Macrothink Institute™

Network Protocols and Algorithms
ISSN 1943-3581
2017, Vol. 9, No. 1-2

A New Packet Scheduling Algorithm for Real-Time Multimedia Streaming

Mehmet Şimşek

Dept. of Computer Engineering, Faculty of Engineering, Düzce University

Konuralp, 81620, Düzce (Turkey)

E-mail: mehmetsimsek@duzce.edu.tr

Nurettin Doğan

Dept. of Computer Engineering, Faculty of Technology, Gazi University

Teknikokullar, 06500, Ankara (Turkey)

E-mail: ndogan@gazi.edu.tr

Muhammet Ali Akcayol

Dept. of Computer Engineering, Faculty of Engineering, Gazi University

Teknikokullar, 06570, Ankara (Turkey)

E-mail: akcayol@gazi.edu.tr

Received: October 5, 2016 Accepted: March 29, 2017 Published: June 30, 2017

DOI: 10.5296/npa.v9i1-2.10943 URL: https://doi.org/ 10.5296/npa.v9i1-2.10943

Abstract

Delivering the real-time services over converged networks is a big challenge. Real-time services need to high Quality of Service (QoS). For this purpose, bandwidth reservation and packet prioritization techniques are used. Thus, real-time data packets can be reached to their targets with minimum delays and losses. But, this situation creates unintended consequences for other internet services such as HTTP and FTP. In this case, establishing a balance between the real-time services and the other services is a must. In this study we introduce a new research question: how to transport real-time multimedia IP packets just in

time? Just in time means that transportation of the packets neither early, nor late. For this purpose we developed a scheduling/prioritizing algorithm called just in time transport (JITT). Following a cross-layer design approach, JITT controls delay and jitter over whole communication path. We evaluated JITT on the different simulations and one experimental testbed for performance analysis. Our findings support that JITT provides stable delay and low jitter and transports the packets nearly just in time.

Keywords: Jitter, Packet Scheduling, QoS, Delay, Real-Time Communication.

1. Introduction

Real-time communication technologies provide many benefits in a broad area. The main areas of real-time communication are personal communication, distance learning, IP TV and telemedicine. In the personal communication area, Voice over IP (VoIP), video conferencing systems and live video broadcasts are becoming increasingly common due to low costs. Similarly, distance learning is popularized due to its advantage of place independence. Another emergent area is Telemedicine. Especially, telesurgery needs very high Quality of Service (QoS) and low error rate. Beside, some other developing areas are real-time multiplayer games, real-time remote monitoring and control systems. If the data of the above applications are transported on a converged network, some problems arise.

At this point, it is useful to distinguish between QoS and Quality of Experience (QoE). QoS is a more technical definition. It defines that the services provided by the service provider should be delivered to the user without error. QoE is overall user experience [1]. In particular, the widespread use of mobile and wireless networks has made these two concepts even more important. For example, an IPTV provider must provide users with a certian QoE in their wireless environment [2]. Problems such as users being mobile, bandwidth fluctuations in wireless access, lack of connectivity are making real-time video streaming difficult. Different algorithms and technologies are being developed for this purpose.

In packet-switched networks, produced data by many users are queued into buffers on the network devices and serviced one by one. This situation often differentiates the arrival times of the packets of the same flow to their destinations. This delay difference is called as jitter. Lower latency and jitter, which means a higher quality of service. Delay and jitter are the most critical components of real-time communication. More technically, delay is the elapsed time between the start of a packet's sending time and the end of its receiving time. From the end user perspective, jitter can be described as unexpected interruption during playback [3]. For reducing delay and jitter, bandwidth reservation and packet prioritization techniques are used. However, prioritization of real-time packets means that the packets belonging to other services (e.g. HTTP, FTP) will experience more latency. The critical issue at this point is to prioritize real-time packets as much as their needs. For example, if a real-time data needs to reach the receiver in 100 ms, it does not make sense to send it to the receiver in 50 ms at the expense of lowering the quality of the other services such as HTTP and FTP. Here, we must establish a balance between services. This is possible only when

real-time data is delivered in a timely manner, namely just in time. In this study, we developed a scheduling/prioritizing algorithm and a packet header extension to reduce delay and jitter.

The remainder of this paper is organized as follows: In Section 2, the related works are presented. JITT algorithm details are described in Section 3. Section 4 gives materials and methods which used in simulations and experiments. Simulation and experimental results are explained in Section 5. Various important points to be considered are discussed in Sections 6.

2. Related Work

QOS improvement methods are categorized as Integrated Services (Intserv) [4] and Differantiated Services (Diffserv) [5]. The idea of Intserv is that each application has to make bandwidth reservation. By contrast, Diffserv does not reserve network resources. Diffserv provides priority for higher classes of service. Intserv is not useful on the internet due to lack of scalability. Diffserv based methods has been well accepted.

Packet by packet scheduling discipline first introduced in [6] and called Weighted Fair Queuing (WFQ). WFQ classifies data flows and reserves resources for these classes. Generalized Processor Sharing (GPS) implements same method with WFQ [7]. GPS is a strategy for rate-based flow control and employs admission control for guaranteeing throughput and delay in the worst-case. Another packet by packet scheduling approach is Stochastic Fair Queuing (SFQ) [8]. SFQ uses hashing to map packets to corresponding queues. Normally, every possible flow needs its own queue, but SFQ offers less number of queues from possible number of flows. So, more than one flow can fall into the same queue and the flows' fairness become stochastic. Similarly, Deficit Round Robin (DRR) algorithm uses stochastic fair queuing to assign flows to queues [9]. DRR's one difference from round robin algorithm is that if a queue was not able to send a packet in the previous round because of its packet size was too large, the previous quantum is added to the next round. Another round robin based study is SATURN [10]. SATURN uses simple dual round robin arbitration scheme to schedule packets. Many queue models shapes traffic to achieve fair share. But Traffic Shaping Algorithm with Delay Jitter Constraints (TSJC) algorithm proposed in [11] shapes traffic with delay jitter constraints. TSJC dynamically configures traffic shaping parameters on the basis of calculating packet delay, jitter and loss in a queue buffer.

At this point it would be useful to talk about playout buffering. Playout buffering provides smooth jitter and gains time for resending lost packets, but introduces additional buffering latency to playback. As an example of this technique, MultiLayer-AudioVisual Streaming System (ML-AVSS) buffers the packets at receiver side and transmits them to application layer [12]. Generally, small start-up delay before playback is acceptable for end users. However, they are much less tolerable towards halt or interruption during playback [3].

