Triple Play Support for the Next Generation Internet

Mario Baldi
Politecnico di Torino — Dipartimento di Automatica e Informatica
Corso Duca degli Abruzzi, 24 — 10129 Torino — Italy
+39 011 564 7067 — mario.baldi@polito.it

ABSTRACT

The Internet has the potential to become the ubiquitous and universal means of accessing any type of information, whether pulled by the user or pushed by a provider. However, in order to be the infrastructure of choice for this role, the Internet must represent a business opportunity to information providers or, at least, be economically self-sustainable. In other words, Internet technology must support services in such a way that both people are willing to pay for them and the cost of delivering them is low. The former implies that services, besides being appealing, must be reliable and of reasonably high and persistent quality. The latter requires low complexity technological solutions and convergence to a single network infrastructure that supports, in an integrated manner, traditional Internet applications (e.g., web, e-mail, and file sharing), telephony, and video, a.k.a. triple play. Since today's solutions resort to overprovisioning to satisfy the quality requirements, they fail to minimize costs due to low resource utilization efficiency. This paper presents a low complexity solution for proving triple play services as it enables integrating different applications so that each received the service it requires, possibly in a guaranteed fashion, while ensuring high resource utilization efficiency.

Preferred Topic Areas: 1c, 2d, 2e, 4c

1. INTRODUCTION

The Internet has been a major and successful cultural phenomenon in the past few years. The drivers to its success have been the convenience and benefits of applications like e-mail and web browsing, combined with low cost and wide reach. However, today the Internet needs something else to remain successful and maintain its promise as a ubiquitous and universal means of accessing any type of information: it must be a profitable business or, at least, economically self-sustainable.

In order to achieve this objective, services people are willing to pay for must be offered over the Internet and the cost of delivering them must be low. In order to be able to properly bill for such services, they must be reliable and feature high and persistent quality. Moreover, a triple play solution with low technological complexity is needed to support on a single network infrastructure traditional Internet applications (e.g., web, e-mail, and file sharing), telephony, and video.

Current solutions to integrating traffic of heterogeneous nature on the same network infrastructure leverage on overprovisioning to ensure satisfactory quality. This leads to poor utilization of network resources, thus conflicting with the objective of minimizing costs. This paper presents an optimal solution for supporting triple play through IP networks, and specifically the Internet, as it offers:

- Low complexity, hence high scalability of routers;
- Quality of service guarantees (deterministic delay and jitter, no loss) for (UDP-based) constant bit rate (CBR) and variable bit rate (VBR) streaming applications;
- Unmodified support for best-effort and differentiated services [6], which enables leaving unchanged
  - The service received by elastic, e.g., TCP-based, applications, and
  - Currently employed network design and traffic engineering practices and models.

Proper and efficient support of streaming UDP-based applications is getting increasingly important due to the fact that more and more streaming media traffic (e.g., Skype) is transported over the Internet and IP networks in general using UDP. Such applications need a minimum level of service quality in order to operate properly and current approaches to offer controlled quality based on overprovisioning do not scale. Moreover, applications such as telephony, videoconferencing, and various types of entertainment services based on video are promising sources of significant revenue.

The network architecture presented in this paper has recently been demonstrated [5][9] by means of a testbed based on the prototypal implementations of a PC-based router [3] and an opto-electronic switch [5]. Section 2 discusses the basic operating principles of UTC-based pipeline forwarding, the technology underlying the presented solution to triple play support. Section 3 outlines a network architecture enabling incremental deployment of the presented solution over the exiting network infrastructures, such as the Internet. Finally, Section 4 discusses how to support various classes of applications such as SIP based telephony, peer-to-peer overlay-based conferencing (e.g., Skype), and video-based services.

2. UTC-BASED PIPELINE FORWARDING

Implementing *UTC-based pipeline forwarding* requires packet switches to be synchronized with a *common time reference* (CTR) [1]. UTC can be ubiquitously received from a time-distribution system such as the global positioning system (GPS) and, in the future, from Galileo. Consequently, it constitutes a natural candidate for a globally available CTR.

2.1 Basic Principles

In the following scheme IP packet switches are synchronized and use a basic time period called time frame (TF). The TF duration is derived from the UTC second. Time frames are grouped into time cycles (TCs) and TCs are further organized into super cycles, each of which typically lasts one UTC second. The transmission capacity during each TF is partially or totally reserved to one or more flows during a resource reservation procedure. The TC provides the basis for a periodic repetition of the reservation, while the super cycle offers a basis for reservations with a period longer than a TC. This results in a periodic schedule for packets to be switched and forwarded, which is repeated every TC or every super cycle. For example a 125 μs time frame is obtained by dividing the UTC second by 8000; sequences of 100 time frames are grouped into one TC, and runs of 80 TCs are comprised in one super cycle (i.e., one UTC second).

