Available online at www.sciencedirect.com
ScienceDirect
Computer Networks 51 (2007) 4092–4111
Computer Networks
www.elsevier.com/locate/comnet

A scalable solution for engineering streaming traffic in the future Internet ☆

Mario Baldi [a], Guido Marchetto [a,*], Yoram Ofek [b]

[a] Dipartimento di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy
[b] Dipartimento di Informatica e Telecomunicazioni, Università di Trento, Via Sommarive 14, 38050 Povo (Trento), Italy

Received 23 June 2006; received in revised form 1 February 2007; accepted 24 April 2007
Available online 13 May 2007

Responsible Editor: J.C. de Oliveira

Abstract

As traffic on the Internet continues to grow exponentially, there is a real need to solve scalability and traffic engineering simultaneously – specifically, without using over-provisioning in order to accommodate streaming media traffic. One of the threats to the current operation stability of the Internet comes from UDP-based streaming media applications, such as Skype (which is currently doubling every 6-month) today and video services in the near future. This paper shows how the Internet can benefit from pipeline forwarding in order to: (i) construct ultra-scalable IP switches, (ii) provide predictable quality of service for UDP-based streaming applications, while (iii) preserving elastic TCP-based traffic as is, i.e., without affecting any existing best-effort applications.
© 2007 Elsevier B.V. All rights reserved.

Keywords: UDP-based streaming; Quality of service; Scalable IP networking; High performance IP switching; Traffic engineering; Time-driven switching; Time-driven priority

1. Introduction

The Internet has been growing steadily in the past few years. However, services so far deployed over the Internet are nothing compared to the ones that can still be deployed – i.e., we are only at the very beginning. Thus, one may envision that service-wise and traffic-wise the growth of the Internet applications is still to come. One likely scenario is that the future Internet will be dominated by applications such as video on demand, high quality videoconferencing, distributed gaming, virtual reality, and many more. The traffic generated by these applications either is by nature a streaming one, or can be easily turned into it and effectively handled as such.

This paper discusses the deployment within the future Internet of *pipeline forwarding*, a solution that is particularly suitable to carry streaming media applications over IP networks since it enables:

☆ This work was supported in part by the European Union under the IP-FLOW contract no. MC-EXC 002807.
* Corresponding author. Tel.: +39 011 2276 616; fax: +39 011 564 7099.
E-mail address: guido.marchetto@polito.it (G. Marchetto).

1389-1286/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2007.04.019

1. high scalability of network switches,
2. quality of service guarantees (deterministic delay and jitter, no loss) for (UDP-based) constant bit rate (CBR) and variable bit rate (VBR) streaming applications – if needed, while
3. preserving elastic, TCP-based traffic as is, i.e., without affecting any existing applications based on a best-effort service.

While the properties, advantages over alternative solutions, and limitations of pipeline forwarding were extensively addressed by previous work [1–7], the aim of this paper is in fact to discuss how pipeline forwarding can be introduced in the Internet to fill existing gaps, while building on existing protocols and solutions. An important feature of the presented solution is that IP packets are transported unchanged as a whole end-to-end. Namely, no change can be seen by observing packets flowing on a link of the Internet as only conventional IP packets, possibly encapsulated into Ethernet or PPP frames travel through the network.

In essence what is presented in this work are traffic management techniques – possibly applicable in the context of traffic engineering – for controlling (UDP-based) streaming traffic, while elastic (TCP-based) traffic can be kept unchanged and unaffected. 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 over-provisioning do not scale. In fact, conventional techniques for guaranteeing service performance in packet networks [8], possibly adopted in the context of the Integrated Services model [9] or ATM networks, 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 [10] for a detailed discussion). As a consequence, on today's Internet, where traffic management is not applied ubiquitously, some applications that should in principle use UDP (e.g., video streaming) actually deploy TCP to ensure an acceptable level of loss (no-loss, in fact, as guaranteed by TCP) at the expenses of higher delay and lower utilization. With the approach proposed in this paper such applications could use UDP as in this case the only loss will be the result of bit errors (say, 10^{-9}) that is certainly acceptable (although possibly not strictly required) in interactive and non-interactive media applications. It is worthwhile noting that the traffic management techniques presented in this paper can be profitably applied also to TCP traffic that requires minimum service guarantees, e.g., controlled end-to-end delay.