Many of the scheduling algorithms in the literature are round robin (RR) based and these algorithms work on single network device. These methods operate on each device independently from the other devices. Therefore they cannot control end-to-end delay and jitter. Another alternative method to achieve end-to-end delay control is reservation based

methods. However, bandwidth reservation for each data stream is not feasible on a great environment involving tens of thousands users. Also, all the previous bandwidth reservation studies are focused on specifying an upper bound of Real-time Multimedia (RTMM) packet's delay. Thus, quality of RTMM flow can be increased. However, the quality of other services must also be considered. For this purpose, RTMM packets must be delivered neither early, nor late. This is possible with just in time transportation of RTMM packets.

Additionally, just-in-time communication requires high service utilization on the one hand and short service response time on the other [13]. JITT tries to overcome these two problems.

Our contributions are summarized as follows:

- We introduce a new research question: how to transport IP packets just in time?
- We propose a new packet scheduling/prioritization method to deliver RTMM packets nearly just in time.
- We conduct comprehensive simulations and experiments to validate our analytical results and evaluate the performance of JITT.

3. The JITT Algorithm

In this section we describe JITT's behaviors in details.

3. 1 JITT Structure

JITT considers all router queues on a whole communication path as a single queue and schedules (RTMM) packets. For this purpose, JITT adds 16 bit Desired Maximum End-to-End Delay (DMEED) field to multimedia data and encapsulates it in UDP packets in order to achieve as shown in Fig. 1. Herein, we may ask that how are applications incentivized to choose proper delay targets? Following the related standards will be a solution. For example, The ITU G.114 specification recommends less than 150 millisecond (ms) one-way end-to-end delay for high-quality real-time traffic such as voice [14]. In ITU G.114, the need for the delay value to be less than 150 ms is explained as follows: "For many intra-regional (e.g., within Africa, Europe, North America) routes in the range of 5000 km or less, users of VoIP connections are likely to experience mouth-to-ear delays <150 ms." and "If delays were kept below 150 ms, then most applications would not be significantly affected."

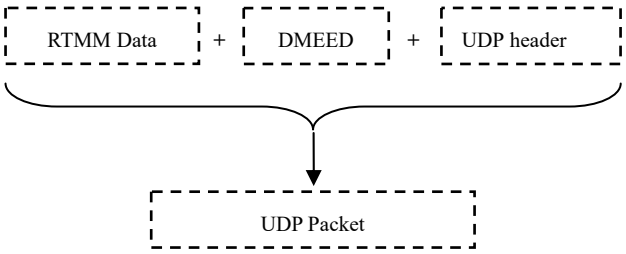

Fig. 1: Data encapsulation

A router on the communication path calculates the packet's Maximum Waiting Time (MWT) using DMEED. It sends the packet when MWT is finished. In this study, we added DMEED as a new field to UDP header in the simulations. Thus, we created a new packet structure. In order to differentiate these packets from ordinary UDP packets, we used a different number from UDP's assigned number in IP header's protocol field. An IP header option field can be used for DMEED. In this case, it would be enough to reach the IP header. Additionally, all routers can reach the IP header while they cannot reach UDP headers. In this study we added a new field to UDP header because of there is no meaningful difference between usage of an IP header or UDP header field in a simulation environment.

JITT uses 2 queues on each router: the JITT queue and the other queue. RTMM packets are buffered in the JITT queue and all other packets are buffered in the other queue.

3. 2 JITT Working Principle

When two users start an RTMM communication, the sender puts the desired maximum end-to-end delay value in the packet's DMEED field. When a router on the path receives this packet, it calculates MWT for the packet; see (1).

$$\mathrm{MWT} = \frac{\mathrm{DMEED}}{\mathrm{Hopcount}} \tag{1}$$

In (1), MWT is maximum waiting time, DMEED is DMEED field of the packet and Hopcount is the number of the routers which are located on the path (sender's subnet to receiver's subnet). If router runs a distance vector routing protocol, it can get Hopcount value from the routing table. Otherwise, it sends a query to the receiver. Once the Hopcount is calculated for a receiver's subnet, the value can be stored in a table and reused. In this study we used distance vector based approach. After the calculation of MWT, the router creates a structure using the packet, MWT and a local timestamp as shown in Fig. 2 and adds it to the JITT queue. The router adds all received other packets to the other queue simultaneously. As a result, router keeps two queues. In parallel with packet adding processes described above, another process searches the JITT queue continuously. The searching process calculates that how long a packet waited in the JITT queue until now; see (2).

Fig. 2: The structure added to JITT queue

$$\mathrm{ET} = \mathrm{CT} - \mathrm{TIMESTAMP} \tag{2}$$

(2), ET is elapsed time, CT is current time. After the calculating of ET, router determines the necessity of sending a packet (see Algorithm 1). ServiceTime parameter in the Algorithm 1 is the time needed to send the packet over the line and it is calculated as follows: packet in bits/bandwidth.

Algorithm 1. Determination of Sending a Packet

```
If ET >= MWT–ServiceTime do
        DMEED = DMEED–ET–ServiceTime; //Update DMEED Field on JITT packet
        Send the JITT packet; //packet waited MWT
Else if   other queue != Empty do
        Send the first packet in the other queue
Else Send the first packet in the JITT queue
```

We can explain Algorithm 1 as follow:

1. If there are no packets in the other queue, send JITT packets as soon as possible

2. If at least one packet is present in the other queue, send this packet as soon as possible; hold JITT packets till as late as possible. "As late as possible" means that DMEED value on the JITT packets.

If router decides to send a JITT packet, router updates DMEED field of the packet (see Eq. 3). Thus, the next router on the path can calculate that how long this packet can wait in the queue. So, DMEED value is decreased on each router and each router can calculate its own MWT value for a particular packet.

$$\mathrm{DMEED} = \mathrm{DMEED} - \mathrm{ET} - \mathrm{ServiceTime} \tag{3}$$

Some studies uses similar technique we implemented in JITT. One of the studies is adaptive per hop differentiation (APHD) [15]. In APHD, data packets carry end-to-end delay requirement just like JITT. In APHD the distance from the source to the destination is calculated at the source node. Hence, APHD needs source routing which is not allowed on the internet. Another of the studies is EstServ [16]. In the EstServ, packets carry their deadlines. But, EstServ has some drawbacks. In the EstServ, the source decides the hop count. Hence, EstServ needs source routing just like APHD. Also, EstServ needs a synchronized network to calculate a packet's deadline. Synchronization of the all routers on the internet is not feasible.