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*. The value of the forwarding delay is determined at resource-reservation time and must be large enough to satisfy (*i*).

The periodic scheduling within each node results in a *periodic packet forwarding* across the network, which is also referred to as *pipelined forwarding* for the ordered, step-by-step fashion, with which packets travel toward their destination. 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.

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. The SVP capacity is determined by the total number of bytes allocated in every TC for the SVP. For example, for a 10 ms TC, if 20000 bits are allocated during each of 2 TFs, the SVP capacity is 400 Kb/s.

2.2 Implementation Options

Two implementations of the pipeline forwarding were proposed: Time-Driven Switching (TDS) and Time-Driven Priority (TDP).

Time-driven switching (TDS) was proposed to realize sub-lambda or fractional lambda switching (FλS) in highly scalable dynamic optical networking [1], which requires minimum optical buffers. In the context of optical networks, SVPs are called fractional lambda pipes (FλPs). In TDS all packets in the same TF are switched the same way. Consequently, header processing is not required, which results in low complexity (hence high scalability) and enables optical implementation.

Time-driven priority (TDP) [1] is a synchronous packet scheduling technique that implements UTC-based pipeline forwarding and can be combined with conventional IP routing to achieve the abovementioned flexibility. Operation in accordance to pipeline forwarding principles ensures deterministic quality of service and low complexity packet scheduling. Specifically, packets scheduled for transmission during a TF are given maximum priority; if resources have been properly reserved, all scheduled packets will be at the output port and transmitted before their TF ends.

2.3 Non-pipelined Traffic

Non-pipelined IP packets, i.e., 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.

Each TDP node performs statistical multiplexing of best-effort traffic, i.e., inserts best-effort packets in unused TF portions. Therefore, SVPs are not at all TDM-like circuits: SVPs are virtual channels providing guaranteed service in terms of bandwidth, delay, and delay jitter, but fractions of the link capacity not used by SVP traffic can be fully utilized. Moreover, any service discipline can be applied to packets being transmitted in unused TF portions. For example, various traffic classes according to the Differentiated Service (DiffServ) paradigm [6] could be implemented for non-pipelined packets. 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.

Since in TDS switching is based on time, statistical multiplexing of best-effort traffic might not provided at each node. Nevertheless, best effort packets can be inserted at the TDS network edge in any unused portion of a proper SVP based on their destination. Although with somewhat limited flexibility compared to TDP, statistical multiplexing of best-effort packets and high link utilization can be achieved. Alternatively, a TDS switch can be designed to include the capability for extracting best-effort packets from the flow entering an input interface by means of a simple filter (for example, based on one bit, e.g., in the DiffServ field, of the packet header). Separated packets are handled by a functional unit performing conventional IP routing and “injected” in the flow of packets exiting the proper interface whenever it is idle. Scalability is not compromised as pipelined packets are handled on a fast data path performing TDS operation (i.e., switched based on their TF). Only best-effort packets — which possibly are a small fraction and anyway do not expect any specific service — use a slower data path.

3. DEPLOYMENT

3.1 General Network Architecture

Fully benefiting from UTC-based pipeline forwarding requires providing network nodes and end-systems with a CTR and implement 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 IP hosts. Thus, especially in an initial deployment phase, the UTC-based pipeline forwarding must coexist and interoperate with current asynchronous packet switches and hosts.

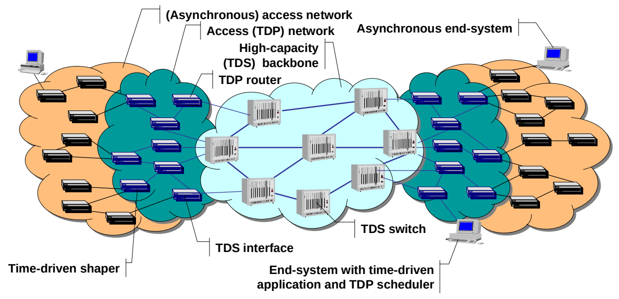
(Asynchronous) access network
Access (TDP) network
Asynchronous end-system
High-capacity (TDS) backbone
TDP router
TDS switch
TDS interface
Time-driven shaper
End-system with time-driven application and TDP scheduler

Fig. 1. Network architecture for the deployment of pipeline forwarding

Fig. 1 depicts a possible network architecture where senders generate asynchronous traffic that passes through an asynchronous access network towards the first TDP access node. *Edge TDP routers* are connected to traditional asynchronous IP routers through a *SVP interface* that controls access to the pipeline forwarding network by policing and shaping the incoming traffic. Fig. 1 also shows how, in order to get the best out of pipeline forwarding, TDS switches are deployed in the network core, while TDP routers are used at the backbone edge. In fact, the former feature particularly low complexity, hence high performance at low cost, while the latter are being used where scalability is not a primary concern, but their flexibility is particularly beneficial.

