Internet Multicasting of IPTV With Essentially-Zero Delay Jitter

Ted H. Szymanski, *Member, IEEE*, and Dave Gilbert, *Member, IEEE*

***Abstract*—A technology for multicasting packetized multimedia streams such as IPTV over the Internet backbone is proposed and explored through extensive simulations. An RSVP or DiffServ algorithm is used to reserve resources (i.e., bandwidth and buffer space) in each packet-switched IP router in an IP multicast tree. Each IP router uses an Input-Queued (IQ) switch architecture with unity speedup. A recently proposed low-jitter scheduling algorithm is used to pre-compute a deterministic transmission schedule for each IP router. The IPTV traffic will be delivered through the multicast tree in a deterministic manner, with bounds on the maximum delay and jitter of each packet (or cell). A playback buffer is used at each destination to filter out residual network jitter and deliver a very low-jitter video stream to each end-user. Detailed simulations of an IPTV distribution network, multicasting 75 high-definition video streams over a fully-saturated IP backbone are presented. The simulations represent the transmission of 129 billion cells of real video data and where performed on a 160-node cluster computing system. In the steady-state, each IP router buffers approx. 2 cells (128 bytes) of video data per multicast output-port. The observed delay jitter is zero when a playback buffer of 15 milliseconds is used. All simulation parameters are presented.**

***Index Terms*—Broadcast, input-queue, IPTV, low-jitter, multicast, packet-switching, scheduling, time-division switching.**

I. Introduction

THE delivery of new services such as IPTV, Video-on-Demand (VOD) and telerobotic control will push network infrastructures to the limit and require very high levels of QoS [1]–[3]. There are two dominant technologies for the delivery of packetized IPTV to multiple recipients over the Internet; peer-to-peer technology and IP multicast tree technology. In both systems, the delay jitter plays an important role in determining the size of queues required within the packet-switched IP routers, and the size of the playback queues at the destinations [4]. According to the IETF RFC 3393 [4], the concept of '*jitter*' is precisely summarized by the term '*IP Packet Delay Variation*' (IPDV). According to [4], the maximum IPDV is used to '*size the play-out buffers*' for applications such as VOIP and IPTV, and it also determines the '*dynamics of queues within a network or router*'. Therefore, minimization of delay jitter will generally reduce the number of packets or cells queued within the IP routers and the playback queues, and enable the delivery of new services which require very low jitter.

ITPV is being touted as a potential driving technology for the next generation Internet [5]. However, there are many technical challenges to be resolved. Network-related QoS challenges include minimization of network congestion, delay, jitter, out of sequence packets and packet loss rates [5]. Video-related QoS challenges include minimization of channel change times and maximization of a user's perceived QoS [5]. Techniques to minimize channel change times are described in [7]–[10]. The severe negative impact of end-to-end delay and delay jitter on a user's perceived QoS, and the importance of minimizing delay and jitter, are described in [11], [12]. Even low amounts of jitter or packet loss can result in a severe degradation of quality as perceived by users [11]. Analyses regarding the playback buffer sizes required to reduce delay jitter are presented in [12]–[14]. An intelligent packet dropping strategy which drops less important video packets when congestion is encountered is described in [14]. Techniques to optimize the performance of '*Video-on-Demand*' systems are described in [15], [16].

The US Federal Communication Commission (FCC) has mandated that all TV broadcasts occur in digital format in 2009. According to Cisco [2]: "*With the deployment of these new IPTV services, existing network infrastructures will be pushed to their limits*". According to [18], problems are already visible: '*Video is clogging the internet. How we choose to unclog it will have far-reaching implications*'. In early 2008, the FCC held meetings at Harvard and Stanford universities to discuss options for relieving the congestion caused by video distribution over the Internet [18]. Recognizing the problems, the US National Science Foundation has embarked on a major initiative called '*Global Initiative for Network Investigations*' (GENI), which is open to a complete '*clean slate*' redesign of Internet in an attempt to solve problems. In summary, new technologies which support the efficient delivery of IPTV and other multimedia services are essential.

The use of '*peer-to-peer*' (P2P) technologies to deliver digital video is an alternative to the use of IP multicast trees. P2P systems rely upon the distributed storage and retrieval of content over the Internet; the content may be stored over potentially hundreds of peers, and a user requesting access typically retrieves the content in parallel from these peers. P2P technology offers high availability, high quality and ease of use, but it can suffer from excessive delays and jitter. The data is delivered to an end-user using best-effort IP service from multiple peers, and typically experiences moderate to large delays, necessitating the use of playback buffers with delays of 1-2 minutes at the end-users.

Manuscript received December 31, 2007; revised September 08, 2008. Current version published February 25, 2009.

The authors are with the Department of Electrical and Computer Engineering, McMaster University, Hamilton, ON LHS4K1 Canada (e-mail: teds@mcmaster.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TBC.2008.2007455

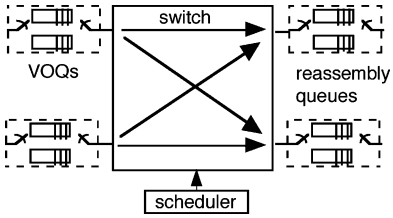

Fig. 1. IQ switch.

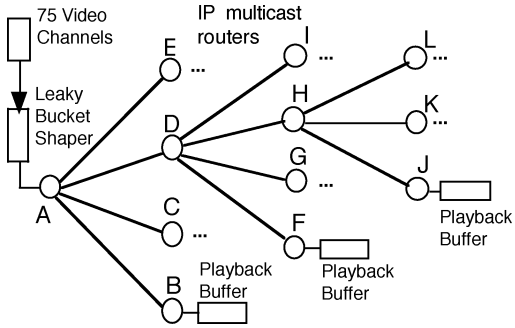

Fig. 2. IP multicast tree.

The use of packet-switched IP multicast trees to deliver content is a well-established technology. According to Cisco [2], a source node containing content is typically the root of an IP multicast tree, which consists of a tree of "access routers". Each access router receives a stream of packets from the root node, and forwards the packets to (a) other access nodes further down the tree, and (b) to end-users. According to Cisco, a typical root node may multicast up to 200 digital TV channels and a typical access-node may supply up to 10,000 users with content [2]. IP multicasting technology also offers relatively high availability, high quality and ease of use, but it can suffer from delays, necessitating the use of playback buffers with delays of $\approx$150 millisec at the end-users [2]. According to Cisco, the introduction of IPTV and VOD services is expected to increase Internet network loads significantly, creating a greater need for delivery schemes which meet QoS guarantees [2].

In this paper, a technology for the real-time delivery of IPTV packet streams over a packet-switched IP multicast tree, with essentially-zero delay jitter and essentially-zero packet loss rate is proposed. In traditional IP multicasting, an RSVP or DiffServ algorithm is used to reserve resources such as buffer space and transmission capacity in each packet-switched IP router in the multicast tree. Each IP router then uses a dynamic switch-scheduling algorithm to schedule the delivery of packets through the switch according to QoS constraints.

Internet routers can exploit either of three basic switch architectures, the *Input-Queued (IQ)* switch, the *Output-Queued (OQ)* switch, or the *Internally Buffered Crosspoint (IBC)* switch. OQ switches can achieve optimal throughput but they require an internal speedup, typically by a factor of 2 or 4, which is difficult to realize and which increases costs. ICB switches can also achieve optimal throughput, but they require many internal buffers in the switching matrix which increase costs. To minimize costs, many high capacity routers exploit an IQ switch architecture. Fig. 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*. Fig. 2 illustrates a packet-switched IP multicast tree, which consists of a tree of packet-switched IP routers as shown in Fig. 1.