3. 3 Waiting Time Distributions of RTMM Packets

We can say that there are two waiting times in a queue for a packet. These are queuing time W_q and service time t. Total waiting time can be expressed as

$$W = W_q + t \tag{4}$$

W_q can be calculated as

$$W_q = t_2 + \cdots t_n \tag{5}$$

W can be calculated as

$$W = t_1' + t_2 + \cdots t_n \tag{6}$$

where t_1' is the service time of the last received packet and $t_2, \ldots, t_n$ are service times of $(n-1)$ packets in the queue. In telecommunication networks, packets arrive at a router

according to Poisson distribution [17]. Let φ be packet size in bits, B_{out} be router's outgoing link bandwidth. Service time for nth packet can be calculated as

$$t_n = \frac{\varphi_n}{B_{out}} \tag{7}$$

Keep the queue length at a certain level over the time is impossible. So, W_q for each packet will be different. Thus, each packet experience different delays and jitters over the time. JITT determines packet's delay bound using DMEED. So, W for RTMM packets in a flow is same. Let τ be MWT. JITT calculates sending time of a packet as

$$\tau - \frac{\varphi}{B_{out}} \geq \Delta \tag{8}$$

Where Δ is elapsed time of the packet in the queue. So, JITT takes into account packet service time and brings τ closer to Δ. Let $F = \{P_1, P_2 \dots P_n\}$ be a RTMM flow, where n is total packet number and $P_i, 1 \leq i \leq n$ are packets. W_q for F can be expressed as

$$F_{W_q} = \left\{\left(\tau - \frac{\varphi_1}{B_{out}}\right), \left(\tau - \frac{\varphi_2}{B_{out}}\right), \dots, \left(\tau - \frac{\varphi_n}{B_{out}}\right)\right\} \tag{9}$$

Service time for F can be expressed as

$$F_{service} = \left\{\left(\frac{\varphi_1}{B_{out}}\right), \left(\frac{\varphi_2}{B_{out}}\right), \dots, \left(\frac{\varphi_n}{B_{out}}\right)\right\} \tag{10}$$

Finally we write

$$F_{W_q} + F_{service} \cong \{(\tau), (\tau), \dots, (\tau)\} \tag{11}$$

So, waiting time in the queue for all packets in the same flow is equal. Technically, jitter has been described as "variation of a metric (e.g., delay) with respect to some reference metric (e.g., average delay or minimum delay). This meaning is frequently used by computer scientists and frequently (but not always) refers to variation in delay." in [18]. We used average delay as reference metric in calculation of jitter. So, jitter of a flow F can be expressed as

$$F_J = \{\tau_1 - \tau_{avg}, \tau_2 - \tau_{avg}, \cdots, \tau_n - \tau_{avg}\} \tag{12}$$

where $\tau_{avg} \cong \frac{\sum_1^n \tau_n}{n}$.

Because of every packet's τ value is same in a JITT flow F, we can write $\tau_{avg} \cong \tau_i, 1 \leq i \leq n$. Thus we get (13).

$$F_J \cong \{\tau_1 - \tau_{avg}, \tau_2 - \tau_{avg}, \cdots, \tau_n - \tau_{avg}\} \cong \{0,0, \dots, 0\} \tag{13}$$

(13) shows that JITT keeps jitter close to zero.

3. 4 Departure Time Analysis of RTMM Packets

In this section we evaluated effects of packet sizes, MWT values, input and output bandwidths of a router and other traffic loads on JITT's performance.

Let P_1 and P_2 be two packets in a JITT queue where $P_1 \prec P_2$. We assumed that there is no time interval between the receiving time of P_1's last bit and receiving time of P_2's first bit from the line. Let queuing times P_1 and P_2 be t_1 and t_2; MWTs be τ_1 and τ_2; service times be η_1 and η_2; dequeuing times be t_1' and t_2' respectively. We write differences of the receiving times of the packets as

$$\delta_1 = t_2 - t_1 = \frac{\varphi_2}{B_{in}} \tag{14}$$

We write differences of sending times of the packets as

$$\delta_2 = (t_2 + \tau_2) - (t_1 + \tau_1) \tag{15}$$

and differences of taking times of the packets from the queue as

$$\omega = t_2' - t_1' \tag{16}$$

Where B_{in} is the input bandwidth of the router. We showed the above equations in Fig. 3. We can say that ω depends on φ_1, φ_2, B_{in}, B_{out}, τ_1 and τ_2. In the worst case with the provision of some pre-conditions, it can be $t_2' = t_1'$. This means that sending time of P_2 shifts amount of η_1. Mentioned worst case may be realized only where there is no intervals between consecutively received RTMM packets from the line, $B_{out} < B_{in}$ and $\varphi_2 > \varphi_1$. On the Internet, all incoming traffics won't be RTMM. If RTMM traffic load is less than B_{in}, there will be intervals between RTMM packets. Also, if there are other traffic types on the line, there will be intervals between RTMM packets, too. So, the worst case that mentioned above is unlikely occurred. Even if all the conditions are fulfilled, shifting of sending time of P_2 inversely correlated with B_{out}. Usually, average RTMM packet size is not very large. Thus, shifting time will be smaller. So, we can ignore this shifting (η_1).

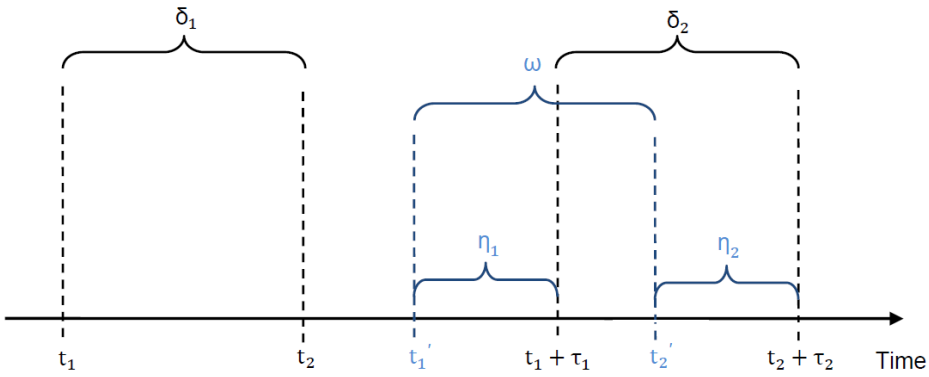

Fig. 3: Departure time analysis of JITT packets

3. 5 Determination of JITT Queue Limit

Determining the JITT queue limit is a crucial point for proper operation of the model. If we set JITT queue limit lower than a certain level, the packets may be dropped while it is possible to deliver to the target. Let us analyze this situation.

