First Video Streaming Experiments on a Time Driven Priority Network

Mario Baldi and Guido Marchetto
Department of Control and Computer Engineering
Politecnico di Torino (Technical University of Torino)

Abstract — As broadband access becomes more widely available and affordable, future Internet traffic will be dominated by streaming media flows, such as video-telephony, video-conferencing, high definition TV, 3D video, virtual reality, and many more. Consequently, networks will have to offer quality of service with scalable solutions — i.e., currently relied-upon overprovisioning is not likely to be a viable solution to accommodate streaming media traffic. This paper describes a testbed and experiments demonstrating the deployment time-driven priority scheduling — an implementation of pipeline forwarding — to support video streaming. The purpose of the presented experiments is to intuitively show the benefits the proposed solution provides to UDP-based streaming applications, while preserving efficient support for elastic TCP-based traffic.

Index Terms— Quality of service, UDP-based streaming, Testbed and experimentation, Efficient utilization of network resources, Traffic engineering

I. Introduction

High speed Internet access has by now become widely available potentially opening a large market to new services, that are potential new source of revenue for an ailing telecom market. Many service providers offer VoIP based telephony services, video broadcasting, and video on demand. Such applications are often referred to as *multimedia* or *real-time* as, contrary to traditional data applications, the timing of packet delivery is important for them to work properly. Packet networks, traditionally designed and deployed for data applications are not engineered to tightly control the delay packet experience in routers where they might contend for resources (e.g., transmission capacity) and consequently queued.

Right now the requirements of multimedia applications are commonly satisfied through *overprovisioning*, i.e., by keeping the network lightly loaded so that contention for network resources is low and queuing time consequently small. This approach is feasible as only a small fraction of broadband access subscribers are currently using such multimedia services and they deploy their Internet connection with traditional applications, such as web browsing and e-mail. Consequently, users do not fully deploy the large bandwidth of their access connections and both access and backbone networks are currently lightly loaded. Although some users more heavily exploit their broadband access with peer-to-peer file sharing applications, these do not require real-time service. As a consequence, the above overprovisioning approach can still be applied if coupled with traffic differentiation, e.g., according to the Differentiated Services solution [1], to separate and prioritize multimedia traffic. However, this approach is not any longer feasible if multimedia traffic grows to become dominant and technology does not evolve fast enough to enable a proportional enhancement of the network infrastructure.

The presentation of this work was supported by the European Commission under the E-Next Project FP6-506869

Pipeline forwarding [2] is particularly suitable to carry streaming media applications over the Internet since it offers:

1. Quality of service guarantees (deterministic delay and jitter, no loss) for (UDP-based) constant bit rate (CBR) and variable bit rate (VBR) streaming applications — as needed;
2. Support of elastic, e.g., TCP-based, traffic — i.e., existing applications based on “best-effort” services are not affected in any way;
3. High scalability of network switches (multi-terabit/s in a single chassis) [3],

This paper reports on the first experiments of video streaming through a testbed network of routers supporting time-driven priority (TDP) scheduling [2] that is an implementation of pipeline forwarding. The aim and contribution of this paper is to demonstrate in an intuitive and visual way, i.e., through the user perceived quality of the video stream played at the receiver, that

- The prototypal router implementation with TDP support works properly, thus providing the expected quality of service, and
- Multimedia streaming applications can benefit from it

Notice that although the experiments presented in this work have been done with one way streaming video, the results and considerations in this paper apply to interactive media as well, where the short and constant end-to-end delay observed in the experiments is even more critical.

Section II focuses on pipeline forwarding, the technology underlying the presented testbed, by presenting its operating principles and properties and how it can be deployed in current packet networks. The testbed on which the presented experiments are run is detailed in Section III that describes its architecture and the implementation of its main component, an IP router implementing TDP scheduling. Section IV describes the experiments, including their setup and outcome. Lesson learned and future research directions are discussed in Section V.

II. Underlying Principles and Technologies

A. *Pipeline Forwarding: Time-Driven Priority*

Pipeline forwarding is a known optimal method that is widely used in computing and manufacturing. The necessary requirement for pipeline forwarding is having common time reference (CTR). In the presented prototypal router UTC (coordinated universal time) is used for CTR, consequently, the method used in the testbed is called *UTC-based pipeline forwarding*. An extensive and detailed description of UTC-based forwarding is outside the scope of this paper and is available in [2].