Although synergistic with existing solutions, such as the Integrated Services [9] and Differentiated Services [11] models, pipeline forwarding is inherently innovative in its operating principles, which are discussed in Section 2.1. The powerful properties of pipeline forwarding stem directly from packet switches sharing a common time reference. This might raise questions of novelty with respect to circuit switching (i.e., SONET/SDH switching). However, pipeline forwarding is fundamentally different from SONET/SDH since it is based on utilizing of UTC (Universal Coordinated Time), which provides phase synchronization, to control the forwarding of data units in the network, while transferring packets as one unit end-to-end (note that SONET/SDH uses only frequency synchronization). Section 1.A of [4] provides a detailed comparison with other solutions that deploy synchronization ranging from SONET/SDH to various experimental and research solutions, some of which in the context of optical networks.

The paper is organized as follows. Section 2 focuses on pipeline forwarding, the technology underlying the presented solution, by discussing its operating principles and properties. Various issues related to resource reservation in networks implementing pipeline forwarding are discussed in Section 3. Section 4 elaborates on network architectures for taking advantage of pipeline forwarding and gradually introducing it into the existing IP networks, specifically the Internet. Section 5 shows how the presented solution fits to technologies developed for the Internet, such as IntServ, DiffServ, MPLS, and GMPLS. In fact, it shows how on the one hand, they can be leveraged on for the implementation of network solutions based on pipeline forwarding and, on the other hand, pipeline forwarding can be instrumental in reaching some of the objectives for which such technologies were originally designed. Section 6 describes prototyping and ongoing experimentation activities on the presented solution and concludes the paper by highlighting the contribution of the reported work and by outlining future work.

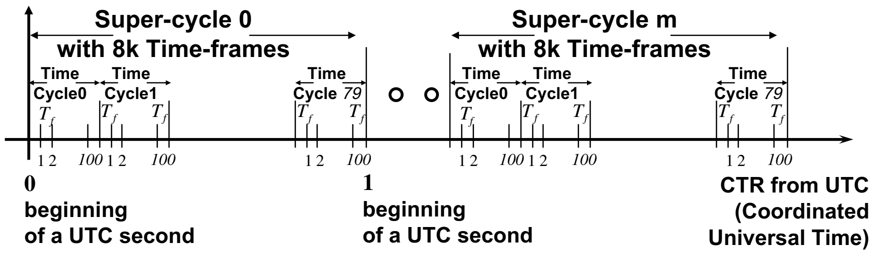


Fig. 1. Common time reference structure.

2. UTC-based pipeline forwarding

This section has two main parts: (i) UTC-based (coordinated universal time) pipeline forwarding is described in Sections 2.1–2.3, and (ii) Section 2.4 discusses how non-pipeline forwarding IP traffic (e.g., best-effort) is preserved unchanged. Implementing UTC-based pipeline forwarding for real-time packet scheduling requires packet switches to be synchronized with a *common time reference* (CTR) [12]. CTR is globally available via UTC (coordinated universal time). UTC can be received from a time-distribution system such as the global positioning system (GPS) and, in the near future, from Galileo.