Let $Q = \{P_1, P_2, \cdots, P_n\}$ be a JITT queue, where $P_i, 1 \leq i \leq n$ are packets in the queue. If τ_i is MWT of P_i, we pick P_{max} which has $\tau_{max} = \max_{1 \leq i \leq n}\{\tau_i\}$. So we can create a subset of Q as $\Phi = Q \backslash \{P_{max}\}$. Let $\forall P_i \in \Phi$, φ_i be packet size in bits. We calculate sum of the packet sizes in Φ as $\Theta = \sum_{i=1}^{n-1} \varphi_i$. If we set queue limit to Θ and if $\Theta / B_{out} < \tau_{max}$ fulfilled, the router rejects $(\tau_{max} - \Theta / B_{out}) \times B_{out}$ amount of the data packet while it is possible to deliver to the target in just in time. If $\Theta / B_{out} = \tau_{max}$ fulfilled, just in time transport is possible and we use B_{out} efficiently. If $\Theta / B_{out} > \tau_{max}$ fulfilled, there will be additional latencies for some packets. As mentioned before, JITT considers all router queues on a whole communication path as a single queue. Additional latencies on a router caused by traffic load can be compensated by the other routers on the communication path. So, we should set JITT queue limit to at least $\{\tau_{max} \times B_{out}\}/8$ bytes.

3. 6 Network of JITT Queues

As we showed in the previous section, JITT can compensate packet latencies on a router caused by traffic load. As a simple example, let there be 2 routers on a path and an RTMM packet's DMEED field is 100 ms. We assumed that the packet waits 50 ms on each router. If the first router sends the packet in 30 ms due to low traffic load, there will be additional 20 ms waiting right (MWT) for this packet on the second router. Namely, the second router can hold this packet in its queue until 70 ms (50 ms+20 ms). Oppositely, waiting right of the packet will be less than 50 ms. We give an example to clarify this situation. Let's look at the last received packet to the router 1 in Fig. 4. It's clear that router 1 cannot send the packet in τ time. Let φ be size of this packet in bits, Θ_1 be total size of all packets at router 1 in bits, Θ_2 be total size of all packets at router 2 in bits, σ_2 be empty place in just in time threshold on router 2 queue, λ_2 be input traffic load of router 2, μ_2 be output bandwidth of router 2 and Θ_1/μ_1 is the necessary time for leaving of the packet from router 1 and reaching to router 2. If Eq. 17 is fulfilled for the packet, the packet can reach to the destination in DMEED time.

$$\Theta_2 + \varphi + \left(\frac{\Theta_1}{\mu_1}\right)(\lambda_2 - \mu_2) \leq \sigma_2 \tag{17}$$

If we generalize Eq. 17 for a communication path which has n routers, we obtain Eq. 18.

$$\sum_{j=2}^{n}\left[\Theta_j + \varphi + \frac{\Theta_{j-1}}{\mu_{j-1}}(\lambda_j - \mu_j)\right] \leq \sum_{j=2}^{n}[\sigma_j] \tag{18}$$

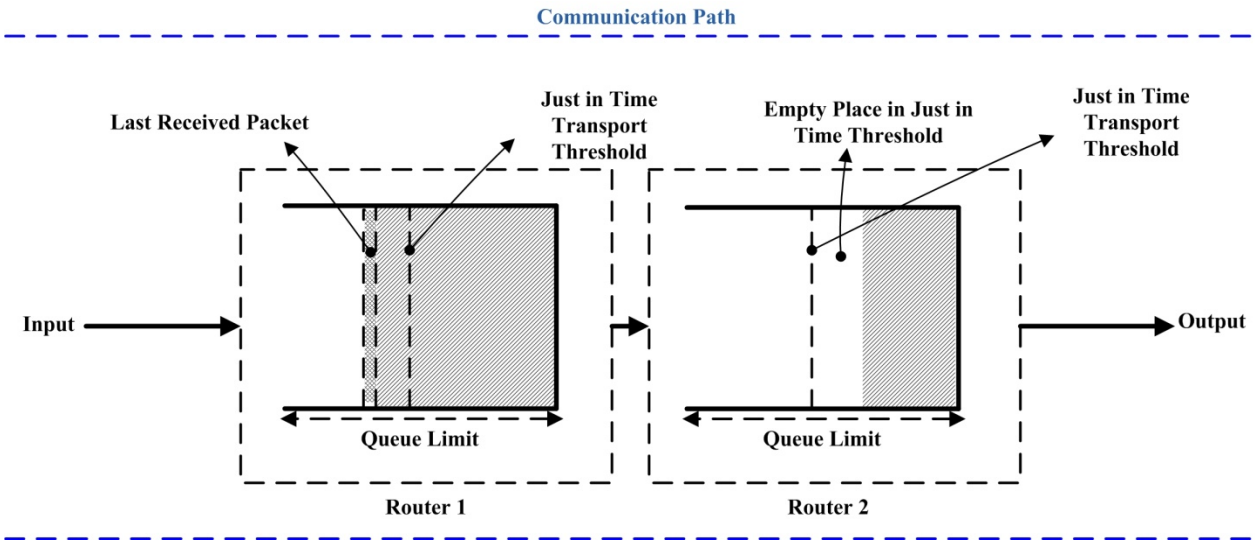

Fig. 4: A communication path with two routers

3. 7 Link Schedulers and JITT

Packet schedulers are divided into two categories: General Schedulers (GS) and Link Schedulers (LS) [19]. GSs work on different queues which hold classified traffics. LSs share bandwidth between classes in the case of congestion. JITT acts as both GS and LS. We think that we use LS with JITT. When JITT scheduler obtains the order from LS to transmit a packet, if there is a packet that must be sent in the JITT queue; JITT transmits it. Otherwise, JITT transmits another packet from the other queue. Thus, sending order is transferred to the other queue. In this study, we used LS with JITT.

3. 8 Hop by Hop Routing and JITT

Source routing is not allowed on the Internet. A router may forwards the packets to the same target over different network interfaces. So, the packets may reach to the same destination over different paths. These paths may contain different number of routers. This situation may at first seem like a problem. However, JITT overcomes this problem thanks to its hop-by-hop working principle. MWT of a particular RTMM packet is calculated on the routers separately; not on the source. We showed an example for this situation in Fig. 5. There are 2 paths to the destination in the Fig. 5. Path 1 has 4 routers and path 2 has 5 routers. The source puts desired maximum end to end delay value (in to DMEED field) to the packets. MWT value of the packets which forwarded on the path 1 is 50 ms, MWT value of the packets which forwarded on the path 2 is 40 ms. As a result, the packets reaches to the destination after 200 ms through both paths. Arrival time of the consecutive packets does not change even if packets are forwarded on different paths.

If we talk about the protocol overhead of JITT, JITT adds only 2 bytes DMEED filed to the packets. Namely, protocol overhead of JITT is 2 bytes for all packets.

Additionally, we must consider IP packet fragmentation. If a router fragments IP packets, we may not obtain desired maximum end-to-end delay bound. If possible, we must keep the size of IP packets smaller than the communication medium's maximum transfer unit (MTU).