In UTC-based pipeline forwarding all packet switches are synchronized and utilize a basic time period called time frame (TF). The TF duration (T_f) may be derived, for example, as a fraction of the UTC second received from a time-distribution system such as the global positioning system (GPS) and, in the near future, Galileo. As shown in Fig. 1, TFs are grouped into time cycles (TCs) and TCs are further grouped into super cycles, each super cycle lasts for one UTC second. TFs are partially or totally reserved for each flow during a resource reservation phase. The TC provides the periodicity of the reserved flow. This result in a periodic schedule for IP packets to be switched and forwarded, which is repeated every TC.

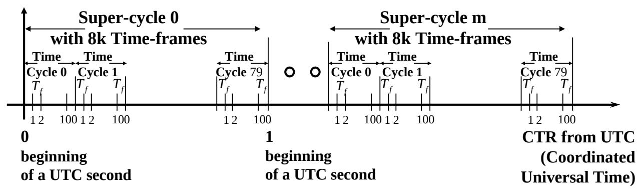

Fig. 1. Common time reference structure

The basic pipeline forwarding operation is regulated by two simple rules: (*i*) all packets that must be sent in TF *t* by a node must be in its output ports' buffers at the end of TF *t*-1, and (*ii*) a packet *p* transmitted in TF *t* by a node *n* must be transmitted in TF $t+d_p$ by node *n*+1, where d_p is an integer constant called *forwarding delay*, and TF *t* and TF $t+d_p$ are also referred to as the *forwarding TF* of packet *p* at node *n* and node *n*+1, respectively. The value of the forwarding delay is determined at resource-reservation time and must be large enough to satisfy (*i*). In pipeline forwarding, a *synchronous virtual pipe* (SVP) is a predefined schedule for forwarding a pre-allocated amount of bytes during one or more TFs along a path of subsequent UTC-based switches

UTC-based forwarding guarantees that reserved real-time traffic experiences: (*i*) bounded end-to-end delay, (*ii*) delay jitter lower than two TFs, and (*iii*) no congestion and resulting losses.

Time-driven priority (TDP) [2] is a synchronous packet scheduling technique that enables combining UTC-based pipeline forwarding with conventional routing mechanisms to achieve the high flexibility together with guaranteed service. While scheduling of packet transmission is driven by time, the output port can be selected according to either conentional IP destination-address-based routing, or multi-protocol label switching (MPLS), or any other technology of choice.

B. *Non-pipelined Traffic*

Non-pipelined (i.e., non-scheduled) IP packets — namely packets that are not part of a SVP (e.g., IP best-effort packets) — can be transmitted during any unused portion of a TF, whether it is not reserved or it is reserved but currently unused. Consequently, links can be fully utilized even if flows with reserved resources generate fewer packets than expected. Moreover, any service discipline can be applied to packets being transmitted in unused TF portions. For example, various traffic classes could be implemented for non-pipelined packets in accordance to the Differentiated Services (DiffServ) model [1]. In summary, pipeline forwarding is a best-of-breed technology combining the advantages of circuit switching (i.e., predictable service and guaranteed quality of service) and packet switching (statistical multiplexing with full link utilization) that enables a true integrated services network providing optimal support to both multimedia and elastic applications.

C. *Multimedia System Architecture*

Fully benefiting from UTC-based pipeline forwarding requires providing network nodes and end-systems with a CTR and implementing network applications in a way that they use it to maximize the quality of the received service. However, the Internet is currently based on asynchronous IP packet switches and hosts. Thus, especially in an initial deployment phase, UTC-based pipeline forwarding must coexist and interoperate with current equipment and applications (e.g., IP video-phones, video-streaming servers and clients, etc.), as depicted in Fig. 2. The experiments presented in this work reproduce such scenario: senders generate asynchronous multimedia traffic then entering a TDP domain. Edge TDP routers are connected to traditional asynchronous IP nodes through a SVP (synchronous virtual pipe) interface that polices and shapes incoming traffic flows. Specifically, asynchronous packets are stored in a buffer waiting for their previously evaluated forwarding TF.