It is worth highlighting that it is not mandatory for each node to receive UTC from an external source (such as GPS or Galileo) in order to implement a ubiquitous CTR. The network can be organized in areas where one node externally receiving UTC distributes timing information to other nodes. Various alternatives with significantly different complexity and performance are possible; as an example, CTR distribution among TDP routers could be based on the alternate bit protocol presented in [3] for TF delineation and time-stamping. Interestingly, the time uncertainty resulting from such a low complexity and robust CTR distribution mechanism does not affect the correct operation of TDP routers. Although distribution of the CTR within the network is subject of ongoing analytical and experimental work, it can be easily anticipated that distribution of CTR can be more suitable for TDP routers, where complexity is not a primary concern and additional processing is not an issue, while deployment of an external source might be the most cost effective solution in TDS switches, where minimal complexity and possibly no packet processing are key to preserve scalability and make optical implementation feasible.

3.2 Resource Reservation

When a SVP is set up, resources — in the form of the capability of transmitting a certain amount of bits during specific TFs — are reserved for packets carried by the SVP. According to the Integrated Services (IntServ) model [7] quality of service (QoS) is guaranteed to single packet flows by reserving resources on a per-flow basis. This approach proved not scalable due to the large number of flows nodes in the core of the network deal with. IntServ scalability issues concern both the data and control plane. In fact, per-flow queuing and complex scheduling is required on the data path, while the control plane must process signaling and store booking information for each flow traversing a node. Pipeline forwarding does not share scalability issues related to the data plane, since it can provide per-flow service guarantees without per-flow queuing. Also the control plane is simpler since resource information consists in arrays of counters (or bits in TDS) accounting for reserved or available capacity during TFs and processing booking requests involves simple addition and subtraction (or logical OR operations in TDS). Nevertheless, avoiding per-flow reservation might be preferable given the large amount of signaling messages that nodes must potentially process when be deployed in the core.

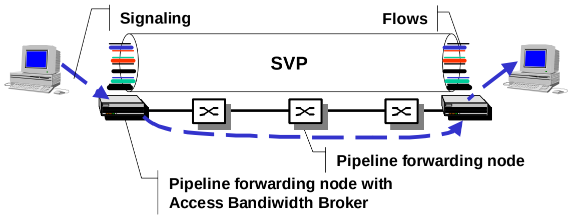

Fig. 2. Per-flow bandwidth reservation with access bandwidth broker

An access bandwidth broker (ABB) [7] at the edges of an SVP can handle signaling requests from applications whose packet flows are to traverse the SVP and determine the availability of resources within the SVP. If a request is accepted, the ABB reserves a fraction of the SVP resources to the corresponding flow. As shown in Fig. 2, intermediate switches are not involved in the signaling operation, but the micro-flow will receive deterministic QoS guarantees, even though intermediate switches on the SVP do not have any awareness of the micro-flow.

4. APPLICATION SUPPORT

This section discusses how pipeline forwarding can be deployed to support various applications currently widely used over the Internet. The discussion is organized in two parts, which are complementary aspects of such support: (*i*) how packets generated by such applications can be transported through the network, and (*ii*) how existing applications can accommodate the needed signaling and resource reservation procedures key for pipeline forwarding to be able to offer service guarantees.

4.1 Data Transfer

The issue is addressed organizing applications into various classes based on the characteristics of their traffic —e.g., streaming, periodic, irregular — and of the service they need — e.g., real-time, minimum bandwidth, elastic.

4.1.1 Streaming Applications with Periodic Bursts

Pipeline forwarding of traffic generated by real-time streaming applications was addressed in detail by previous work (see, for example, [3]). The following subsections include only a summary of the issue to provide a basis for discussion; the reader is redirected to existing literature for details.