Lastly, if a packet experiences larger delay than DMEED value, DMEED will be negative at the receiver. This situation does not affect JITT algorithm. However, it is an indicator of misadjustment of initial DMEED value or inefficiency of the bandwidth. If one of the cases mentioned above occurs, the RTMM receiver detects continuously negative values in the DMEED. As a future work, we can develop a DMEED feedback mechanism. Thus, the receiver can notify the sender about the status of the connection. Sot, the sender can re-adjust DMEED value or data rate.

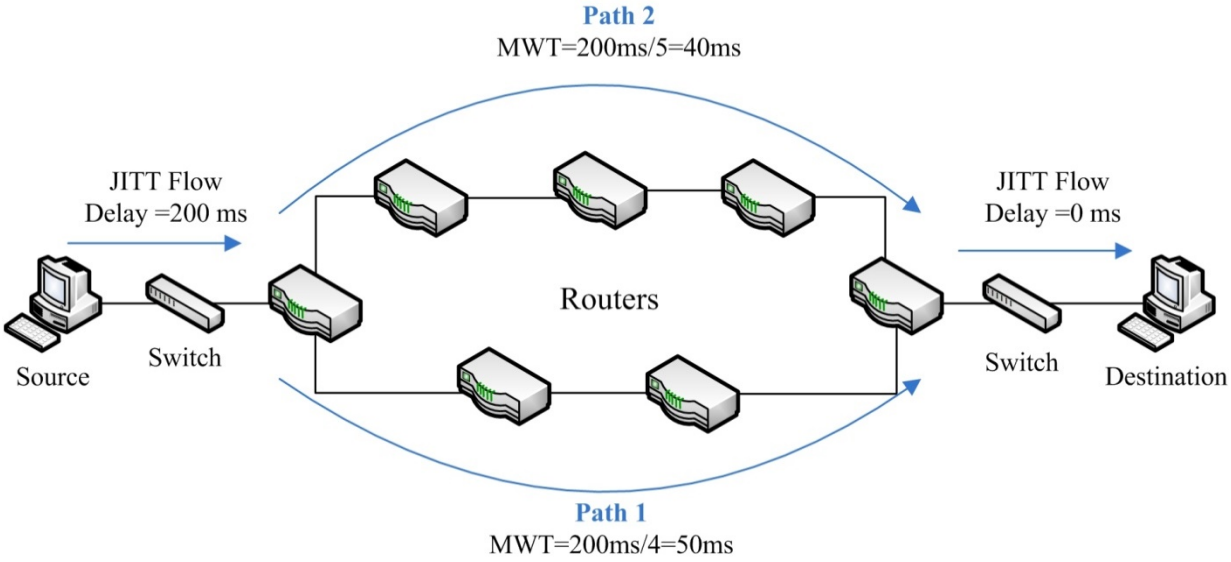

Fig. 5: JITT behavior in hop by hop routing

4. Materials and Methods

In this study, we choose delay, maximum jitter, packet loss ratio and bandwidth utilization as evaluation criteria. There are numerous packet scheduling algorithms in the literature. These algorithms have been implemented on different operating systems or simulators and their performances have been compared with different packet scheduling algorithms. Furthermore, some of the studies are specific for wireless networks. Therefore, deciding the best packet scheduling algorithm is a very difficult task. Also, this topic is out of scope of this study. Therefore, we compared JITT's performance with FIFO (non-QoS support), ordinary Round Robin (RR - we can say that RR is the simplest DiffServ method) and Ideal Situation (IS - zero delay, zero jitter, full bandwidth utilization).

Delay and jitter are the most critical components of RTMM communication. We gave description of jitter in previous sections. In the multimedia applications, calculation of playout buffer is done according to maximum jitter [20]. A playout buffer is a holding area for packets which will be played in the future. Therefore, the accumulation of a certain amount of packets is expected in the buffer before the packets' contents start playing. The time required for the accumulation of packets adds a virtual delay. If the playout buffer is too large, it adds an artificial delay to communication. Thus, we must keep jitter as minimum as possible.

In the jitter tables, we assumed that every experiment's average delay equals to 1. Namely, 0.6 jitter value shows that the deviation from average delay is 60%. Lastly, higher packet loss ratio negatively affects quality of RTMM communications as well as quality of other traffic types. Therefore, we measured RTMM traffics and Background Traffics (TCP traffics) packet loss ratios. Measurements on intercontinental links show that 90-95% of the bytes belonging to TCP traffic [21]. Therefore, we analyzed impact of JITT and the other methods to Background traffic.

Theoretically, JITT can work on the whole internet but it may be unnecessary to control every possible RTMM flows. Thus, JITT deployments will be more useful on relatively small networks such as campus networks, local campus-to-remote campus networks and autonomous systems. Hence, we created a small simulation and experimental testbed which have a few routers. Also, after the detailed mathematical analysis of JITT, we kept the simulation and real-world experiments simple.

4. 1 Simulation Setup

We tested JITT with ns2 simulator [22] on 3 different simulation scenarios and compared with FIFO, RR and Ideal Situation (IS). The essential of simulating of multimedia data transmission is simulating more realistic multimedia data source. Domoxoudis et al. have obtained traffic patterns of 7 Variable Bit Rate (VBR) encoders with 1 hour real environment experiment [23]. With follow up this study, we used 327.6 Kbps for video and 47.2 Kbps for audio data source. We created VBR traffic over UDP. We used FTP over TCP as background traffic. In the simulations we set JITT queue size and the other queue size to 25 packets (each). In FIFO and RR simulations we set queue sizes to 50 packets. Thus, we used same queue sizes in all simulations. We ran all simulations 10 times and used average values for comparison. When the simulation starts, VBR immediately begins to generate packets. At the same time, FTP begins to transfer data and TCP uses the bandwidth effectively in a short time (FTP runs over TCP). Thus, we chose shorter simulation times because of there is no meaningful changes in traffic loads during the simulations. Additionally, we have made experiments for different packet sizes in the simulation environment. But we did not include the results in this study because there were no significant changes in the results.

All simulation topologies are dumbbell topology and they have same router numbers but different communicating host numbers. First two simulation topologies have 2 hosts on the left side and 2 hosts on the right side of the routers. The third simulation topology has 20 hosts on the left side and 20 hosts on the right side of the routers. General simulation topology is shown in Fig. 6.