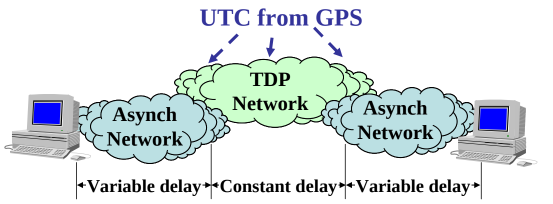

Fig. 2. Interoperation between TDP and asynchronous networks

III. Testbed

The network architecture for the interoperation of TDP and asynchronous networks presented in the previous sections has been demonstrated by building the testbed for video distribution shown in Fig. 3. In particular, we aim at showing the effectiveness of UTC-based pipeline forwarding by means of the quality of the service perceived by a viewer of a streaming video routed though the TDP network along the path highlighted in Fig. 3. Although current experiments have been done with one way streaming media, the results and considerations apply to interactive media as well, where short and constant end-to-end delays are even more critical requirements.

A. Architecture and Components

The testbed, which reproduces the network scenario in Fig. 2, consists of various asynchronous end-systems — implemented by two personal computers (PCs) and a router tester — interconnected by a UTC-based pipeline forwarding network consisting of four TDP routers. The two end systems, 2.4 GHz Pentium IV personal computers running Linux Fedora Core 3, implement a distribution system for one-way video based on the Fenice ver. 1.9 video server software and the Nemesi ver. 0.5.2 video streaming client [4]. A video stream enters the TDP network through TDP Router 1 and reaches the client through the other three TDP routers along the path shown in Fig. 3.

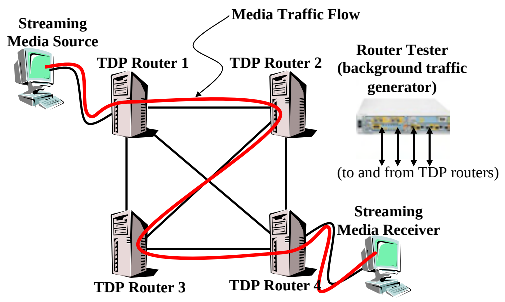

Fig. 3. Experimental testbed

All network interface cards deployed in the testbed are Intel PRO/1000 MT Gigabit Ethernet server adapters operating at 100 Mb/s. An Agilent N2X Router Tester is used to generate the two types of asynchronous flows described below that act as background traffic.

Delay insensitive flows — possibly modeling traditional Internet data traffic such as file transfers and e-mail exchanges — enter the network as non-pipelined traffic and are handled as best-effort traffic in the TDP network. This type of background traffic is used to experimentally verify that the capability to perfectly isolate pipelined traffic from the non-pipelined one has been properly implemented in TDP routers. In fact, a large amount of non-pipelined traffic, possibly overloading the network, is expected not to affect the service provided to pipelined traffic.

Delay sensitive flows — possibly modeling real-time traffic such as voice over IP (VoIP), video on demand, and videoconferencing — are handled as pipelined traffic in the TDP network. This type of background traffic is used to demonstrate the ability of pipeline forwarding to guarantee deterministic quality of service (QoS) also in case most — potentially 100% — network resources (e.g., transmission bandwidth) are dedicated to traffic with specific QoS requirements, e.g., real-time service. This is a significant improvement over other QoS approaches:

- The DiffServ model [1] assumes that differentiated traffic is only a small fraction on the network capacity;
- Conventional (asynchronous) techniques for guaranteeing service performance in packet networks [5], possibly adopted in the context of the Integrated Services model [6], do not allow to fully load the network with packet flows that require short delay and jitter, especially if they are low rate flows (see [7] for a detailed discussion).

B. TDP Router Implementation

The core of the testbed is the TDP network composed of four TDP routers. These are based on the routing software of the FreeBSD 4.8 operating system running on a 2.4 GHz Pentium IV PCs; the TDP scheduling algorithm is implemented in the FreeBSD kernel [8].

Generically, a router data plane moves packets from input ports to output ports through three modules that perform *input processing*, *forwarding*, and *output processing*, respectively. The operations performed by each module of a TDP router are briefly discussed in the following (see [8] for details).

The input module of an interface connected to another TDP router determines the forwarding TF of each pipelined packet by adding the forwarding delay to the estimated forwarding TF at the previous node. The current router implementation leverages on the DS field [1] of the IP header to

- Distinguish pipelined packets from non-pipelined packets
- Ensure that the estimate of the forwarding TF is correct even in case multiple packets are lost (see Section III.B of [8] for protocol details).