Consider an IP router which uses an IQ crossbar switch. IP packets containing video data arrive at the input ports. Each IP packet is disassembled into small 64-byte cells, which are stored in the appropriate VOQ at the input side of the switch. Each $VOQ(j,k)$ stores cells at input port j destined for output port k. The cells are transferred across the IQ switch in a series of time-slots. The IP packets are reconstructed at the output side of the router and 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 VOQs, and (2) each output ports receives at most 1 cell from any VOQ.

Scheduling for IQ switches is known to be a difficult problem [19], [20]. The selection of a set of N cells to transfer per time-slot is equivalent to finding a matching in a bipartite graph. Assuming a 40 Gbps link rate, the duration of a time-slot is 12.8 nanosec. Therefore, a scheduler for an IQ switch must compute a new bipartite graph matching every 12.8 nanosec. Existing 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 schedulers*' which periodically compute a sequence of F matchings to be used in F consecutive time-slots called a '*frame transmission schedule*'. The schedule can be re-used repeatedly, and the schedule is re-computed when the long-term traffic demands of the switch are modified.

It has been shown that dynamic schedulers for IQ switches can achieve optimal (100%) throughput, if a *Maximum Weight Matching* (*MWM*) algorithm is used to compute the matching for each time-slot, wherein the largest queues receive preferential service [19], [20]. However, the MWM algorithm has complexity $O(N^3)$ time and is considered far too computationally expensive for use in real IP routers [19]–[21]. Therefore, existing dynamic schedulers typically use sub-optimal heuristic schedulers. However, due to the severe time constraints all heuristic schedulers have sub-optimal throughput efficiencies and significant delay and jitter at high loads [21]–[24]. The iSLIP algorithm [22] used in the Cisco 1200 series routers is an iterative heuristic scheduler which can achieve throughput efficiencies as high as 80% for nonuniform 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.

The use of TCP flow control will also introduce a significant *application-specific* delay and jitter at the source of an IPTV traffic flow. Given these large delays and jitter, 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 [26]. 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 [26] or equivalently several tens of millions of fixed-sized cells.

A '*small buffer rule*' was proposed in [27], where $B = O(C \cdot T/\sqrt{N})$, 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 [26] or several hundred thousand cells. More recently, [28], [29] proposed a '*tiny buffer rule*' where $B = O(logW)$, where W is the maximum TCP congestion window size. Using the same parameters, a buffer size of between 20–50 packets or several hundred 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, [28], [30], [31] 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 IP routers with low jitter and unity speedup is in itself a significant unsolved problem [19]–[41]. In summary, there are significant problems when using existing schedulers and the inherently high jitter TCP flow control protocol in IP networks.

Guaranteed-Rate (GR) schedulers provide an alternative to dynamic schedulers. Many GR scheduling algorithms have been proposed in the literature [32]–[47], and a brief summary is provided in Section III. To date all previous proposed GR schedulers [32]–[41] cannot achieve 100% throughput efficiency while maintaining unity '*switch speedup*', and they cannot provide sufficiently small bounds on (a) the number of cells queued in each IQ switch, (b) the queuing delays within an IQ switch, or (c) the delay jitter within an IQ switch given unity speedup. To date, no commercial IP router has used a GR scheduler.

In this paper, a recently proposed GR scheduler which can provably achieve very low jitter and 100% throughput efficiency while maintaining unity speedup is explored. This GR algorithm can also provably achieve near-minimal queue sizes and delay in each IP-router [42]–[44]. In this paper, this GR algorithm is used to multicast IPTV traffic over an IP multicast tree in a fully saturated IP network, and the performance is examined through extensive simulations.

In GR schemes, an RSVP, IntServ or DiffServ algorithm is used to maintain (or incrementally update) a traffic rate matrix for each IP router. The matrix is doubly substochastic or stochastic, and specifies the guaranteed traffic rates between the IO ports of each packet-switched IP router. The matrix can then be processed to yield a sequence of switch permutations (or configurations) which can deliver the GR packets through the IP router according to delay and jitter constraints.

The scheduling algorithm in [42]–[44] converts a doubly-stochastic traffic rate matrix into a quantized 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 permutations (or switch configurations). The resulting sequence forms a '*frame transmission schedule*', for transmitting cells through the packet-switched IP router. The sequence of permutations in a frame is repeatedly re-used, as long as the traffic rate matrix remains unchanged. When the traffic rate matrix is updated by the RSVP of DiffServ algorithm, the frame transmission schedule is recomputed.

Due to the recursive and fair nature of the scheduling algorithm in [42]–[44], the frame transmission schedules have very low-jitter. Theoretical bounds on the delay and jitter for every flow are presented in [43], [45]. The scheduling of multimedia streams through a chain of 10 IP routers was explored in [46]. Simulations indicate that essentially-zero delay jitter can be achieved in a 100% saturated IP network. In the steady-state, each IP router buffers roughly 2 cells per traffic flow, several orders of magnitude less buffering than current IP routers require. The scheduling of supercomputer traffic through a Fat-Tree network was explored in [47]. Simulations indicate that essentially-zero delay jitter can also be achieved in a 100% saturated network, and that in the steady-state each IP router also buffers roughly 2 cells per traffic flow, also several orders of magnitude less buffering than current IP routers require. In this paper, we apply the scheduling algorithm in [42]–[44] to the problem of multicasting real IPTV traffic through a multicast tree in a fully saturated IP network, to explore the issue of buffer sizes in a multicast tree.

Referring to the IP multicast tree in Fig. 2, assume that a stream of IPTV channels must be distributed to every leaf node. Therefore, a guaranteed rate traffic matrix must be specified for each IP router in Fig. 2. The matrix will specify the amount of guaranteed bandwidth being provisioned between the IO ports of each IP router.

Assume a scheduling frame size of $F = 1024$ time-slots. A frame transmission schedule to transmit cells of data between the IO ports of each IP router is computed from the GR traffic rate matrix associated with each IP router. Consider the following traffic rate matrix shown in (1), which is used to configure crossbar switch A in the multicast tree in Fig. 2. Input port 3 is configured to transmit to output ports (2,4,5,7), at the rate of 48 cells per frame. Assuming a line-rate of 10 Gbps, a frame size of $F = 1024$, and 64-byte cells, then each cell reservation provides a guaranteed rate of 10 Mbps between an IO pair. Using the traffic rate matrix in (1), input port 3 reserves 48 cell reservations per frame for each output port 2, 4, 5 and 7, as can be seen by observing row 3 of the matrix. Therefore, input port 3 will transmit cells to output ports (2,4,5,7) at the rate of $48 * 10\ \text{Mbps} = 480\ \text{Mbps}$:

$$M = \begin{bmatrix} 145 & 150 & 154 & 174 & 95 & 111 & 109 & 86 \\ 130 & 129 & 173 & 88 & 174 & 117 & 96 & 117 \\ 128 & 48 & 85 & 48 & 48 & 148 & 48 & 120 \\ 116 & 122 & 115 & 127 & 157 & 153 & 120 & 114 \\ 105 & 130 & 133 & 118 & 114 & 155 & 160 & 109 \\ 124 & 130 & 102 & 120 & 130 & 96 & 150 & 172 \\ 141 & 125 & 113 & 135 & 115 & 116 & 98 & 181 \\ 135 & 117 & 149 & 111 & 96 & 128 & 163 & 125 \end{bmatrix} \quad (1)$$

Let $P(M, F)$ denote the problem of scheduling the traffic matrix M given a frame of length F time-slots. The low-jitter scheduling algorithm in [42]–[44] recursively partitions the

matrix M in (1) to yield 2 smaller scheduling problems $P(M1, F/2)$ and $P(M2, F/2)$, where representative matrices M1 and M2 are shown in (2). 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. Repeated application of the relatively fair recursive partitioning described in [42]–[44] will result 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.