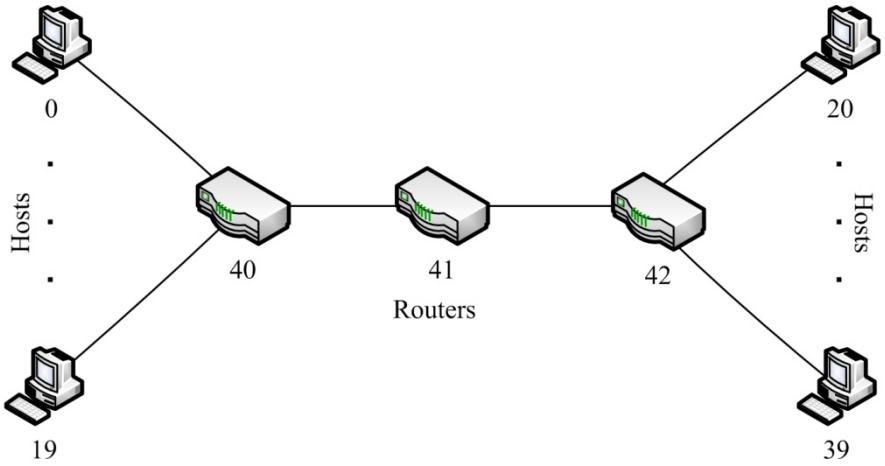

Fig. 6: General simulation topology

In the first simulation, inter-routers link capacity is 0.5 Mbps and latency is 10 ms. We created 1 mutual RTMM connection between 2 nodes and 1 mutual Background Traffic connection between 2 other nodes. Simulation time is 65 s. In the second simulation all parameters are same with the first simulation, except Background Traffic connections. We created 2 mutual background connections between 4 nodes. In the third simulation inter-routers link capacity is 2 Mbps and latency is 10 ms. We created 2 mutual RTMM connections between 4 nodes and 18 one-way Background Traffic connections between 36 other nodes. Simulation time is 110 s. We conducted different simulation scenarios to evaluate JITT performance on different traffic loads and link capacities.

4. 2 Experimental Setup

We prepared a testbed in order to evaluate JITT performance in application and compare with FIFO, RR and Ideal Situation (IS).

The testbed topology is shown in Fig. 7. The routers are shown in Fig. 7 are desktop computers. We developed routing software with WinPcap packet capture libraries. In the experiments, we used 598s part of Planet Earth Deserts Documentary as RTMM flow source. Its video component is encoded with H.264; the resolution is 640x480 pixels; bitrate is 533Kbps; and audio component has 2 channels; bitrate is 132Kbps. Inter-routers link capacity is 2.5 Mbps. The experiments carried out with 2 different background traffic loads: 1 Mbps and 2 Mbps. We chose Constant Bit Rate (CBR) background traffic loads different from the simulations (in the simulations we have used TCP traffic sources as background traffic).

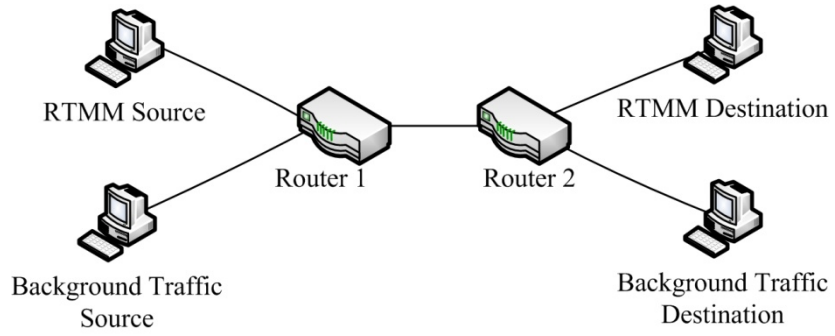

Fig. 7: Testbed topology

5. Results

In this section we presented simulation and experimental results.

5. 1 Simulation Results

The results of the simulation scenarios are shown in the Table 1, Table 2, Table 3 and Table 4. The best results are marked with bold style in the tables. As we have shown before mathematically, JITT keeps the jitters at a minimum level. Also, JITT didn't drop any multimedia packet. The performance of JITT and performance of other methods for Background packet loss ratio and Bandwidth utilization parameters are close to each other.

Table 1. Maximum jitter

Maximum jitter	**Scenarios**		
	Scenario 1	***Scenario 2***	***Scenario 3***
JITT	0.0495	0.0232	0.0516
FIFO	0.5818	0.6153	0.1470
RR	0.1129	0.0685	0.0519
IS	0	0	0

Table 2. Multimedia packet loss ratio (%)

Multimedia packet loss ratio (%)	**Scenarios**		
	Scenario 1	***Scenario 2***	***Scenario 3***
JITT	0	0	0
FIFO	1.51	0.09	3.70
RR	2.39	0.43	0.86

Table 3. Background packet loss ratio (%)

Background packet loss ratio (%)	**Scenarios**		
	Scenario 1	***Scenario 2***	***Scenario 3***
JITT	0	0	5.95
FIFO	2.83	0.11	9.72
RR	12.51	1.44	17.42

Table 4. Bandwidth utilization (%)

Bandwidth utilization (%)	Scenarios		
	Scenario 1	*Scenario 2*	*Scenario 3*
JITT	99.30	99.30	97.85
FIFO	97.47	99.20	94.16
RR	94.46	98,62	94.41
IS	100	100	100

5. 2 Experimental Results

The results of the experiments are shown in the Table 5, Table 6, Table 7 and Table 8. We calculated bandwidth utilization as $(\text{total throughput}/\text{link bandwidth}) \times 100$. In the scenario 1, total traffic load (so, total throughput) does not reach to available link bandwidth. Thus, IS's bandwidth utilization rate is less than 100%. The best results are marked with bold style in the tables. As we had expected, JITT gave best performance for Maximum jitter and Multimedia packet loss ratio parameters. The performance of JITT and performance of other methods for Background packet loss ratio and Bandwidth utilization parameters are close to each other.

Table 5. Maximum jitter

Maximum jitter	Scenarios	
	Scenario 1	*Scenario 2*
JITT	0.8752	0.7625
FIFO	3.2534	1.0953
RR	2.3057	0.7975
IS	0	0

Table 6. Multimedia packet loss ratio (%)

Multimedia packet loss ratio (%)	Scenarios	
	Scenario 1	*Scenario 2*
JITT	0	0
FIFO	0	58.70
RR	0	28.55

Table 7. Background packet loss ratio (%)

Background packet loss ratio (%)	Scenarios	
	Scenario 1	*Scenario 2*
JITT	0	10.40
FIFO	0	9.39
RR	0	9.62

Table 8. Bandwidth Utilization (%)

Bandwidth Utilization (%)	Scenarios	
	Scenario 1	*Scenario 2*
JITT	66.60	98,28
FIFO	66.60	83.44
RR	66.60	91.28
IS	66.60	100

We can see that from Table 5, JITT has kept end-to-end delay stable. Because the maximum jitter is maximum deviation from the end-to-end delay over the time. Also, Fig. 8 and Fig. 9 can be examined to see the end-to-end delay values obtained from the experiments. As shown in Fig. 8 and Fig. 9, JITT provides stable end-to-end delay. Low end-to-end delay oscillation is a sign of low jitter. Low end-to-end delay across all communication means that higher quality of service is provided. In this respect, Fig. 8 and Fig. 9, provides significant information on the performances of the methods. Again, the zoomed in regions presented in the figures provide detailed information on how the methods are stabilizing the end-to-end delay. It can be seen from these small figures that JITT is more stable in end-to-end delay value.