The forwarding module processes packets according to the specific network technology; the presented prototype performs conventional IP routing and forwarding.

Consequently, TDP support concerns mainly the output module where a per-TF, per-output queuing system is needed to store packets while waiting for their forwarding TF to begin. The queue in which each packet ends up is determined by both the input module — deciding the forwarding TF — and the forwarding module — determining the output interface. The output module also responsible for the timely transmission of all the packets stored in the queues corresponding to a TF when it begins and before it ends.

The input module of SVP interfaces (i.e., interfaces of a boundary TDP router not connected to a pipeline forwarding node) includes mechanisms to

- Classify each incoming packet to identify the data flow it belongs to
- Determine, based on the flow's resource reservation, the TF in which the packet should be forwarded by the output module (i.e., its forwarding TF).

UTC is provided to our prototypal router by a Symmetricom GPS receiver PCI card that can generate interrupts at a programmable rate ranging between less than 1 Hz (1PPS — pulse per second) and 250 kHz (every 4 μs). Such interrupts are used to pace the beginning of TFs; whenever an interrupt occurs the values of the current TF and TC are updated. The current version of the prototype does not implement signaling functions, i.e., TFs and TCs are statically allocated to flows through manual configuration.

IV. Experiments

A. Basic System Parameters

The current router implementation does not support preemptive priority, i.e., the capability of interrupting the transmission of a non-pipelined packet in a non-disruptive way when a new TF begins (see Section 2.2 of [2] for details). Consequently, if a portion of a TF is not used by pipelined traffic and a non-pipelined packet does not fully fit in it, the current implementation leaves the link idle, thus lowering link utilization. As this would not happen in a full-fledged implementation with support for preemptive priority, system parameters are chosen to avoid the above situation: packet size is chosen such that the transmission of an integer number of packets lasts (basically) a whole TF. Having set the TF duration to 250 μs, 25,000 bits (i.e., about 3 KB) can be transmitted during one TF and 1 KB packets are deployed..

B. *Resource Reservation*

The needed amount of bandwidth should be reserved in proper portions of TFs for the reference real-time flow depicted in Fig. 3, a 10 Mb/s MPEG video stream. As the deployed video server generates 1 KB packets, capacity for at most three video packets could be reserved during each TF. Since the video stream results from encoding 25 video frames per second, the average frame size is about 49 KB. However, an MPEG codec produces a significantly different amount of bits for a frame depending on which of the following encoding it is using[1].

Intra-frame Coding eliminates spatial redundancy inside pictures and the resulting encoded picture is called *I-frame*.

Predictive Coding eliminates temporal redundancy between a picture and the previous one through motion estimation. The obtained encoded picture is called *P-frame* and it is typically from 2 to 4 times smaller than an I-frame.

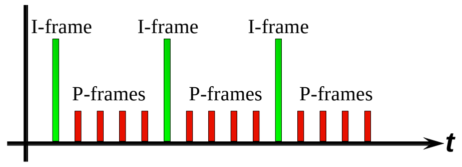

Fig. 4. MPEG video stream

Normally codecs apply intra-frame coding and predictive coding on different frames according to a fixed, re-occurring pattern, as depicted in Fig. 4. However, I-frame and P-frame size is generally unpredictable. Moreover, some codecs can use intra-frame coding and predictive coding on different portions (macroblocks) of the same frame. This results in a packet flow with a very variable bit rate.

Having configured the TC to be composed of 160 TFs, one video frame is transmitted every TC. A network analyzer was used to observe the traffic corresponding to the video stream deployed in our experiments in order to determine how much transmission capacity to reserve during each TC. The maximum size of a video frame, i.e., the maximum burst size, resulted to be about 100 KB. In order to minimize end-to-end delay while avoid packet loss (i.e., in order to ensure that a whole frame can be transferred during a single TC) the capability of transmitting 100 KB (i.e., 100 packets) each TC must be ensured. For example, capacity for transmitting 3 packets could be booked during 34 TFs. This results in bandwidth overallocation (about 20 Mb/s for a 10 Mb/s flow) and low efficiency in the utilization of network resources. However, such issue is beyond the scope of the current experiments that primarily aim at verifying the correct operation of the system. Section V discusses ways to improve utilization of reserved resources, which is key in engineering a scalable solution.