$$M1 = \begin{bmatrix} 72 & 75 & 77 & 87 & 43 & 55 & 54 & 43 \\ 65 & 65 & 86 & 44 & 87 & 58 & 48 & 57 \\ 64 & 24 & 42 & 24 & 24 & 74 & 24 & 60 \\ 58 & 61 & 57 & 64 & 79 & 77 & 60 & 57 \\ 52 & 65 & 66 & 59 & 57 & 78 & 80 & 55 \\ 62 & 65 & 51 & 60 & 65 & 48 & 75 & 86 \\ 71 & 63 & 56 & 67 & 57 & 58 & 49 & 90 \\ 68 & 58 & 74 & 56 & 48 & 64 & 82 & 63 \end{bmatrix}$$

$$M2 = \begin{bmatrix} 73 & 75 & 77 & 87 & 42 & 56 & 55 & 43 \\ 65 & 64 & 87 & 44 & 87 & 57 & 48 & 58 \\ 64 & 24 & 43 & 24 & 24 & 74 & 24 & 60 \\ 58 & 61 & 58 & 63 & 78 & 76 & 60 & 57 \\ 53 & 65 & 67 & 59 & 57 & 77 & 80 & 54 \\ 62 & 65 & 51 & 60 & 65 & 48 & 75 & 86 \\ 70 & 63 & 57 & 68 & 58 & 58 & 49 & 91 \\ 67 & 59 & 75 & 55 & 48 & 64 & 81 & 62 \end{bmatrix} \quad (2)$$

By applying the resulting pre-computed deterministic frame transmission schedules to each IP router in Fig. 2, the GR traffic will move through the packet-switched IP multicast tree of Fig. 2 in a deterministic pattern. Therefore, the delivery of a packetized multimedia stream across the IP backbone with very low delay jitter may be possible if 2 conditions can be met: (a) each frame transmission schedule is relatively fair such that no cells wait excessively long for service, and (b) each packet-switched IP router buffers a sufficient number of cells per flow to compensate for any service lead/lag it may experience, which will keep the transmission pipeline active. Furthermore, if (c) the playback buffer at the destination end-user has sufficient buffer space to filter out any residual network jitter, then essentially-zero delay jitter may be achievable. Under these conditions, packets or cells in a multimedia flow will be transmitted through each IP router in a multicast tree in an IP or MPLS backbone in a deterministic pattern, where the end-to-end delay and jitter are deterministically bounded. The real challenge is to find a good guaranteed-rate scheduling algorithm with low computational complexity, with low speedup requirements and with low delay jitter, which can be incorporated into an IP router. The GR scheduling algorithm proposed in [42]–[44] is computationally very efficient and offers very low delay jitters and is explored in this paper.

Section II describes video traffic model. Section III describes some prior guaranteed-rate scheduling algorithms from the literature. Section IV describes the low-jitter scheduling algorithm presented in [42]–[47] in more depth. Section V describes the IP multicasting and network simulation environment, and presents detailed simulation results on traffic delivery through one IP router. Section V presents detailed statistics on queue occupancies and inter-arrival time distributions, on selected IP routers and links in the multicast tree of Fig. 2. Section VI contains concluding remarks.

II. Video Traffic Model

Cisco Systems estimates that up to 200 video channels requiring 750 Mbps may be distributed over the IP backbone to support emerging IPTV applications [2]. To gather realistic data for our simulations, a high definition video stream entitled '*From Mars to China*' available at the University of Arizona [17] web-site was processed. The video stream includes ≈51 K video frames which arrive at the rate of 30 video frames/sec. The minimum, mean and maximum video frame sizes are ≈20 K, 80 K, 320 K bytes respectively, illustrating a very bursty behavior. We assume these video frames are disassembled into fixed-size 64-byte cells before transmission into the IP multicast tree. These video frame sizes correspond to a mean of 1,280 cells per video frame, with a minimum and maximum of 320 and 5,120 cells per video frame respectively. The single video stream has a compression ratio of 154, with a mean bit rate of 4.85 Mbps, and a peak bit rate of 78 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 web-site [17] provides video frame size statistics for only one high-definition video channel. According to Cisco, a typical service provider may multicast 200 MPEG2 digital video channels, each at a mean rate of 3.75 Mbps, for an aggregate stream traffic rate of 750 Mbps [2]. In this paper, we assume 75 high-definition video channels are to be multicast, each with a 4.63 Mbps average rate, for an aggregate stream traffic rate of 347.4 Mbps. To achieve the statistics for the 75 video channels used in our network simulations, the single video channel data from [17] was re-used; Each simulated video channel, from 1...75, was assigned a random starting video frame number from 1...51 K, and the video frame size sequence for the single video channel in [17] was reused in a circular manner.

Table I lists some properties of the single video channel and several aggregated video traffic streams. The single video channel has a mean data rate of 4.632 Mbps, with a peak rate of 74.82 Mbps. In Table I, the ratio of the peak-to-mean rates is an indication of the burstiness of the traffic. The single channel has a ratio of 16.15, indicating a high degree of burstiness. Referring to Table I, the aggregated stream of 75 channels has an aggregate data rate of 347.4 Mbps, with a peak rate of 601.3 Mbps. The ratio of peak-to-mean rates is 1.731, indicating a considerable reduction in burstiness.

Fig. 3 illustrates visually the effect of aggregation of multiple video channels on the burstiness of the aggregated video stream. Fig. 3(a) illustrates the instantaneous bandwidth versus time, for the aggregation of 25, 50 and 75 video channels. The aggregation of multiple video channels into a single stream results in a proportionally higher mean rate, and a much lower ratio of the peak-to-mean rates. Fig. 3(b) illustrates the instantaneous normalized bandwidth versus time for the same aggregated streams.

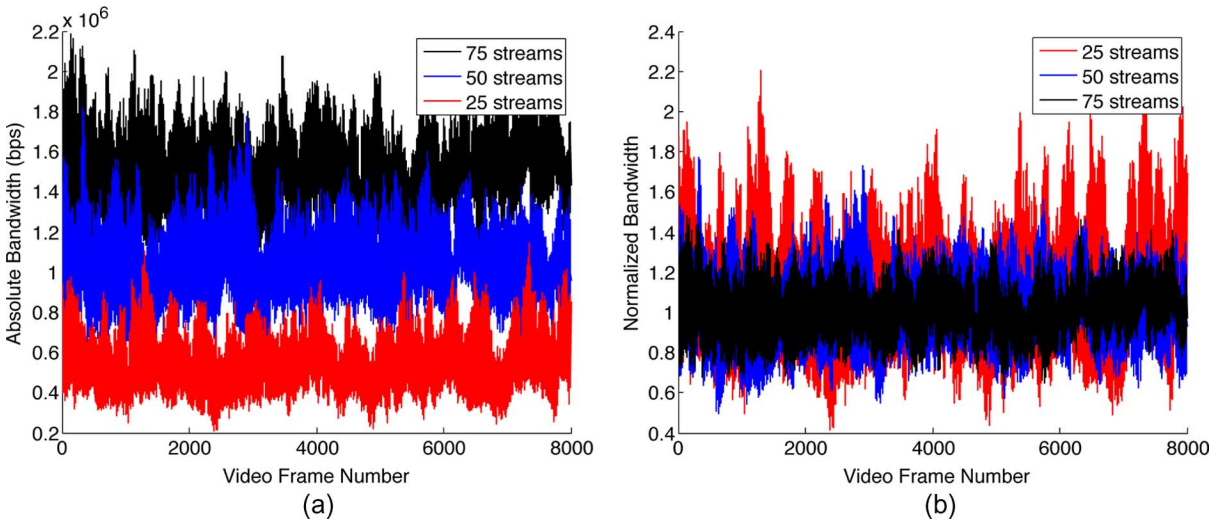

Fig. 3. (a) Instantaneous BW versus video frame number; (b) normalized instantaneous BW versus video frame number.