Streaming applications are characterized by more or less continuous, long lasting flows of packets, i.e., the duration of a flow is much longer than the inter-packet time. This justifies the signaling and resource reservation overhead required to set-up an SVP. If the packet flow has a constant bit rate (e.g., telephony, voice, audio streaming, constant bit rate video streaming), resources can be reserved in a number of possibly equally spaced TFs in such a way that the corresponding bit rate is greater than or equal to the average bit rate of the flow rate. This guarantees that packets will be transported though the network without loss, with known delay, and with limited jitter bounded by the maximum time difference between two TFs with an allocation for the flow.

However, many streaming applications deployed today generate periodic bursty traffic. Consider, for example, both audio and video transmission. Audio is most often encoded at (low) constant bit rate, however, encoded samples are not sent out immediately as enough of them have to be gathered first to assemble a payload large enough. In fact, the protocol stack commonly deployed for real-time applications on IP networks (i.e., data-link protocol, IP, UDP, RTP) results in headers overall longer than 40 bytes. Consequently, packets exit the sending end system at constant intervals, i.e., each packet can be seen as a short periodic burst whose period depends on the encoder output rate and the chosen payload size.

Digital video is based on capturing, digitizing, compressing, and encoding video frames at periodic intervals ranging from 25 ms to somewhere around 1 s for very low quality, low bandwidth video. Some encoding algorithms compress and encode each video frame by itself[1], package the resulting bytes into one or more packets, and send them in the network. This results in a periodic burst of packets whose period is the video frame rate (i.e., ranging from 1 to 40 bursts per second).

Given that the source generates traffic periodically and pipeline forwarding is based on periodic scheduling, if resources can be reserved with the same period and packet burst can be transmitted right before the TF in which a reservation was made — which implies some level of synchronization between the video source and the network — end-to-end delay can be minimized, as analyzed in detail in [3]. If the video source is not synchronized with the network, additional delay is introduced by the interface; however, if the reservation has been properly performed, the video flow is carried with deterministic quality, i.e., without congestion and with bounded delay.

More effective video encoding algorithms achieve better compression, i.e., lower flow bit rates, by eliminating temporal redundancy between subsequent frames. As a consequence, some video codecs encode a number of frames, called P-frames, by reference to a previous one, called an I-frame, thus producing a smaller amount of bits. Simply put, the larger the ratio between the number of P-frames and the number of I-frames, the smaller the bit rate of the video stream. Obviously, I-frames result in the transmission of more packets than P-frames and the corresponding flow is characterized by bursts at regular intervals, i.e., the video frame period, but with different size. Such a flow is said to have a complex periodicity and it can be accommodated by reserving a different amount of transmission resources in different TFs. Some level of synchronization between the video source and the network is needed in order to avoid a large delay at the SVP interface.

Bursts generated by sources with both normal and complex periodicity might not have constant size. A burst smaller than the corresponding capacity allocation is not problematic since unused bandwidth will be exploited by non-pipelined traffic, i.e., the extra allocated capacity is not wasted. A burst larger than the corresponding allocation must be properly handled by the SVP interface where a proper policy must be implemented. As an example, packets exceeding their reservation can be either discarded or forwarded as best-effort by the SVP interface. Alternatively, if the pipeline forwarding network implements a differentiated service (DiffServ) on non-pipelined traffic, non compliant packets can be marked as belonging to a class receiving, for example, an expedite forwarding service. Although, in accordance to the DiffServ paradigm, no quality of service is guaranteed to packets exceeding their reservation, the differentiated service they receive might be good enough.

In summary, pipeline forwarding is particularly suitable to real-time streaming applications with periodic bursts as they can receive deterministic service with minimum delay. However, the deployment of pipeline forwarding is advantageous also for applications that do not have real-time requirements, but simply need their data stream to be

[1] Such video encoders generally feature higher robustness, lower computational complexity, shorter encoding delay, but lower compression rates; consequently, are not the most commonly deployed.

transferred through the network. In this case, the bounded, possibly minimum, delivery time comes as a bonus as it does not have any cost in terms of additional network node complexity or extra resource allocation.

4.1.2 Real-time Non-streaming Applications

The pipeline forwarding deployment model presented in this paper is not fitted to real-time applications that generate traffic only occasionally. In fact, it is not practical to setup an SVP before transmitting an occasional packet (burst) and tear it down right afterwards. Depending on the size of the bursts and how often the source generates traffic, at least two approaches, or a combination thereof, are possible.