In order to limit the variation of the delay introduced by the SVP interface on video packets, the TFs in which resources are allocated should be as evenly distributed as possible across the TC. Issues related to minimizing the jitter due to the SVP interface are outside the scope of this work.

C. *Background Traffic Pattern*

Delay sensitive background traffic is generated by the router tester as a set of CBR flows with destination and bit rate chosen to maximize TF utilization and contention on the links traversed by the streaming video. Such links can host up to 78.75 Mb/s of additional pipelined traffic to be transmitted during the 126 TFs not reserved to the video stream.

A possible solution to have delay sensitive background traffic fully load the links traversed by the streaming video flow is to define ten 7.875 Mb/s CBR flows that follow the same path. However, such a traffic pattern does not maximize resource contention, which potentially causes long queuing delays. In fact, after packets from the video stream and other delay sensitive flows have contended for transmission on the link between TDP router 1 and TDP router 2 in Fig. 3, they can stream through the subsequent links in the same order without further contention, hence without being possibly queued. This obviously results in limited delay and jitter even if no particular QoS oriented scheduling mechanisms, whether TDP or conventional ones, are deployed in output modules. Consequently, the experiment would have little significance[2].

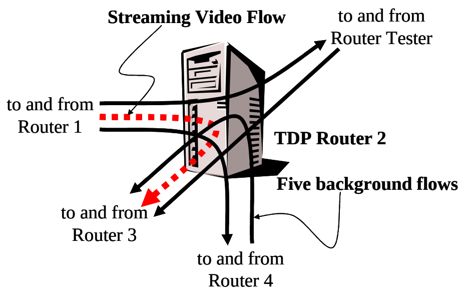

Fig. 5. Link contentions on TDP Router 2

A more complex traffic scenario is therefore used in the presented experiments. Thirty 7.875 Mb/s flows enter the TDP network from the various TDP routers and follow paths defined in such a way the reference video flow competes with other ten delay sensitive flows at each hop, as shown in Fig. 5 for TDP router 2. The dotted line represents the video stream, while each of the continuous lines represents a group of five delay sensitive background flows. The ten background flows sharing an output link with the video stream arrive from different input links and consequently actually contend for the

[1] Actually, a third type of encoding, called bi-directional predictive coding exists; without loss of generality, it was not considered in this work.

[2] Notice that as far as TDP is concerned, its principles of operation are such that contention is avoided in any case. Realizing a scenario in which contention naturally occurs is essential in order to (*i*) verify proper operation of the TDP implementation and (*ii*) demonstrate TDP theoretical properties.

transmission capacity as their packet arrival processes are, in general, independent. Once they reach TDP router 3, the 2 groups of five background flows follow different paths, thus contending again with the streaming video but for different links.

According to the described traffic scenario, the total amount of bandwidth reserved on the links traversed by the video stream is 99.15 Mb/s (out of 100 Mb/s). Unused bandwidth, i.e., parts of TFs, are deployed for transmitting delay insensitive background traffic that is provided with a best-effort service, i.e., queued in a queue served by a FIFO (first in first out) scheduler. Delay insensitive background flows follow similar routes as delay sensitive ones in order to maximize contention for them as well. As several 10 Mb/s flows are generated, links are overloaded and a significant amount of packets is discarded.

D. *Jitter Control and Compensation*

Replay buffers are commonly implemented in media streaming clients to absorb the jitter experienced by packets across the network. Since the network topology in the presented testbed is very simple (hence routers have few interfaces), even with QoS unaware packet scheduling algorithms, such as FIFO, jitter does not grow very large in spite of the complex traffic patterns defined for maximizing resource contention. Although replay buffers with typical sizes would certainly suffice to absorb the jitter accumulated in the presented testbed, this is not the case in general.

On the other hand, the jitter on a TDP network is very low and independent of the path (i.e., number of nodes) and traffic. In order to offer a user perceivable demonstration of the effectiveness of TDP in limiting jitter, the replay buffer size is minimized. Specifically, in our experiments the replay buffer is set to 1,096 bytes — which is the minimum size allowed by the deployed client software. As 1 KB packets are generated by the video server, the content of each packet is decoded immediately as the corresponding packet is received, without waiting for the replay buffer to fill up in order to compensate possibly late packets.

E. *Results*

In all the experiments run in the described scenario the video stream is replayed at the receiver with optimal user perceived quality and unnoticeable delay. As no visible losses occur, neither router output buffers nor video client replay buffer overflow, i.e., TDP scheduling actually limits packet jitter as expected.