TABLE I
STATISTICS ON AGGREGATED VIDEO TRAFFIC STREAMS

# Video chan-nels.	Mean Data Rate (Mbps)	Std Dev.	Peak Data Rate (Mbps)	Ratio (Peak/ Mean)	GR (Mbps) 72% util.	Time-Slot Reserva-tions
1	4.632	0.230	74.82	16.153	N/A	N/A
25	115.8	0.917	278.6	2.406	160	16
50	231.6	1.354	446.7	1.929	320	32
75	347.4	1.499	601.3	1.731	480	48
100	463.2	1.546	747.1	1.613	640	64
200	926.4	2.586	1313	1.417	1280	128
400	1852	3.729	2374	1.282	2560	256

The mean rate of each stream has been normalized to 1, and the reduction in burstiness when many video channels are aggregated is evident. Video traffic is generally considered to exhibit trends of self-similarity, i.e., it exhibits burstiness and correlations which can be observed over several different time scales. It is known that the aggregation of self-similar traffic generally yields self-similar traffic, which also exhibits burstiness over large time-scales. Fig. 3(b) illustrates that the correlations and burstiness of the aggregated video stream does decrease significantly, however it still appears to exhibit some self-similar behavior.

There are several methods to handle bursty traffic at the source of the network [48], [49]: (a) A *Leaky/Token Bucket* traffic shaping module can be used at the source to smoothen bursty traffic to conform to an appropriate mean traffic rate, which will introduce an *application-specific* delay at the source which is independent of the network [48], [49]; (b) The IP routers along a path can be configured to provide GR bandwidth sufficient to meet the peak traffic rate of a flow, minimizing the delay contribution at the source; and (c) The intermediate IP routers can be configured to provide GR bandwidth sufficient to meet the average rate of a flow, and bursts are left to be handled by the best-effort dynamic packet scheduler within each IP router, which will introduce a delay and jitter contribution in each the IP routers. Assume method (a) is used here, consistent with [48], [49] which use method (a) to determine theoretical delay bounds for an end-to-end path of OQ switches using a Weighted Fair Queueing (WFQ) scheduling algorithm. However, method (b) can also be used if the application warrants that bursts be preserved across the Internet.

Referring to Fig. 2, assume that the 75 video channels are available at the root of the IP multicast tree for distribution, consistent with Cisco's assumptions [2]. The aggregated stream of 75 channels has an average data rate of 347.4 Mbps, and a peak rate of 601.3 Mbps, and the IP multicast tree must be provisioned to support this traffic. There are several choices of *Token Bucket* traffic shaper parameters and IP tree provisioning parameters that will yield acceptable solutions. In this paper, we assume that the IP multicast tree is provisioned such that the average data rate of the aggregated video stream (347.4 Mbps) consumes 72% of the provisioned tree link capacity, thereby providing 28% excess capacity for bursts. Therefore, the IP multicast tree must be provisioned to support 480 Mbps of guaranteed rate traffic on every link in the tree. Given the line-rate of 10 Gbps and the use of 64-byte cells, this guaranteed rate requires the transmission of 48 cells per scheduling frame of length 1024, or $\approx$4.7% of the line-rate. Observe from Table I that the maximum (burst) data rate of the aggregated stream equals 601.3 Mbps, which exceeds the provisioned Guaranteed Rate of 480 Mbps. Therefore when bursts of cells arrive at the traffic shaping module, these cells will be temporarily stored in the traffic shaping module, and will be released into the network at the maximum data rate of 480 Mbps.

The root of the multicast tree has a *Token Bucket* traffic shaper module, with a token bucket depth of zero and with infinite storage capacity for video cells. The module is configured for a maximum data rate of 480 Mbps. Assume that video frames associated with any one video channel will arrive to the traffic shaper queue at the fixed rate of 30 frames per second. The frame

arrival rate for the aggregated stream of 75 channels is therefore 2,250 video frames per second, with an average inter-arrival time (IAT) of $1/2,250 = 0.44$ millisec. To simplify the network simulation model, assume that video frames arrive to the traffic shaper module at a constant rate of 2,250 frames/sec, with an IAT $= 0.444$ millisec. The arriving traffic is quite bursty, given that each video frame has a mean size of 1,280 cells, with a min/max size of 320/5,120 cells respectively. Cells are allowed to depart the shaper module at the maximum rate of 2,250 cells/sec, when cells are available.

The root node A in Fig. 2 implements a 1-to-4 multicasting of the cells arriving on input port 3. These cells appear on output ports 2, 4, 5 and 7. The nodes D and H in the tree in Fig. 2 implement 1–4 and 1–3 multicasting as well. There are potentially hundreds or thousands of aggregation nodes in the IP multicast tree, distributing content to potentially hundreds of thousands, or millions, of end-users (i.e., households). Each end-user node has a playback buffer which receives the video cells corresponding to one selected video channel. The purpose of the playback queue is to filter out residual network-introduced jitter and deliver reconstructed video frames to the end-user with very low delay jitter. In an ideal situation, the playback buffer will reconstruct a video frame for the desired video channel at the rate of 30 frames per second, at each destination IP router. Each video frame must be reconstructed from numerous 64-byte cells.

IP networks typically transmit variable-size IP packets. Packets are typically disassembled into fixed sized cells at the input size of each IP router, and IP packets are re-assembled at the output size of the IP router, before they are transmitted to the next IP router. The use of variable-size IP packets typically leads to large delays and jitter associated with disassembling and re-assembling 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 of 64 bytes. IP packets are disabled 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 large IP packets at each IP router, and reduces the delay and jitter in each IP router.

III. Prior Guaranteed-Rate Scheduling Algorithms

Several schemes have been proposed for scheduling GR traffic through an IQ packet switch [32]–[41], which are briefly reviewed. In the Birkoff von-Neuman (BVN) scheme proposed in [37], [38], a doubly stochastic traffic rate matrix is decomposed into a sequence of permutation matrices and associated weights. These matrices are then scheduled using the GPS or WFQ algorithm [48], [49], 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. BVN decomposition has a time complexity of $O(N^{4.5})$ and is generally considered too slow for use in real-time packet-switched IP routers.

The problem of scheduling traffic through an IQ packet-switched IP router, while attempting to simultaneously minimize the service lag among multiple competing IP flows and to minimize the speedup, was considered in [35]. A doubly stochastic traffic rate matrix is first quantized to contain integer values and then 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 over all IO pairs is bounded by $O((N/4)(S/(S-1)))$ time-slots. According to [35]; "*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 with the goal to minimize delay jitter among simultaneous competing IP flows through an Input-Queued packet-switch was introduced in [39], [40]. 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 which is 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 $O(logN)$. Hard analytic bounds on the jitter were not available.

A heuristic scheduling algorithm for multicasting cells in an IQ switch is described in [23]. The heuristic algorithm is validated through extensive simulations in [23]. The input queues were configured to have infinite capacity, so that no cells would be dropped. Cell latencies through one IP router were reported: Latencies varied from as low as 1 time-slot at very light loads, to 1000 time-slots at heavy loads.