As a first option, a SVP is setup by reserving transmission capacity in one or more TFs beforehand. Hence, when the source generates traffic, it can be transported through the SVP without requiring previous signaling. Reserved capacity will not be used whenever the source does not transmit longer than the time between two reserved TFs, e.g., a TC duration. Nevertheless, such capacity is not wasted as it can be used for transmitting non-pipelined traffic. If reservation efficiency, i.e., the fraction of resources reserved that are actually used, is a concern, than this approach is not advisable for applications that generate traffic with either a period much longer of the TC or very large bursts.

Another option involves implementing a differentiated service (DiffServ) of non-pipelined traffic; occasional real-time packets are marked as belonging to a class receiving, for example, an expedite forwarding service. In general, any other policy commonly applied on such traffic on a conventional IP network, like giving real-time packets static priority over the rest of the traffic, can equally be implemented in the context of a pipeline forwarding network. However, given that in the latter case periodic traffic is handled separately, the amount of traffic in the differentiated class or at high priority will be limited. This simplifies network and traffic engineering and complying with the main underlying assumption of the DiffServ paradigm: differentiated traffic shall be a small fraction of the network capacity. Although, in accordance to the DiffServ paradigm, no quality of service is guaranteed to the above non-pipelined packets, a (fraction of transmission capacity in a) number of TFs can be left unallocated in order to ensure that enough transmission resources are available to non-pipelined traffic so that the differentiated service they receive be good enough.

4.1.3 Non-real-time Elastic Applications

Today most data exchanges over IP networks are carried out over TCP by applications such as file transfers, web browsing sessions, e-mail transfer. These applications, that usually have no or very loose real-time requirements, are said to be elastic as they can operate with widely ranging services. As an example, whether data can be transferred at a high bit rate (say, 10 Mb/s) or at a very low bit rate (e.g., a few Kb/s), a file can anyway be transferred successfully; the larger the available bandwidth, the shorter the time needed[2]. TCP, by varying the size of its transmission window, adapts to resource availability in the network trying to make sure that applications achieve maximum performance.

Various approaches can be deployed to support elastic applications on TDP networks, the main characterization being based on whether their traffic is pipelined or not. For example, it can be transferred with best-effort service, which maintains the found by TCP on today's Internet. As a variant, differentiated services can be implemented for non-pipelined traffic and packets from elastic applications can be assigned to DiffServ classes based on any policy as possibly deployed on conventional IP networks.

Alternatively, a SVP with given allocated capacity can be setup for each application. Since packet transmission is regulated by TCP flow control mechanism, the application is not necessarily able to exploit all the reserved bandwidth. In fact, TCP throughput is limited by the transmission window size over the round trip time packets spend traveling from source to destination and acknowledgments back. At the beginning the transmission window size is small; in a first startup phase the transmission window increases exponentially, and then its size is incremented linearly up to its maximum in order to achieve a transfer rate as high as possible. Consequently, if the maximum size of the transmission window is not large enough compared to the round trip time, part of the allocated bandwidth might not be used. On the other hand, when the transmission rate equals the bandwidth allocated to the SVP, packets exceeding the reservation are, for example, sent as best-effort traffic. Consequently, if the network is congested they can be discarded, thus triggering a reduction of the transmission window. Depending on the amount of non-pipelined traffic in the network and the network diameter, the transfer rate experienced by an application might be either smaller or larger than the amount of reserved bandwidth.

A different transport protocol could be used in order to take full advantage of pipeline forwarding. Deployment of UDP is not an option as most elastic applications deployed over the Internet require a reliable service. However, only error control and flow control capability is required by a transport protocol operating on a pipeline forwarding network. In fact, as long as the bandwidth reserved to an SVP is not exceeded packets, are not affected by congestion, i.e., they do not get dropped, and the transport protocol needs only to deal with transmission errors and packet loss within end-system. For example, flow control could be rate based and driven by the bandwidth reserved to the corresponding SVP. Alternatively, such a protocol could be implemented as a variant of TCP

[2] Notice that the same does not apply for most streaming applications that require a minimum service in order to be usable. Consider, for example, video broadcasting: if a minimum bandwidth is not obtained, the received video stream cannot be properly viewed.

and possibly try to never reduce rate below reserved bandwidth and exploit transmission of non-pipelined packets in order to achieve higher transmission rate. The design, implementation, and performance evaluation of such a transport protocol are outside the scope of this paper and are left for future investigation.