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.[1] 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, in Fig. 1, the 125-μs time frame duration T_f is obtained by dividing the UTC second by 8000; sequences of 100 time frames are grouped into one time cycle, and runs of 80 time cycles 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*, 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, involves solving a scheduling problem, and must be large enough to satisfy (i). Considering that in real systems the propagation delay between nodes n and $n+1$, and the processing and switching time of node $n+1$ will not be null, rule (i) implies that $d_p \geqslant 2$ in node $n+1$, i.e., the delay experienced by a packet traveling from node n to node $n+1$ varies between 2 and the maximum forwarding delay introduced by a node D_p. The end-to-end delay $d(h)$ (excluding propagation delay and assuming processing and switching time shorter than a TF) on a path encompassing h links, i.e., the time elapsed from transmission on link 1 to transmission on link h (see Fig. 2), ranges between the following values depending on the forwarding delay chosen by each node:

$$d(h) = 2 \cdot (h-1) \cdot T_\text{f} + \Delta d,$$
$$d(h) = (h-1) \cdot D_p \cdot T_\text{f} + \Delta d,$$

where $\Delta d = [0, T_\text{f}]$ is the end-to-end jitter resulting from packets being asynchronously transmitted within a TF.

As exemplified in Fig. 3, which depicts the journey of a packet from node A to node D along three UTC-based switches, the forwarding delay may have different values for different nodes, due to dif-

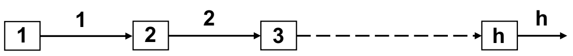


Fig. 2. Path encompassing h links.

[1] For the sake of brevity we sometimes simply say that a TF is, partially or totally, reserved.

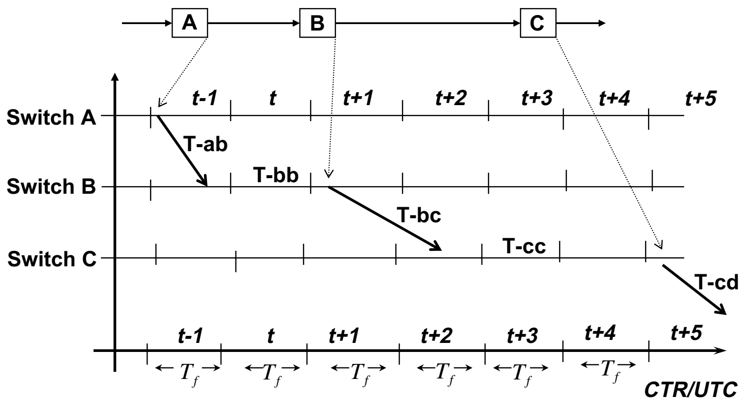


Fig. 3. Pipeline forwarding operation.

ferent propagation delays on different links (e.g., T_{ab}, T_{bc}, and T_{cd}), and different packet processing times in heterogeneous nodes (e.g., T_{bb} and T_{cc}). Moreover, two variants of the basic pipeline forwarding operation are possible. When node n deploys *immediate forwarding*, the forwarding delay has the same value for all the packets received by node n on input link i and it is usually chosen as the minimum value satisfying (i). When implementing *non-immediate forwarding*, node n may use different forwarding delays for packets belonging to different flows.

The periodic scheduling within each node results in a *periodic packet forwarding* across the network, which is also referred to as *pipeline 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.

PF operation requires both propagation delay between adjacent nodes and processing and switching delay within nodes to be known in advance in order to properly perform scheduling. Processing and switching delays can be upper bounded at a known value by proper network node design and implementation; when UTC is available accurate delay measurements are simple. Therefore, neighboring nodes can measure the propagation delay on the link between them and use such information, together with an upper bound on their processing and switching delay, during the distributed scheduling procedure. This can raise an issue with interdomain links, i.e., links interconnecting routers belonging to different domains (e.g. Autonomous Systems). As today it is accepted that such routers exchange routing information in a "safe" and controlled way, an analogous solution, i.e., a "safe" interdomain delay measurement protocol, can be envisioned for deployment with pipeline forwarding. In case network administrators will not agree on border routers performing delay measurements on interdomain links, each domain may have to perform its own independent scheduling operation and packets could be rescheduled at the ingress of each domain, similarly to what described in Section 4.2 for traffic traversing an asynchronous domain on its way between two pipeline forwarding domains.