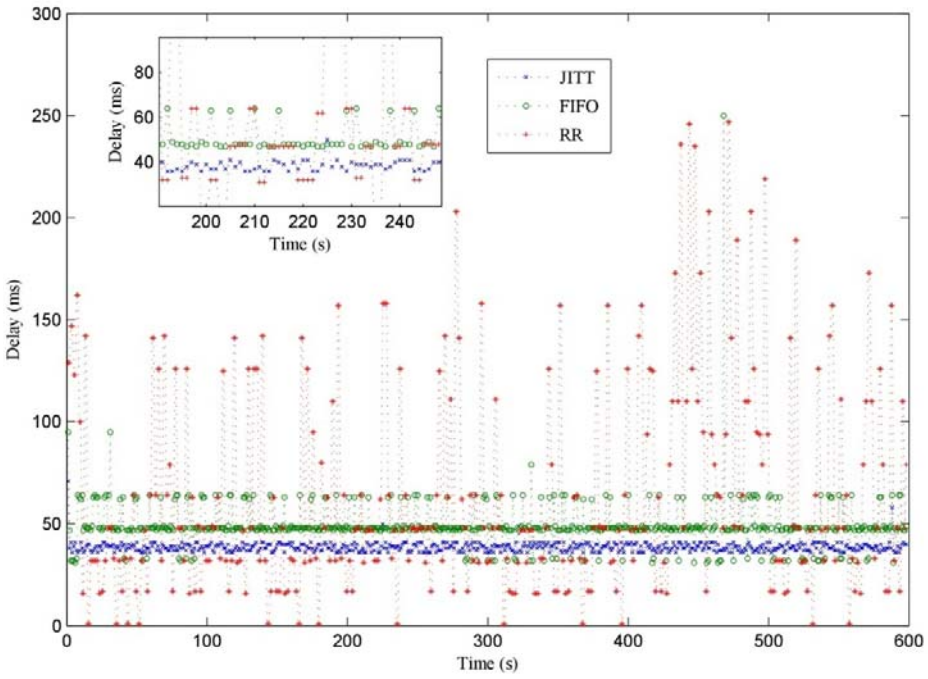

Fig. 8: Measured end-to-end delays in the first experiment

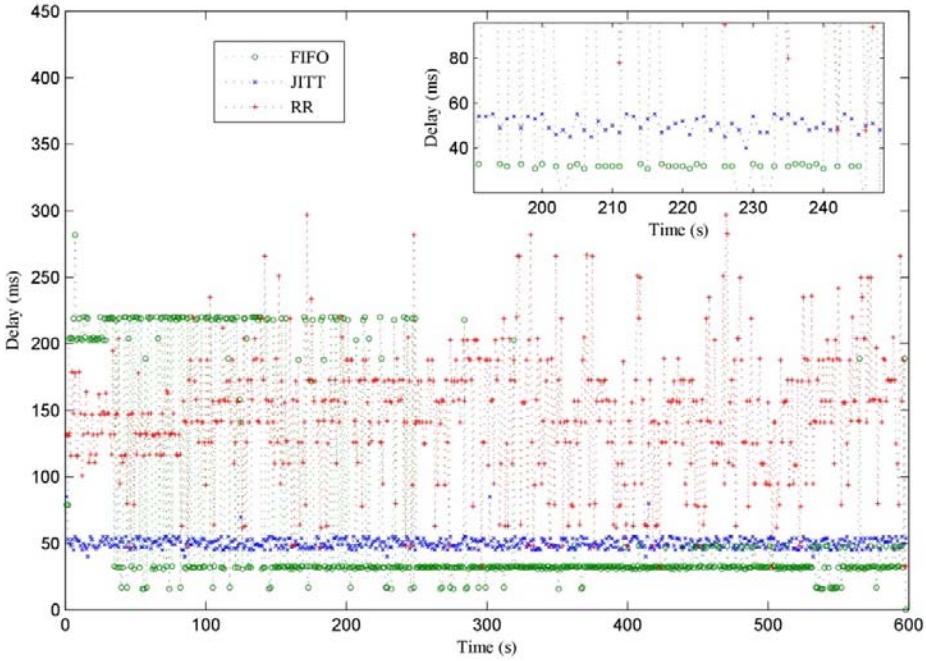

Fig. 9: Measured end-to-end delays in the second experiment

6. Discussion and Conclusions

As shown in the Results section, JITT provides significant improvement in jitter, delay and packet loss ratios. The challenge here is to do it without affecting other traffic classes. According to the simulation results and real experiments, JITT has little effect on the other traffic classes. In contrast, other methods that are compared greatly affect the other traffic classes. This is an undesirable situation.

Large delay and jitter are two important factors that disrupt traffic RTMM. To improve these properties, we proposed the JITT scheduling/prioritizing algorithm and a packet header extension, which guarantees the delay and jitter bounds for RTMM traffics. We evaluated JITT's performance with simulations in ns2 and experiments on the prepared testbed. Our findings support that JITT provides stable delay and lower jitter. In addition, JITT provides high bandwidth utilization. On the one hand, traditional approaches reduce delay and jitter; but on the other hand they decrease the quality of other traffic classes. The purpose of our development of JITT is to achieve a balance between the QoS received by different traffic classes.

The current RTMM communication applications uses playout buffers for absorbing jitter. However, playout buffering increases initial playback time. The larger playout buffer reduces QoS of communication in terms of delay. As mentioned before, calculation of playout buffer capacity is done according to maximum jitter. JITT provides stable delay and low jitter, so, smaller playout buffers can be used with JITT.

JITT does not need source routing to compute packet's deadlines. Also, it uses all router queues as a single queue unlike existing scheduling algorithms. This makes JITT usable in

larger networks (wired or wireless). We categorize the existing packet schedulers into LS or GS. JITT acts as both GS and LS unlike existing scheduling algorithms. JITT schedules RTMM packets; on the other hand, it shares bandwidth between classified RTMM traffic and best effort traffic. Additionally, JITT can be used in Device-to-Device communications thanks to its hop-by-hop working principle. Also, JITT can be easily integrated into the existing technologies due to its simple logic and low protocol overhead.