IV. Low-Jitter GR Scheduling

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

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

Each element $\lambda_{j,k}$ represents the fraction of the transmission line rate reserved for GR traffic between IO pair (j,k). The transmission of cells through the switch is governed by the 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:

$$R = \begin{bmatrix} R_{0,0} & R_{0,1} & \cdots & R_{0,N-1} \\ R_{1,0} & R_{1,1} & \cdots & R_{1,N-1} \\ \cdots & & & \cdots \\ R_{N-1,0} & R_{N-1,1} & \cdots & R_{N-1,N-1} \end{bmatrix}, \begin{matrix} \sum_{i=0}^{N-1} R_{i,j} \leq F, \\ \sum_{j=0}^{N-1} R_{i,j} \leq F \end{matrix}$$

TABLE II
STATISTICS, 8 × 8 IP ROUTER, 100% SATURATED ($N = 8, F = 1024$)

Flow Class	IIDT in time-slots	Observed IDT in time-slots	Std. Dev. time-slots	Min IDT time-slots (IIDT)	Max IDT time-slots (IIDT)	Min Norm. Lead (IIDT)	Max Norm. Lag (IIDT)
36	28.44	28.44	16.1	1 (0.035)	96 (3.38)	-2.36	1.79
40	25.60	25.60	13.6	1 (0.039)	93 (3.63)	-3.1	1.95
44	23.27	23.27	12.7	1 (0.043)	82 (3.52)	-3.07	1.98
48	21.33	21.33	11.4	1 (0.047)	85 (3.98)	-2.98	1.81
52	19.69	19.69	10.3	1 (0.051)	63 (3.20)	-3.20	2.18
56	18.29	18.29	9.38	1 (0.055)	59 (3.23)	-3.25	2.48
60	17.07	17.07	8.62	1 (0059)	59 (3.46)	-3.08	2.00
64	16.00	16.00	7.70	1 (0.063)	53 (3.31)	-3.00	1.88

Several of the following definitions will be useful (see [35], [42], [46] for similar definitions).

Definition: 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 $\text{bandwidth} = L/F$. To set the minimum quota of reservable bandwidth to $\leq$1% of L, set $F \geq 100$, i.e., $F = 128$.

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

Definition: 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: 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, i.e., $Lij(t) = Sij(t) - (t/F)Rij(t)$. A positive Service Lag denotes the case where the received service is less than the requested service, i.e., 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, i.e., a cell arrive sooner than its ideal service time.

Example #1: Consider an 8×8 crossbar switch with a frame size of $F = 1024$, operating at 100% utilization for GR traffic, a very heavy and worst-case load. Application of the scheduling algorithm in [42]–[44] on 1,000 randomly generated fully-saturated traffic rate matrices yields the results in Fig. 4 and Table II. Each matrix represents 64 simultaneous GR traffic flow requirements, and all 1,000 matrices represent 64,000 GR traffic flow requirements to be met. Each flow contains on average 128 cells, so that the 1,000 matrices represent $64,000 * 128 = 8.192$ million cells to be scheduled.

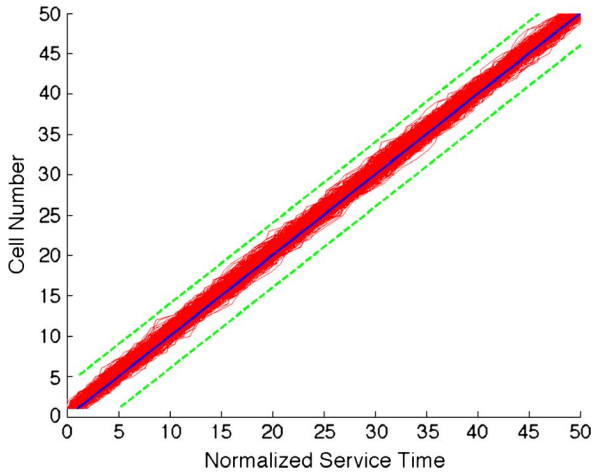

Fig. 4. Normalized service, $\mathrm{N} = 8$, $\mathrm{F} = 1024$.

Fig. 4 illustrates the normalized received service for the 8 × 8 packet switch. The solid diagonal line represents ideal normalized service time, where the actual departure time of cell j equals the ideal departure time of $j \cdot IIDT$. The upper/lower dashed diagonals represent a service lead/lag of $4 \cdot \mathrm{IIDTs}$. The received service is normalized by the IIDT, such that a cell which departs $2 \cdot \mathrm{IIDTs}$ after its ideal departure time has a service lag of 2.

Table II illustrates statistics on the observed '*Inter-Departure Times*' (IDTs) for selected classes of flow. A flow which reserves j time-slots per frame is said to belong to class j. The IIDT for each flow class is shown in column 2. The observed IDT and the standard deviation are shown in columns 3 and 4. The minimum and maximum observed IDTs are shown in columns 5 and 6, expressed in time-slots and in terms of IIDTs. The minimum service lead and the maximum service lag are shown in columns 7 and 8, expressed in terms of IIDTs. Observe that (a) the average observed IDT equals the IIDT, and that (b) the standard deviation of the cell IDT is very small, typically equaling about one half of the IIDT, for every class of flow shown. Table I indicates that cell departures tend to be evenly distributed over the frame. Table I is consistent with theoretical bounds on the service lead and lag established in [42], [43].

The results in Fig. 4 and Table II indicate that the service lead/lag of the scheduling algorithm are small, and suggests that GR traffic can be transported across the IP backbone with very low delay jitter provided each IP router has sufficient buffer space to compensate for any service lead/lag it may experience. According to Fig. 4 and Table II, the service lead/lag is typically less than 4 IIDTs, suggesting that the buffering of 4 cells per IP router per flow may suffice. (Theoretical bounds are presented in [45].)

V. Results of IPTV Multicasting

Assume an IP/MPLS backbone with 10 Gbps links, with a frame length $F = 1024$ (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 frame, or equivalently 10 Mbps. The provisioned GR rate of 480 Mbps established in Section II requires 48 time-slot reservations per frame. Each IP router in Fig. 2 must reserve and schedule 48 cell transmissions per frame between the appropriate IO pairs. Assume that every other incoming and outgoing link in each IP router will be 100% saturated with other GR traffic. This assumption represents a very heavy load for GR traffic, not likely to be seen in a real IP network. Given this worst-case load, the IP routers should find it more challenging to schedule the traffic to meet QoS guarantees.

The performance of the packet-switched IP multicast tree was evaluated using a discrete-event simulator written in the C programming language, with over 20,000 lines of code. The simulation was run on a large cluster-based supercomputing system in the Dept. of ECE at McMaster University, with 160 dual processing nodes running the LUNIX 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.

Table III presents statistics on the number of late video frames delivered at an end-user, versus the Playback queue depth at the end-user. Each row in Table III represents a distinct playback queue depth in milliseconds, while each column exhibits the number of late video frames observed along a specific path of the IP multicast tree. Each entry was obtained by the simulation of 60 seconds of real video traffic from the aggregated traffic stream, i.e., $75 * 60 * 30 = 135,000$ real video frames, and was repeated 15 times to compute a standard deviation. From Section II, recall that each video frame represents on average 1,280 cells, so that each entry in Table II represents the simulation of $15 * 135,000 * 1,280 = 2.592$ billion cells on average. Table III has 50 entries, representing the simulation of 129.6 billion cells. The entire simulation was distributed over the 160 nodes in our computing cluster, and represents an aggregate simulation time of over 500 CPU hours. The IP routers were configured with randomly generated doubly stochastic traffic rate matrices that were fully saturated, and which supported the multicasting required for the IP multicast tree in Fig. 2.