If scheduling is to be performed in a centralized way, propagation and delay information can be exchanged by means of routing protocols together with other topological information needed for routing. This is possible both within and outside the same network domain – several protocols have been defined for both intra and inter domain routing. However, if interdomain communications are not possible for particular policies and agreements, each domain may have to perform its own independent centralized scheduling operation.

Notice that buffering time just fills the gap between the time difference between forwarding TFs in subsequent nodes and the sum of propagation, processing, and switching delay. After a packet has been received (which follows propagation), processed, and switches, it is buffered until its transmission takes place during its forwarding TF. Consequently, buffering time is automatically adjusted and does not need to be explicitly known, i.e., it is implicitly determined by the beginning of the forwarding TF, which is known (derived from UTC).

In pipeline forwarding, a synchronous virtual pipe (SVP) is a pre-defined 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 bits allocated in every TC for the SVP. For example, for a 10 ms TC, if 20,000 bits are allocated during each of 2 TFs, the SVP capacity is 4 Mb/s.

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

2.2. Time-driven IP switching

Time-driven switching (TDS) was proposed to realize sub-lambda or fractional lambda switching (FλS) in highly scalable dynamic optical networking [2–7], which requires minimum (possibly optical) buffers. In this context, TDS has the same general objectives as optical burst switching and optical packet switching: realizing all-optical networks with high wavelength utilization. TFs can be viewed as virtual containers for multiple IP packets that are switched at every TDS switch based on and coordinated by the UTC signal. In the context of optical networks, SVPs are called fractional lambda pipes (FλPs). Since non-immediate forwarding requires buffering packets in network switches, with current optical technologies its extensive deployment in all-optical FλS switches would be expensive. However, limited non-immediate forwarding capability with support for small forwarding delays, i.e., small values of D_p, could be included in switches to decrease blocking probability. Although this will require optical buffering, its requirements in terms of size and access are far simpler than the one posed by optical bust switching and optical packet switching.

In TDS all packets in the same TF are switched in the same way. Consequently, header processing is not required, which results in low complexity (hence high scalability) and enables optical implementation. The TF is the basic SVP capacity allocation unit; hence, the allocation granularity depends on the number of TFs per time cycle. For example, with a 10 Gb/s optical channel and 1000 TFs in each time cycle, the minimum FλP capacity (obtained by allocating one TF in every time cycle) is 10 Mb/s.

Scheduling through a switching fabric is based on a pre-defined schedule, which enables the implementation of a simple controller. Moreover, low complexity switching fabric architectures, such as Banyan, can be deployed notwithstanding their blocking features, thus further enhancing scalability. In fact, blocking can be avoided during schedule computation by avoiding conflicting input/output connections during the same TF. Previous results [4] show that (especially if multiple wavelength division multiplexing channels are deployed on optical links between fractional λ switches) high link utilization can be achieved with negligible blocking using a Banyan network without speedup.

2.3. Time-driven priority

TDS is suitable for very high speed (optical) backbones, where traffic can be organized in large capacity SVPs handled by high performance switches. More flexibility could however be required at the edge of the network as offered, for example, by conventional IP destination-address-based routing. *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 above-mentioned flexibility. Packets entering a switch from the same input port during the same TF can be sent out from different output ports, according to the rules that drive IP packet routing. 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.

2.4. 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 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 is 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.

The hybrid switch architecture shown in Fig. 4 trades some of the simplicity of TDS switches for higher flexibility (as provided by enabling statistical multiplexing at each node), while maintaining the same level of scalability. Best-effort packets are extracted 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 module performing conventional IP routing and "injected" in the flow of packets exiting the proper interface whenever it is idle. The complexity of the hybrid node is higher than the one of a plain TDS node and all-optical implementation might be impractical (comparable to optical burst switching and optical packet switching). However, scalability is not compromised as SVP 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. Resource reservation

In order to ensure deterministic service, SVP setup must involve a scheduling and resource reservation procedure which is the object of this section.

