end-to-end delay and jitter in input-buffered and internally-buffered IP/MPLS networks," in *Proceedings of the IEEE Sarnoff Symposium (SARNOFF '09)*, Princeton, NJ, USA, 2009.

[12] S. Orlowski, R. Wessaly, M. Pioro, and A. Tomaszewski, "SNDlib1.0-survivable network design library," ZIB report ZR-07-15, Networks, October 2009, http://sndlib.zib.de/home.action.

[13] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 27, pp. 1936–1948, 1992.

[14] V. Anantharam, N. McKeown, A. Mekkittikul, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *IEEE Transactions on Communications*, vol. 47, no. 8, pp. 1260–1267, 1999.

[15] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, pp. 188–201, 1999.

[16] C. E. Koksal, R. G. Gallager, and C. E. Rohrs, "Rate quantization and service quality over single crossbar switches," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '04)*, pp. 1962–1973, 2004.

[17] C.-S. Chang, W. J. Chen, and H.-Y. Huang, "On service guarantees for input buffered crossbar switches: a capacity decomposition approach by birkhoff and von neuman," in *Proceedings of the IEEE International Workshop on Quality of Service (iWQoS '99)*, pp. 79–86, 1999.

[18] C.-S. Chang, W.-J. Chen, and H.-Y. Huang, "Birkhoff-von neumann input-buffered crossbar switches for guaranteed-rate services," *IEEE Transactions on Communications*, vol. 49, no. 7, pp. 1145–1147, 2001.

[19] I. Keslassy, M. Kodialam, T. V. Lakshman, and D. Stiliadis, "On guaranteed smooth scheduling for input-queued switches," *IEEE/ACM Transactions on Networking*, vol. 13, no. 6, pp. 1364–1375, 2005.

[20] M. S. Kodialam, T. V. Lakshman, and D. Stilladis, "Scheduling of Guaranteed-bandwidth low-jitter traffic in input-buffered switches," *US patent application no. #20030227901*, 2003.

[21] S. R. Mohanty and L. N. Bhuyan, "Guaranteed smooth switch scheduling with low complexity," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '05)*, pp. 626–630, 2005.

[22] N. McKeown and B. Prabhakar, "Scheduling multicast cells in an input-queued switch," in *Proceedings of the 15th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '96)*, vol. 1, pp. 271–278, March 1996.

[23] M. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri, "Multicast traffic in input-queued switches: optimal scheduling and maximum throughput," *IEEE/ACM Transactions on Networking*, vol. 11, no. 3, pp. 465–477, 2003.

[24] J. F. Hayes, R. Breault, and M. K. Mehmet-Ali, "Performance analysis of a multicast switch," *IEEE Transactions on Communications*, vol. 39, no. 4, pp. 581–587, 1991.

[25] J. Y. Hui and T. Renner, "Queueing analysis for multicast packet switching," *IEEE Transactions on Communications*, vol. 42, no. 2, pp. 723–731, 1994.

[26] B. Prabhakar, N. McKeown, and R. Ahuja, "Multicast scheduling for input-queued switches," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 5, pp. 855–866, 1997.

[27] M. Song, W. Zhu, A. Francini, and M. Alam, "Performance analysis of large multicast switches with multicast virtual output queues," *Computer Communications*, vol. 28, no. 2, pp. 189–198, 2005.

[28] S. R. Gulliver and G. Ghinea, "The perceptual and attentive impact of delay and jitter in multimedia delivery," *IEEE Transactions on Broadcasting*, vol. 53, no. 2, pp. 449–458, 2007.

[29] S. Chand and H. Om, "Modeling of buffer storage in video transmission," *IEEE Transactions on Broadcasting*, vol. 53, no. 4, pp. 774–779, 2007.

[30] Y. Bai and M. R. Ito, "Application-aware buffer management: new metrics and techniques," *IEEE Transactions on Broadcasting*, vol. 51, no. 1, pp. 114–121, 2005.

[31] Y. Ganjali and N. McKeown, "Update on buffer sizing in internet routers," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 5, pp. 67–70, 2006.

[32] G. Vu-Brugier, R. S. Stanojevic, D. J. Leith, and R. N. Shorten, "A Critique of recently proposed buffer sizing strategies," *ACM/SIGCOMM Computer Communication Review*, vol. 37, no. 1, pp. 43–47, 2007.

[33] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, "Routers with very small buffers," in *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM '06)*, Barcelona, Spain, April 2006.

[34] A. Dhamdhere and C. Dovrolis, "Open issues in router buffer sizing," vol. 36, no. 1, pp. 87–92.

[35] T. H. Szymanski and D. Gilbert, "Provisioning mission-critical telerobotic control systems in internet backbone networks with essentially-perfect QoS," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 5, pp. 630–643, 2010.