In this study, we used software based routers in the testbed. Therefore, delays caused by the operating system are reflected in the measurements. To conduct more healthy experiment, hardware design is required. In addition, RTMM throughput has decreased at heavy RTMM traffic load (~100 Mbps) in the experiments. The underlying reason is that the software based routers cannot run JITT algorithm enough faster at high data rates. In order to faster routing, JITT algorithm should be implemented on the hardware. In recent years, with the development of mobile technologies, the increase and acceleration of wireless internet access, there has been a great increase in real-time multimedia applications. Many messaging and social networking applications offer features like video calls and live broadcasts. In this context, the importance of RTMM communication is further increased. The methods developed in this study can be used in wireless mesh networks. wireless mesh networks are more static. The routers that make up the network are controlled by the same authority. Therefore, it may be possible to use JITT on all routers for end-to-end QoS control. Similarly, JITT can also be used in Ad Hoc networks. However, due to the rapidly changing nature of Ad Hoc networks, packets may experience excessive overhead on devices. This is a problem that needs to be resolved.

References

[1]M. Garcia, A. Canovas, M. Edo, and J. Lloret, “A QoE Management System for Ubiquitous IPTV Devices,” in *2009 Third International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, 2009, pp. 147–152. doi: https://doi.org/10.1109/UBICOMM.2009.31

[2]M. Garcia, J. Lloret, M. Edo, and R. Lacuesta, “IPTV distribution network access system using WiMAX and WLAN technologies,” in *Proceedings of the 4th edition of the UPGRADE-CN workshop on Use of P2P, GRID and agents for the development of content networks - UPGRADE-CN ’09*, 2009, p. 35. https://doi.org/10.1145/1552486.1552513

[3]X. Lou and K. Hwang, “Quality of data delivery in peer-to-peer video streaming,” *ACM Trans. Multimed. Comput. Commun. Appl.*, vol. 8s, no. 1, pp. 1–23, Feb. 2012. https://doi.org/10.1145/2089085.2089089

[4]R. Braden, D. Clark, and S. Shenker, “Integrated Services in the Internet Architecture: an Overview”, RFC, Jun. 1994.

[5]D. Grossman, “New Terminology and Clarifications for Diffserv,” RFC 3260. Apr. 2002. http://www.rfc-base.org/rfc-3260.html

[6]A. Demers, S. Keshav, and S. Shenker, “Analysis and simulation of a fair queueing algorithm,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 19, no. 4, pp. 1–12, Aug. 1989.

https://doi.org/10.1145/75247.75248

[7] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," *IEEE/ACM Trans. Netw.*, vol. 1, no. 3, pp. 344–357, Jun. 1993. https://doi.org/10.1109/90.234856

[8] P. E. McKenney, "Stochastic fairness queueing," in *Proceedings. IEEE INFOCOM '90: Ninth Annual Joint Conference of the IEEE Computer and Communications Societies@m_The Multiple Facets of Integration*, 1990. pp. 733–740. https://doi.org/10.1109/INFCOM.1990.91316

[9] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," *IEEE/ACM Trans. Netw.*, vol. 4, no. 3, pp. 375–385, Jun. 1996. https://doi.org/10.1109/90.502236

[10] J. Chao, "Saturn: a terabit packet switch using dual round robin," *IEEE Commun. Mag.*, vol. 38, no. 12, pp. 78–84, 2000. https://doi.org/10.1109/35.888261

[11] H. Zhou, J. Li, F. Hu, G. Hu, Y.-Q. Song, and L. He, "A novel traffic shaping algorithm with delay jitter constraints for real-time multimedia networks," in 2011 IEEE 16th Conference on Emerging Technologies & Factory Automation (ETFA), *ETFA2011*, 2011, pp. 1–4. https://doi.org/10.1109/ETFA.2011.6059155

[12] Chung-Ming Huang, Chung-Wei Lin, and Cheng-Yen Chuang, "A Multilayered Audiovisual Streaming System Using the Network Bandwidth Adaptation and the Two-Phase Synchronization," *IEEE Trans. Multimed.*, vol. 11, no. 5, pp. 797–809, Aug. 2009. https://doi.org/10.1109/TMM.2009.2021719

[13] R. Yang, R. D. van der Mei, D. Roubos, F. J. Seinstra, and H. E. Bal, "Resource optimization in distributed real-time multimedia applications," *Multimed. Tools Appl.*, vol. 59, no. 3, pp. 941–971, Aug. 2012. https://doi.org/10.1007/s11042-011-0782-5

[14] ITU, "One Way Transmission Time," 2003. *https://www.itu.int/rec/T-REC-G.114/en*

[15] J. Li, Z. Li, and P. Mohapatra, "Adaptive per hop differentiation for end-to-end delay assurance in multihop wireless networks," *Ad Hoc Networks*, vol. 7, no. 6, pp. 1169–1182, Aug. 2009. https://doi.org/10.1016/j.adhoc.2008.10.005

[16] I. Vaishnavi and D. C. A. Bulterman, "Estimate and serve: Scheduling Soft Real-Time Packets for Delay Sensitive Media Applications on the Internet," *18th international workshop on Network and operating systems support for digital audio and video - NOSSDAV '09*, 2009, p. 109. https://doi.org/10.1145/1542245.1542270

[17] Jyotiprasad Medhi, *Stochastic Models in Queueing Theory*. Academic Press, 2002.

[18] C. Demichelis and P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)," Nov. 2002. *https://tools.ietf.org/html/rfc3393*

[19] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Trans. Netw.*, vol. 3, no. 4, pp. 365–386, 1995. https://doi.org/10.1109/90.413212

[20] C. J. Sreenan, Jyh-Cheng Chen, P. Agrawal, and B. Narendran, "Delay reduction techniques for playout buffering," *IEEE Trans. Multimed.*, vol. 2, no. 2, pp. 88–100, Jun. 2000. https://doi.org/10.1109/6046.845013

[21] S. Floyd, "Measurement Studies of End-to-End Congestion Control in the Internet," 2013. [Online]. Available: http://www.icir.org/floyd/ccmeasure.html.

[22] K. Fall and K. Varadhan, *The ns Manual (formerly ns Notes and Documentation)*, no. 3. 2011. *https://www.isi.edu/nsnam/ns/doc/*

[23] S. Domoxoudis, S. Kouremenos, V. Loumos, and A. Drigas, "Modelling and Simulation of Videoconference Traffic from VBR Video Encoders," *in proceeding of the 2nd International Working Conf. on Performance Modelling and Evaluation of Heterogeneous Networks*, HET-NETs '04. Ilkley, West Yorkshire, UK, *2004*