3.1. Scheduling problem

Given the route of a SVP in the network, the identity of the TFs reserved on a link is bound to the identity of TFs reserved on the previous link. For example, if TDP immediate forwarding is used, once the identity of a TF to be reserved on a link is fixed, the identity of the corresponding TFs on all the other links of the path is uniquely determined; this is exemplified in Fig. 3. As a consequence, TFs partially or completely allocated on a link impose constraints on reservations to be performed on adjacent nodes. Reserving resources for a SVP requires solving a *scheduling* problem to find a feasible sequence of TFs, called *schedule*, on links on the route from source to destination. When a new SVP has to be established and resources are being

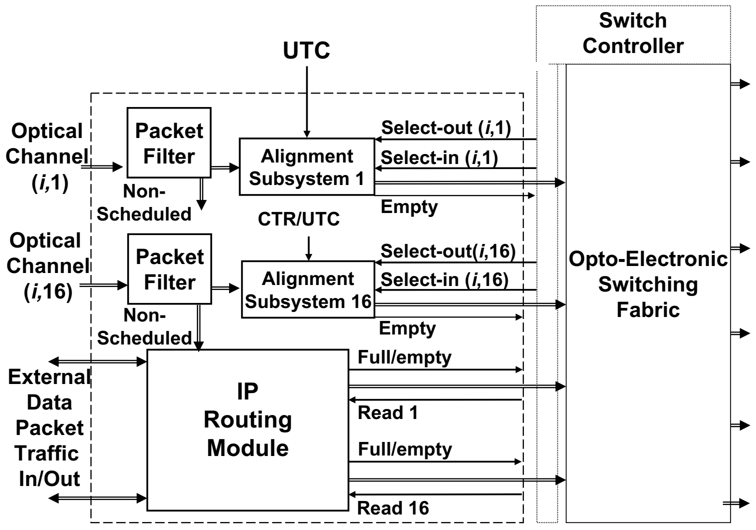


Fig. 4. Hybrid IP routing and FλS system.

looked for, the reservation can be denied even though enough capacity is available on all the links on that SVP's path. This happens if the identity of the TFs on the various links does not satisfy the requirements imposed by UTC-based forwarding. The session is said to be *unschedulable*. Unschedulability has no meaning in asynchronous packet networks, thus, resource allocation is based on various heuristic procedures for *admission control* that consider only the amount, not the timing, of resource availability. Asynchronous admission control maintains link and network utilization well below 100%. Unschedulability in UTC-based networks can be compared to *blocking* in digital circuit switching. A blocking in digital circuit switching can happen when there is no available switching path from an idle inlet to an idle outlet. In this case, a call request from the idle input port to the idle output port cannot be accepted, even thought there are transmission resources on the corresponding links, i.e., the call is *blocked*. Thus, *SVP blocking probability* is defined as the probability for a resource reservation on a UTC-based network to be denied because of unschedulability. The blocking probability depends on many parameters, such as the size of the packet and the number of bits that can be sent during a TF.

The SVP blocking probability can be reduced by using *non-immediate* forwarding. In fact, contrary to when immediate forwarding is performed, since each node can delay a packet for up to D_p TFs, the TF to be reserved by each node on its outgoing link is not uniquely determined by the TF reserved on the incoming link: it can be chosen from a set of $(D_p - 1)$ TFs. Note that when D_p is equal to the time cycle size k, scheduling is always possible when resources are available, i.e., there is no blocking and full link utilization can be achieved under any condition. Clearly, the decrease in blocking probability offered by non-immediate forwarding with a large D_p value is traded for an increase in the end-to-end delay, as shown by the end-to-end delay analysis in Section 2.1.

The blocking problem has been studied extensively, both analytically [12 (Section 4.2),7] and by simulation [4,5,15] in a large set of scenarios featuring varying values for the network parameters (e.g., link capacity, time frame duration, and packet size, SVP capacity). Furthermore, a complexity study regarding the scheduling problem, including when non-immediate forwarding is performed and although the solution space grows exponentially there are polynomial algorithms that can find a schedule whenever such schedule exists.