The criterion for correct delivery of a video frame was as follows: Video frames are delivered to the end-user at the ideal rate of 30 video frames/second, after the playback queue has buffered up data during its initialization period. The duration of the initialization period is called the playback Q depth in Table III. A video frame which could not be delivered at its scheduled delivery time was classified as late. Five specific paths through the IP multicast network in Fig. 2 were simulated, paths AB, ADF, ADHL, ADHK, and ADHJ, to observe the statistical behavior of the broadcast traffic stream at varying distances from the root of the tree. Referring to Table III and assuming a playback Q depth of 5 milliseconds, 2,317 video frames which traversed path AB were observed to be late at the destination playback buffer, corresponding to a late frame rate of 1.72 percent. Along paths ADHL, ADHK and ADHJ, the observed late frame rates were also ≈1.72 percent. The late frame rate is essentially independent of the distance of the end-user from the root of the multicast tree. Assuming a playback Q depth of 10 milliseconds, the late frame rates along the same 4 paths are essentially identical and equal to 0.014 percent. Using a playback Q depth of 15 milliseconds or greater, no late video frames were observed on any path. Table III illustrates that numerous video streams can be multicast over an IP/MPLS network, with essentially-zero delay jitter and essentially-zero cell loss rate at each receiver, using a relatively small playback Q depth of 15 milliseconds. Recall that each of the 50 entries in Table III represents the simulation of 15 minutes of the aggregated traffic stream, representing 2.592 billion cells, using real video trace data provided by [21], resulting in fairly small standard deviations.

TABLE III
Number Late Video Frames versus Playback Q Depth, on Selected Network Paths

Playback Q depth (millisec)	Path AB	Path ADF	Path ADHL	Path ADHK	Path ADHJ	Std Dev.
1	32,753	32,784	32,815	32,815	32,820	9,480
2	17,759	17,777	17,796	17,795	17,798	5,820
3	9,421	9,430	9,440	9,439	9,441	3,620
5	2,317	2,319	2,323	2,323	2,323	1,280
7.5	287	291	291	291	291	260
10	18	19	19	19	19	32
15	0	0	0	0	0	0
20	0	0	0	0	0	0
30	0	0	0	0	0	0

Fig. 5(a) illustrates the occupancy of the shaper queue and playback queue versus time, along path ADHJ, assuming a playback Q depth of 15 millisec. The shaper queue begins receiving video frames at time $t = 0$, and exhibits a step increase as each video frame arrives, with a magnitude proportional to the number of cells in the video frame. Thereafter, video frames arrive every $(1/75) * 33.3$ millisec, i.e., every 0.444 millisec. The shaper queue occupancy (in cells) also decreases linearly with time in between arrival instances, corresponding to the departure of cells into the IP network at the guaranteed rate of 480 Mbps. Observe that the shaper queue often empties before the next video frame arrives, indicating that the provisioned guaranteed traffic rate of 480 Mbps on the broadcast tree is sufficient to handle this aggregated traffic stream rate of 347.3 Mbps.

The playback Q is initially empty at $t = 0$, and receives cells corresponding to one video stream out of the 75 aggregated video streams being broadcast. (We assume each video channel will have its own playback Q.) The playback Q exhibits a linear increase in cell occupancy starting at $t = 0$ as it receives

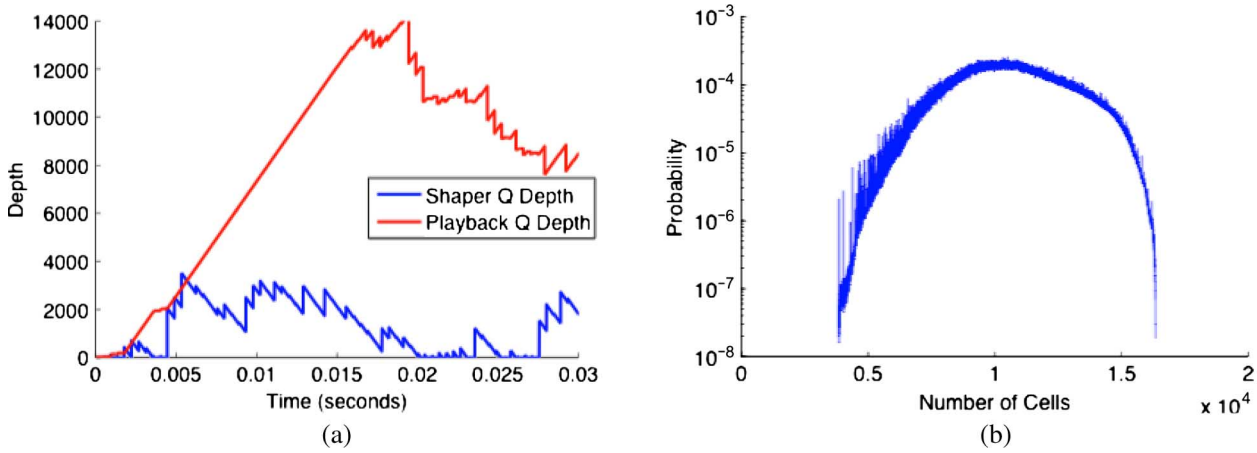

Fig. 5. (a) Shaper and playback Q cell occupancy vs time, $\mathrm{Depth} = 15$ millisec; (b) playback Q cell occupancy PDF.

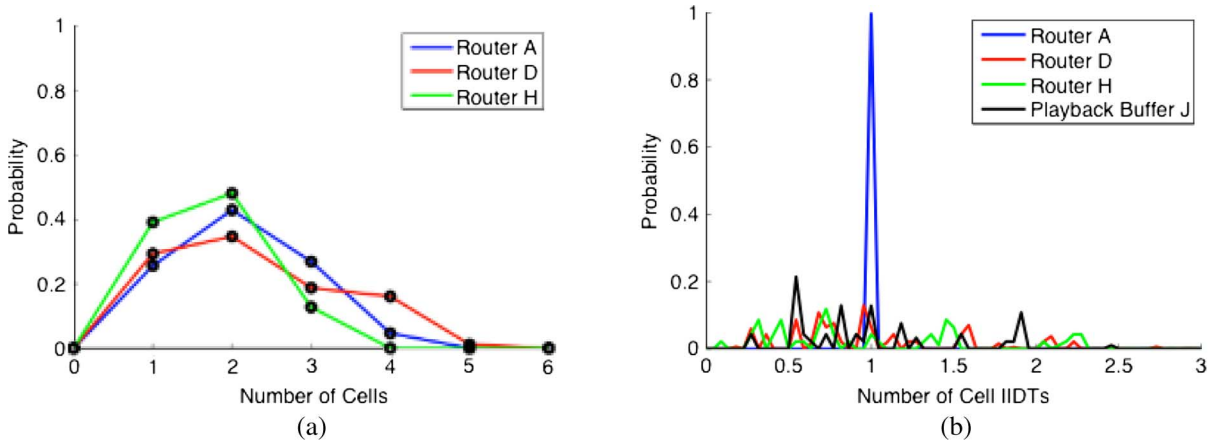

Fig. 6. (a) Queue size PDF at selected IP routers; (b) IAT PDF at selected IP routers.

cells from the IP multicast tree. After the initialization period of 15 millisec, the playback queue begins to release reconstructed video frames at the fixed rate of 1 video frame every 33.3 millisec. It exhibits stepwise decreases in the cell occupancy count, with a magnitude proportional to the number of cells in each video frame, as these video frames are released to the end-user. Observe that the playback Q never becomes empty, i.e., it always has a sufficient number of stored cells to be able to deliver video frames at the guaranteed rate of 1 frame every 33.3 millisec.

Fig. 5(b) illustrates the probability density function (PDF) for the discrete occupancy of the playback queue in cells, assuming a playback queue depth of 15 millisec. The average number of cells is approx 10,000, with a min/max of 4,000/16,000 cells respectively. Each video frame requires an average of 1,280 cells, indicating that the playback Q stores on average of ≈7.8 video frames of data.

Table IV illustrates statistics for the number of cells and the queueing delay in selected IP routers along a specific path ADH, for a Playback queue depth of 15 millisec. In the steady-state, the shaper queue buffers 1,040 cells and contributes a delay of 1.52 millisec on average. The playback queue buffers 10,050 cells and contributes a delay of 14.68 millisec on average. The shaper and playback queues are external to the IP network. Referring to Table IV, the average number of cells buffered within the IP routers A, D and H is 1.996 cells per router, consistent with Fig. 4 and Table II, which indicated that each flow will buffer $\leq$4 cells in an IP router on average.