This result is especially significant considering that about 89% of the network capacity is on average used by delay sensitive traffic — each 100 Mb/s link traversed by the 10 Mb/s video stream carries an additional 79 Mb/s of delay sensitive background traffic. This result is quite far from what could be achieved with a DiffServ approach [1] that heavily relies on the assumption that differentiated (e.g., delay sensitive) traffic is only a small fraction of the network capacity. Also conventional (asynchronous) QoS techniques [5], possibly adopted in the context of the Integrated Services model [6], hardly enable delay sensitive traffic to account for 89% of the link capacity, while guaranteeing short delay and jitter. This is especially hard if delay sensitive traffic is composed of low rate flows [7]; although the experiments presented here are related to a high rate video stream, the achieved results are independent of the flow rate, as it can be easily inferred from the simple TDP operating principles [2]. Finally, TDP is a very simple and scalable scheduling discipline, while conventional QoS techniques with best properties (e.g., WFQ) feature high implementation complexity and suffer from limited scalability (i.e., applicability in large scale, significant scenarios).

Measurements taken in experiments run on a similar network topology demonstrated TDP properties in terms of limited jitter and expected delay, independently of the number of hops, also when links where fully loaded (not “just” 90%) by (synthetic) delay sensitive traffic [8].

V. Discussion and Improvements

As previously mentioned, in the presented experiments the network could not be fully loaded because, because a large amount of bandwidth is allocated to the video stream as a simple way of coping with the unpredictability of its rate. Specifically, enough capacity is allocated during each TC to transmit a maximum size video frame. This results in allocating twice the average video stream rate; an even larger ratio of allocation over average rate could result for video streams that have few very detailed and/or very fast scenes that result in few frames much larger than all the others. This approach was used only as a “quick fix” to enable us to obtain first results that visually demonstrate both proper operation of the testbed and TDP properties. However, the approach goes against the very principles that motivated this work as it results in low utilization of reserved resources, i.e., low *reservation efficiency* as it is called in the context of this paper. The following subsections discuss various ways to maximize reservation efficiency. Although they are not implemented in the current testbed, some of them are the object of ongoing work.

A. *Limited allocation without losses and large delay*

Capacity is booked in each TC so that the allocated rate is larger than the video stream rate (although smaller than the reservation deployed in the presented experiments). When the amount of packets generated by the video server for a video frame is larger than the capacity booked during a TC, the exceeding packets can be transmitted by the SVP interface in the following TC (or TCs). Consequently, packets are buffered in the SVP interface for a time, possibly spanning multiple TCs (i.e, video frame periods), that depends on the burstiness of the video stream and the reservation. Moreover, as the delay experienced at the SVP interface by the packets of the video stream is highly variable, a correspondingly large replay buffer is required at the client.

A larger capacity allocation reduces the delay (and jitter) experienced by packets and buffering requirements at the SVP interface, but lowers reservation efficiency. In general, this solution is not suitable for interactive applications, such as telephony and videocoferencing, but could be applied for one way streaming media possibly featuring a low bit rate (in order to limit buffer requirements at the SVP interface).

B. *Limited allocation with possible loss and limited delay*

Delay could be reduced by relaxing the requirements on loss, i.e., by allowing the possibility that a certain percentage of packets be lost. In particular, the size of the queue for the video stream at the SVP interface is limited and possibly overflowing packets are either discarded or forwarded in the network as non-pipelined traffic. Obviously, the quality of the video played at the client will be degraded depending on the allocated capacity in each TC, the burstiness of the video stream, the size of the queue at the SVP interface, and the network load.

The user perceived quality can be improved, while keeping the same delay bound (i.e., queue size and allocation), by

- Handling non-pipelined traffic according to the DiffServ model;
- Taking into account the perceptive importance of the video packets when deciding which ones to forward in the network using TDP and which ones to handle as non-pipelined traffic.

The latter approach, which we consider very promising, needs to be defined in detail, thoroughly studied, and validated, which is the object of ongoing work.