3.2. Availability vector

Scheduling can be performed in each node executing a *distributed scheduling algorithm* derived from that presented in [12] for ATM switching. Scheduling and resource reservations are based on a data structure called an *availability vector*, which has a size k-one element for each TF in the time cycle. An availability vector, the *link availability vector*, is associated with each link of the network and it contains the amount of bits that have not yet been reserved during each TF. When the SVP ingress node processes a SVP request event, it builds up an availability vector intended to contain the amount of bits that can be reserved on the whole path of the SVP (*SVP availability vector*) per each TF of the time cycle. Resource allocation will be performed by selecting the TFs to be reserved for the SVP based on the information carried within the SVP availability vector. The procedure is a little bit different between the immediate and non-immediate forwarding. A brief explanation of the procedure related to the immediate forwarding technique is presented here. For further information see [12].

The SVP availability vector is initialized to the link availability vector of the outgoing link from the source. Then, the SVP availability vector is passed to the next node. Here it is cyclically shifted to the right a number of times equivalent to the delay, measured in TFs, experienced by a packet traveling from the previous node's output buffer to the current node's output buffer, plus 1 since the transmission takes place in the TF following the arrival. The resulting availability vector is combined with the availability vector of the next link on the path, and so on until the availability vector reaches the SVP egress point. Given the current SVP availability vector, AV_S, and the link availability vector, AV_l, associated with the next link l, the ith element of the combined SVP availability vector is

$$AV_S(i) = \min\{AV_S(i), AV_l(i)\}, \quad \forall i : 0 < i < (k-1).$$

Fig. 5 shows a sample computation of a SVP availability vector; the labels on the links represent the delay, in TFs, between (the output buffer of) each pair of nodes. After the shifting and combining have been performed along the whole path to the destination, if

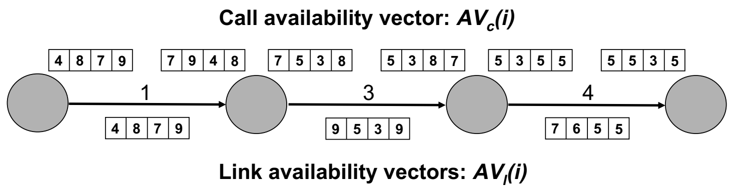

Call availability vector: $AV_c(i)$
4 8 7 9
7 9 4 8
7 5 3 8
5 3 8 7
5 3 5 5
5 5 3 5
1
3
4
4 8 7 9
9 5 3 9
7 6 5 5
Link availability vectors: $AV_l(i)$

Fig. 5. Computation of a SVP availability vector on a TDP network.

a SVP availability vector contains enough TFs with sufficient space, the egress node chooses the required number of TFs. Then, the information regarding the chosen TFs is sent back to the ingress node and resources are reserved on each node along the path by updating the link availability vectors according to the chosen TFs. This step may not be successful since many SVP setups can occur concurrently. Thus, it is possible that by the time the allocation is done for a SVP, the resources are not available anymore. In this case, the setup procedure for this SVP is *aborted*. Following an abort, a cancellation message is forwarded along the path to the egress node in order to free the resources already allocated for this SVP. Another message is also sent to the ingress node so it can initiate another setup massage.

The set of TFs chosen by the egress node is called a *schedule*. The choice of the schedule when there is more than one possibility affects the utilization achievable on the network, and thus, the blocking probability. However, analysis of the implications of schedule choice is outside the scope of this work and left for further study.

Similarly to what discussed in Section 2.1 regarding delay information, network nodes have to exchange, possibly across multiple domains, availability vectors in order to perform end-to-end scheduling and resource reservation. Specifically, if scheduling is performed in a distributed fashion, link availability vectors are needed only locally, and SVP availability vectors can be transmitted within SVP setup signaling messages (e.g., PATH messages if RSVP is deployed for SVP setup). If scheduling is centralized, the node performing scheduling must gather link availability vectors across the network; this could be done as for routing information, i.e., link availability vectors can be carried by routing protocols. If, due to inter domain policies, this not possible across domain boundaries, each domain may have to perform its own independent scheduling operation. This is also necessary when different PF domains are connected through an asynchronous network. In both cases, traffic must be rescheduled, i.e., time shaped, at the ingress of the next domain by an SVP interface, as discussed in Section 4.2.