TABLE IV
VIRTUAL OUTPUT QUEUE STATISTICS, SELECTED IP ROUTERS

Component	E(Nq) (cells)	E(Wq) microsec
Shaper Q	1,040	1,520
Router A	2.106	3.076
Router D	2.246	3.281
Router H	1.638	2.393
Playback Q	10,050	14,680

Fig. 6(a) illustrates the PDF for the distribution of the number of cells from the aggregated video stream queued in each of the 3 selected IP routers in Fig. 2, along path ADH. The number of cells queued in each router is small, 2 cells on average per multicast output port, and the distribution is tight, i.e., the deviation from the mean is small. No router buffers more than 6 cells per multicast output port for this aggregated flow. These curves indicate that the cells move at a relatively consistent rate through the IP network.

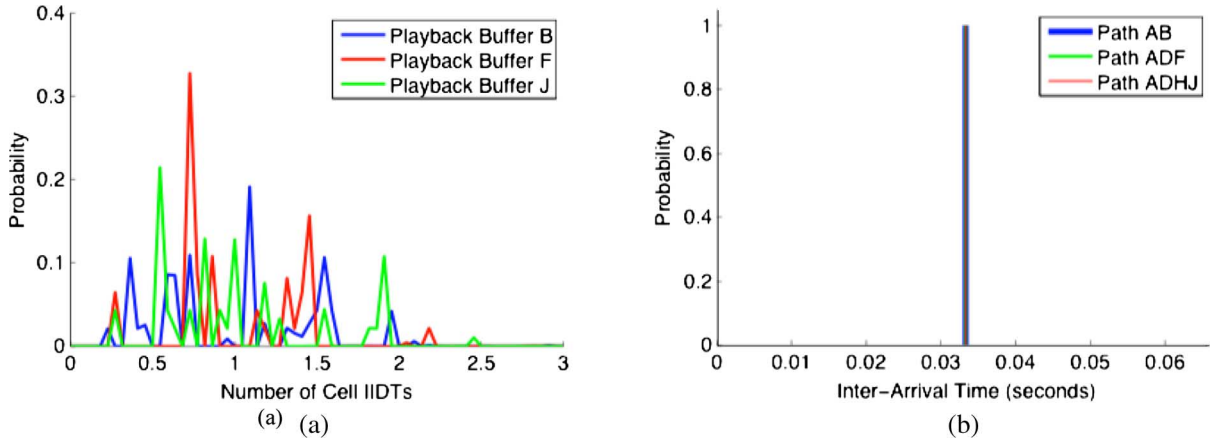

Fig. 7. (a) IAT PDF of cells at selected playback buffers; (b) IAT PDF of reconstructed video frames at selected destination IP routers, with zero delay jitter.

Fig. 6(b) illustrates the PDF for the inter-arrival time (IAT) of cells arriving to these 3 IP routers, conditional on the event that the shaper Q is non-empty. This condition omits the large inter-arrival times of cells which terminate an idle-period during which the shaper Q has no cells to transmit. Cells arriving to IP router A have an IAT equal to 1 cell every IIDT interval, corresponding to the perfect departure of cells from the shaper Q at the ideal rate of 1 cell every IIDT, when cells are available. This departure rate is set by the Leaky Bucket Shaper parameters. Cells arriving to IP router D have an IAT of between 0.25 and 2.75 IIDTs, indicating a small jitter on the path. Similarly, cells arriving to router H and to the playback queue J exhibit a small jitter, between 0.1 and 2.5 IIDTs.

Fig. 7(a) illustrates the IAT PDFs of cells arriving to 3 playback queues at different depths of the tree in Fig. 2, along paths AB, ADF, and ADHJ. The IATs vary from between 0.25 and 2.5 IIDTs, indicating a relatively small jitter along any path. Fig. 7(b) illustrates the IAT PDF of video frames delivered to the end-users from the playback buffers at nodes B, F and J, of the multicast tree in Fig. 2. Fig. 7(b) illustrates that the IAT is exactly 33.3 millisec and that the delay jitter is precisely zero. The playback Q has reconstructed the video frames, filtered out the residual network jitter and delivered video frames to the destination IP routers in the multicast tree with essentially-zero delay jitter, with a relatively small playback Q depth of 15 millisec. This result is consistent with Table III, which indicated that the delay jitter is precisely zero when the playback queue depth is $\geq$15 millisec.

VI. Conclusion

A technology to multicast multimedia streams such as IPTV over a packet-switched Internet backbone with essentially-zero delay jitter and essentially-zero packet loss rate has been presented. A Guaranteed Rate scheduling algorithm [42]–[44] is used to compute deterministic low-jitter transmission schedules for each IQ packet-switched IP router in the multicast tree. The low-jitter GR scheduling algorithm removes much of the cell delay variability associated with the dynamic scheduling algorithms used in existing IP routers, and minimizes the amount of buffering required in the IP routers. Extensive simulations indicate that essentially-zero delay jitter is achievable, as confirmed by theoretical results [42], [43], [45] and other experimental results [46], [47]. The extensive simulations indicate that each IP router typically buffers 2 cells (128 bytes) of video data per output port for the aggregated IPTV flow being multicast, significantly less buffering than current IP routers require. The technology is also applicable to low-jitter multicasting of generic multimedia streams over the Internet, including VOIP, Video-on-Demand, Telemedicine and Telerobotic control over IP.

References

[1] Cisco Systems White Paper, "Optimizing Video Transport in your IP Triple Play Network," 2006 [Online]. Available: www.cisco.com
[2] A. Sardell, Juniper Networks, "Video Transport on an MPLS Backbone," White Paper, Mar. 2005 [Online]. Available: www.junipercom
[3] Cisco Systems New Release, "Bell Canada Utilizes Cisco Systems Technology to Help Deliver Surgical Grade Network to Power Historic Telerobotics Assisted Surgery," 2003 [Online]. Available: http://newsroom.cisco.com/dlls/prod_030403.html
[4] Internet Engineering Task Force (IETF), Request for Comments RFC-3393, "IP Packet Delay Variation," Nov. 2002 [Online]. Available: http://tools.ietf.org/html/rfc3393
[5] Y. Siao, X. Du, J. Zhang, F. Hu, and S. Guizani, "Internet Protocol Television (IPTV): The Killer Application for the Next-Generation Internet," *IEEE Comm. Mag.*, pp. 126–134, Nov. 2007.
[6] W. Sun, K. Lin, and Y. Guan, "Performance analysis of a finite duration multichannel delivery method in IPTV," *IEEE Trans. Broadcasting*, vol. 54, no. 3, pt. 1, pp. 419–429, Sep. 2008.
[7] H. Joo, H. Song, D.-B. Lee, and I. Lee, "An effective IPTV channel control algorithm considering channel zapping time and network utilization," *IEEE Trans. Broadcasting*, vol. 54, no. 2, pp. 208–216, Jun. 2008.
[8] U. Jennehag, T. Zhang, and S. Pettersson, "Improving transmission efficiency in H.264 based IPTV systems," *IEEE Trans. Broadcasting*, vol. 53, no. 1, pt. 1, pp. 69–78, Mar. 2007.
[9] D. lee, H. Joo, and H. Song, "An effective channel control algorithm for integrated IPTV services over DOCSIS CATV networks," *IEEE Trans. Broadcasting*, vol. 53, no. 4, pp. 789–796, Dec. 2007.
[10] S. Gulliver and G. Ghinea, "The perceptual and attentive impact of delay and jitter in multimedia delivery," *IEEE Trans. Broadcasting*, vol. 53, no. 2, pp. 449–458, Jun. 2007.
[11] G. Muntean, P. Perry, and L. Murphy, "A new adaptive multimedia streaming system for All-IP multi-service networks," *IEEE Trans. Broadcasting*, vol. 50, no. 1, pp. 1–10, Mar. 2004.
[12] S. Chand and H. Om, "Modeling of buffer storage in video transmission," *IEEE Trans. Broadcasting*, vol. 53, no. 4, pp. 774–779, Dec. 2007.