4.2 Control Architecture

In general, in order to achieve deterministic quality of service, applications need to reserve resources. Although pipeline forwarding SVPs can be used to provision bandwidth on backbones, in order to fully benefit from pipeline forwarding transmission capacity within TFs should be reserved to single applications prior them starting generating packets. The reminder of this section discusses possible ways of introducing signaling in the communication architecture of a few widespread applications. The aim is not to provide detailed guidelines for the implementation of a triple play architecture based on pipeline forwarding, but rather to give a sense of how such a novel paradigm can be easily accommodated.

The one pass with advertisement reservation model adopted by the Resource reSerVation Protocol (RSVP) [7] is well suited to implementing the steps required to find a schedule through a pipeline forwarding network as outlined in [1]. Consequently, any reservation model and signaling solution previously elaborated in the context of the Integrated Services (IntServ) [7] architecture can be simply adopted for pipeline forwarding networks.

Services based on video streaming, such as video broadcasting and video on demand, are usually centered around a video server distributing content. The Real-Time Streaming Protocol (RTSP) is the standard protocol for a client to interact with the video server to choose and control the content it intends to receive. When a server receives a request to start streaming a given content, knowing the profile of the corresponding packet and the needed service, it can issue a signaling message (e.g., an RSVP PATH message) to gather information on resource availability across the network. The message crosses the network along the path the video stream will later follow probing for available resources in each TF. Upon reception of the signaling message, the client can select available transmission capacity in a suitable number of TFs and generates a reservation message (e.g., an RSVP RESV message) towards the server, which specifies the amount of resources to be allocated to the video stream in each TF. Notice that if the server or, as it is most likely in initial introductory phases, the client are not capable of performing the reservation, an access node to the pipeline forwarding network can infer the characteristics and QoS requirements of a video flow by intercepting RTSP messages and participate in the signaling procedure on behalf of the end-system (see the network architecture depicted in Fig. 1).

Although other solutions exist, the Session Invitation Protocol (SIP) is currently the preferred standard option for audio and video conferencing over IP networks. Although Calling and called parties initially get in touch through an infrastructure of servers (i.e., SIP proxies, location services, etc.) they eventually exchange messages to negotiate the parameters of their media session(s) — such as the audio codec to be used for a phone call. Pipeline forwarding aware end-systems can initiate resource reservation procedures once they have agreed on the characteristics of the media sessions, thus knowing the profile of the corresponding traffic and the required QoS. When pipeline forwarding unaware end-systems are deployed, access nodes can intercept SIP messages to infer the characteristics and QoS requirements of media sessions and carry out signaling procedures on behalf of the end-systems.

Today audio and video conferencing solutions based on a peer-to-peer overlay to forward media streams, such as Skype, are gaining significant importance and diffusion as they are capable of working around firewalls and network address translation (NAT) functionalities. In general, when a peer-to-peer overlay is deployed, resource reservation might follow or accompany the procedures that lead to establishing peering between nodes. Pipeline forwarding might be particularly beneficial when the peer-to-peer paradigm is exploited. In fact, media might travel along a sub-optimal path through a number of relay nodes. Given the delay introduced by relay nodes and the non-minimal length of the path, end-to-end delay becomes an even more critical issue and deploying pipeline forwarding might be the only viable solution in certain operating conditions to keep it within acceptable bounds.

5. REFERENCES

[1] C-S. Li, Y. Ofek, A. Segall, K. Sohraby, “Pseudo-isochronous cell forwarding,” *Computer Networks and ISDN Systems*, 30:2359-2372, 1998.

[2] M. Baldi, Y. Ofek, “Fractional Lambda Switching,” *Proc. of IEEE ICC 2002*, New York, vol.5, 2692 – 2696.

[3] 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.

[4] M. Baldi, G. Marchetto, G. Galante, F. Risso, R. Scopigno, F. Stirano, “Time Driven Priority Router Implementation and First Experiments,” *IEEE ICC 2006,* Istanbul (Turkey), June 2006.

[5] D. Agrawal, M. Baldi, M. Corra, G. Fontana, T. T. Huong, G. Marchetto, V. T. Nguyen, Y. Ofek, D. Severina, O. Zadedyurina, “Ultra Scalable UTC-based Pipeline Forwarding Switch for Streaming IP Traffic,” *IEEE INFOCOM 2006*, Barcelona (Spain), Apr. 2006.

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

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

[8] K. Nichols, V. Jacobson, L. Zhang, “A Two-bit Differentiated Services Architecture for the Internet,” *IETF Std. RFC 2638*, July 1999.

[9] Communicating European Research (CER 2005) International Conference, Brussels, Belgium, Nov. 2005, http://europa.eu.int/comm/research/conferences/2005/cer2005/index_en.html.