C. *Optimal allocation without losses and optimal delay*

As argued in [9], an optimal solution for the transmission of (interactive) media is obtained by synchronizing to UTC the video server (or video codec in case of real-time video). In fact, in this case the allocation can be minimized while optimizing the delay introduced by the network and applying UTC-based forwarding to all video packets. In particular, a different amount of capacity can be allocated in different TCs following the pattern of I-frames and P-frames. For example, with reference to the sample video stream depicted in Fig. 4 and given a super-cycle of 25 TCs (as it is in our experiments):

- The capacity needed to send the amount of bytes encoding an I-frame is allocated during TCs 0, 5, 10, 15, and 20. In order to minimize end-to-end delay, the allocation should be done during subsequent TFs.
- The capacity required to send the amount of bytes encoding a P-frame is allocated during the remaining TCs. In order to minimize end-to-end delay, the allocation should be done during the first TFs allocated in TCs 0, 5, 10, 15, and 20.

In order to minimize end-to-end delay, the video server should be programmed to (encode a frame and) generate the packets corresponding to a frame during the TFs reserved to the video stream. Additionally in case of real-time video, the codec should be programmed to finish encoding each video frame right before the set of allocated TFs.

However, as pointed out before, the size of I-frames, as well as P-frames, is not constant. This can be handled in one of the following ways:

- A reservation larger than the average stream rate is performed by allocating the size of the largest frame for both I-frame and P-frame reservation. Reservation efficiency might still be quite high compared to the solution described in Section IV.B because the size difference between I-frames, as well as P-frames, is typically smaller than the one between I-frames and P-frames. However, this solution might be impractical because, especially in real-time (e.g., interactive) video applications, the maximum size of encoded frames cannot be known at resource reservation time.
- Allocation is performed based on the, possibly estimated, average frame size and packets exceeding the reservation are either discarded or forwarded in the network as non-pipelined traffic. The resulting quality degradation can be controlled by
  - deploying the DiffServ model for non-pipelined traffic,
  - taking into account the perceptive importance of the video packets when deciding which ones to forward in the network as non-pipelined traffic
  - dynamically adjusting the resource reservation based on the actual size of encoded frames, i.e., the characteristics of the video scenes.
- In case of real-time video, the codec is modified to take into account the resource reservation and generate an amount of bytes for each encoded video frame as close as possible to the corresponding reservation [9], while maximizing the user perceived quality. In addition, in order to optimize the perceived quality the resource reservation could be adjusted based on the feedback from the codec on the characteristics of the video scenes.

All the approaches presented in this section are considered very promising as long term solutions (as they require UTC-aware end systems and applications) and will be subject of our future research.

Acknowledgment

The authors wish to thank Flavio Bonatesta for setting up the testbed to run the presented experiment.

References

[1] S. Blake *et al.*, *An Architecture for Differentiated Services*, IETF Std. RFC 2475, Dec. 1998.

[2] C.-S. Li, Y. Ofek, and M. Yung, Time-driven priority flow control for real-time heterogeneous internetworking, in *Proc. IEEE (INFOCOM' 96), vol. 1*, (Mar. 24–28, 1996), 189–197.

[3] M. Baldi, Y. Ofek, "Multi-Terabit/s IP Switching with Guaranteed Service for Streaming Traffic," *IEEE INFOCOM 2006 High-Speed Networking Workshop*, Barcelona (Spain), Apr. 2006.

[4] (LS)³ - Libre Streaming, Libre Software, Libre Standards. An open multimedia streaming project, "(LS)³ Tools," [Online] Available at: http://streaming.polito.it/tools

[5] H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks," *Proceedings of the IEEE*, Vol. 83, No. 10, Oct. 1995.

[6] R. Braden, D. Clark, S. Shenker, "Integrated Services in the Internet Architecture: an Overview," *IETF Std. RFC 1663*, July 1994.

[7] M. Baldi, F. Risso, "Efficiency of Packet Voice with Deterministic Delay," *IEEE Communications Magazine*, Vol. 38, No. 5, May 2000, pp. 170-177.

[8] M. Baldi, G. Marchetto, G. Galante, F. Risso, R. Scopigno, F. Stirano, "Time Driven Priority Router Implementation and First Experiments," *IEEE International Conference on Communications (ICC 2006), Symposium on Communications QoS, Reliability and Performance Modeling*, Istanbul (Turkey), June 2006.

[9] M. Baldi and Y. Ofek, "End-to-end Delay of Videoconferencing over Packet Switched Networks," *IEEE/ACM Transactions on Networking*, Vol. 8, No. 4, Aug. 2000, pp. 479-492.