[13] S. Aissa and G. Aniba, "Queuing models for dimensioning interactive and streaming services in high-speed downlink packet access networks," *IEEE Trans. Broadcasting*, vol. 53, no. 3, pp. 619–627, Sep. 2007.
[14] Y. Bai and R. Ito, "Application-aware buffer management: New metrics and techniques," *IEEE Trans. Broadcasting*, vol. 51, no. 1, pp. 114–121, Mar. 2005.
[15] W. Poon, K. Lo, and J. Feng, "Adaptive batching scheme for multicast video-on-demand systems," *IEEE Trans. Broadcasting*, vol. 47, no. 1, pp. 66–70, Mar. 2001.
[16] C. Choi and M. Hamdi, "A scalable video-on-demand system using multi-batch buffering techniques," *IEEE Trans. Broadcasting*, vol. 49, no. 2, pp. 178 –191, Jun. 2003.
[17] P. Seelingm, M. Reisslein, and B. Kulapa, "Network performance evaluation using frame size and quality traces of single layer and two layer video: A tutorial," *IEEE Comm, Surveys*, vol. 6, no. 3, pp. 58–78, 3rd Q 2004.
[18] L. Hardesty, "Internet Gridlock: Video is clogging the Internet. How we choose to unclog it will have far-reaching implications," *MIT Technology Review*, July/August 2008.
[19] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Automatic Control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.
[20] V. Anantharam, N. McKeown, A. Mekittikul, and J. Walrand, "Achieving 100% throughput in an input queued switch," *Trans. Comm.*, vol. 47, no. 8, pp. 1260–1267, 1999.
[21] A. Gourgy, T. H. Szymanski, and D. Down, "On tracking the behaviour of an output queued switch using an input queued switch," *IEEE Trans. Networking*, 2006, submitted for publication.
[22] N. McKeown, "The iSLIP scheduling algorithm for input queued switches," *IEEE Trans. Networking*, vol. 7, no. 2, pp. 188–201, Apr. 1999.
[23] B. Prabhakar, N. Mckeown, and R. Ahuja, "Multicast scheduling for input-queued switches," *IEEE Journal Selected Areas in Comm.*, vol. 15, no. 5, pp. 855–864, Jun. 1997.
[24] E. Leonardi, M. Mellia, F. Neri, and M. A. Marsan, "Bounds on the delay and queue lengths in input-queued cell switches," *JACM*, vol. 50, no. 4, pp. 520–550, Jul. 2003.
[25] P. Giaccone, E. Leonardi, and D. Shah, "Throughput region of finite-buffered networks," *IEEE Trans. Parallel and Dist. Systems*, vol. 18, no. 2, pp. 251–263, Feb. 2007.
[26] Y. Ganjjali and N. McKeown, "Update on buffer sizing in internet routers," *ACM Sigcomm Comp. Comm. Rev.*, vol. 36, no. 5, pp. 67–70, Oct. 2006.
[27] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," in *Sigcomm'04*, NY, USA, 2004, pp. 281–292.
[28] G. Raina and D. Wishick, "Buffer sizes for large multiplexers: TCP queueing theory and instability analysis," in *EuroNGI*, Rome, Italy, April 2005.
[29] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, "Routers with very small buffers," in *IEEE Infocom*, Barcelona , Spain, Apr. 2006, pp. 1–11.
[30] A. Dhamdhere and C. Dovrolis, "Open issues in router buffer sizing," *ACM/SIGCOMM Comp. Comm. Rev.*, vol. 36, no. 1, pp. 87–92, Jan. 2006.
[31] G. Vu-Brugier, R. S. Stanojevic, D. J. Leith, and R. N. Shorten, "A Critique of recently proposed buffer sizing strategies," *ACM/SIGCOMM Comp. Comm. Rev.*, vol. 37, no. 1, pp. 43–47, May 2007.
[32] P. Goyal and H. M. Vin, "Generalized guaranteed rate scheduling algorithms: A framework," *IEE/ACM Trans. Networking*, vol. 5, no. 4, pp. 561–571, Aug. 1997.
[33] 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, Dec. 1997.
[34] A. Hung, G. Kesidis, and N. McKeown, "ATM input buffered switches with guaranteed rate property," in *Proc. IEEE ISCC Conference*, 1998, pp. 331–335.
[35] C. E. Koksal, R. G. Gallager, and C. E. Rohrs, "Rate quantization and service quality over single crossbar switches," in *IEEE Infocom*, 2004, pp. 1962–1973.
[36] 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.
[37] 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 Neumann," in *IEEE iWQoS'99*, 1999, pp. 79–86.
[38] W. J. Chen, C.-S. Chang, and H.-Y. Huang, "Birkhoff-von Neumann input buffered crossbar switches," in *Proc. Infocom 2000*.
[39] 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, pp. 1364–1375, Dec. 2005.
[40] 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.
[41] S. R. Mohanty and L. N. Bhuyan, "Guaranteed smooth switch scheduling with low complexity," in *IEEE Globecom*, 2005, pp. 626–630.
[42] T. H. Szymanski, "QoS switch scheduling using recursive fair stochastic matrix decomposition," in *IEEE High Performance Switching and Routing Conf (HPSR)*, 2006, pp. 417–424.
[43] T. H. Szymanski, "Low-jitter guaranteed-rate scheduling algorithm for packet-switched IP routers," *IEEE Trans. Commun.*, 2008, Accepted (with revision), to be published.
[44] T. H. Szymanski, "Method and Apparatus to Schedule Packets through a Crossbar Switch with Delay Guarantees," U.S. , 2007, U.S. Patent Application.
[45] T. H. Szymanski, "Bounds on End-to-End Delay and Jitter for Input-Queued IP/MPLS Networks," Submitted, 2008.
[46] T. H. Szymanski and D. Gilbert, "Delivery of guaranteed rate internet traffic with very low delay jitter," in *IEEE Pacific Rim Conf. on Computers, Communications and Signal Processing*, 2007, pp. 450–455.
[47] T. H. Szymanski and D. Gilbert, "Low-jitter guaranteed-rate communications for cluster computer systems," *Pacific-Rim Special Issue, Inderscience Int. Journal of Computer Networks and Distributed Systems*, vol. 1, no. 2, pp. 140–160, 2008.
[48] 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, no. 3, pp. 344–357, Jun. 1993.
[49] 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, Apr. 1994.

Ted H. Szymanski (M'82) holds the Bell Canada Chair in Data Communications at McMaster University. He completed his Ph.D. at the University of Toronto, and has held faculty positions at Columbia University, where he was affiliated with the Center for Telecommunications Research (CTR), and McGill University, where he was affiliated with the Canadian Institute for Telecommunications Research (CITR). From 1993 to 2003, he was an architect in a national research program on Photonic Systems funded by the Canadian Networks of Centers of Excellence (NCE) program. Industrial and academic collaborators included 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-optical channels packaged per square centimeter of bisection area, for which he holds two patents. His current interests include switching, scheduling, and network QoS.

Dave Gilbert (M'97) completed his Ph.D. in the Dept. of Electrical and Computer Engineering (ECE) at McMaster University in 2007, in the area of nuclear reactor modeling. He is currently a Post-Doctoral Fellow in the Department of ECE. His research interests include nuclear reactor modeling, software-based problem-solving environments, and network performance modeling.