3.3. SVP setup in a TDS network

When a SVP has to be created in a TDS network, some aspects have to be taken into account. First of all, in the context of TDS TF state can be represented by a bit; in fact, according to the TDS principles, TF are the basic allocation unit, i.e., a TF can be either allocated (to a single SVP) or available. Therefore, also the SVP availability vector and the link availability vector simply contain the bit map of the TFs that have not yet been reserved and the bit map of the TFs that can be reserved for the SVP respectively. Furthermore, these vectors are combined by bit-by-bit logical AND operations as shown in Fig. 6.

The switching fabric deployed in TDS switches is another potential cause for a SVP setup to be impossible even though enough transmission capacity is available.[2] A blocking switching fabric might be unable to provide a switching path from an idle inlet to an idle outlet due to multiple input/output paths contending for internal switching resources. In this case, a SVP cannot be accepted, i.e., it is *blocked*, even thought there are transmission resources on the corresponding channels.

Deployment of blocking switching fabrics (e.g., Banyan) in TDS switches restricts the solution space of the SVP scheduling problem to the TFs during which at least one connection is possible from one of the input link to the output link on the path of the SVP. However, the suitability of Banyan switching fabrics for the implementation of TDS switches is confirmed by the results reported in [4].

[2] Although in this work blocking switching fabrics are being considered only in the context of TDS switches where scalability is a major design objective, they might be deployed in the implementation of TDP routers as well. In such a case, the considerations made in this section apply.

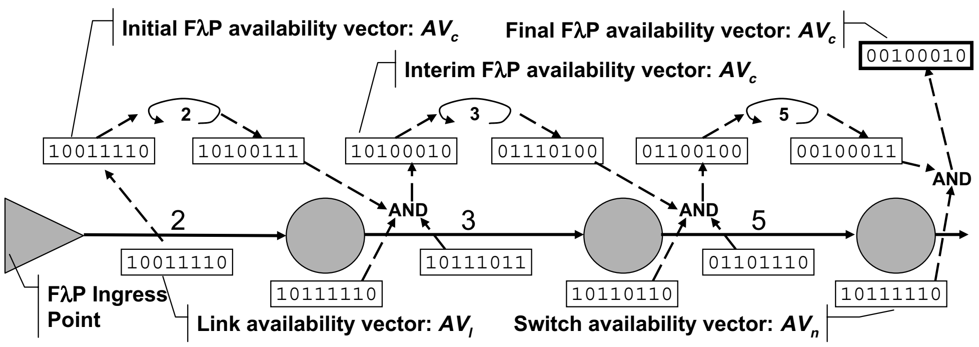

Initial FλP availability vector: AV_c
Final FλP availability vector: AV_c
Interim FλP availability vector: AV_c
00100010
2
3
5
10011110
10100111
10100010
01110100
01100100
00100011
AND
AND
AND
2
3
5
10011110
10111011
01101110
10111110
10110110
10111110
FλP Ingress Point
Link availability vector: AV_l
Switch availability vector: AV_n

Fig. 6. Computation of a SVP availability vector on a TDS network.

Another data structure, called *switch availability vector*, is needed and is associated with each input/output pair of a network node. It contains the bit map of the TFs during which a connection can be established between the input/output pair, given the existing input/output connections through the switching fabric during each TF. Switch availability vectors are combined with link availability vectors in the scheduling procedure, as shown in Fig. 6, and updated when resources are reserved to a SVP.

4. Deployment issues

4.1. 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.

Fig. 7 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 flows (see Section 4.6).

Fig. 7 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 perfor-

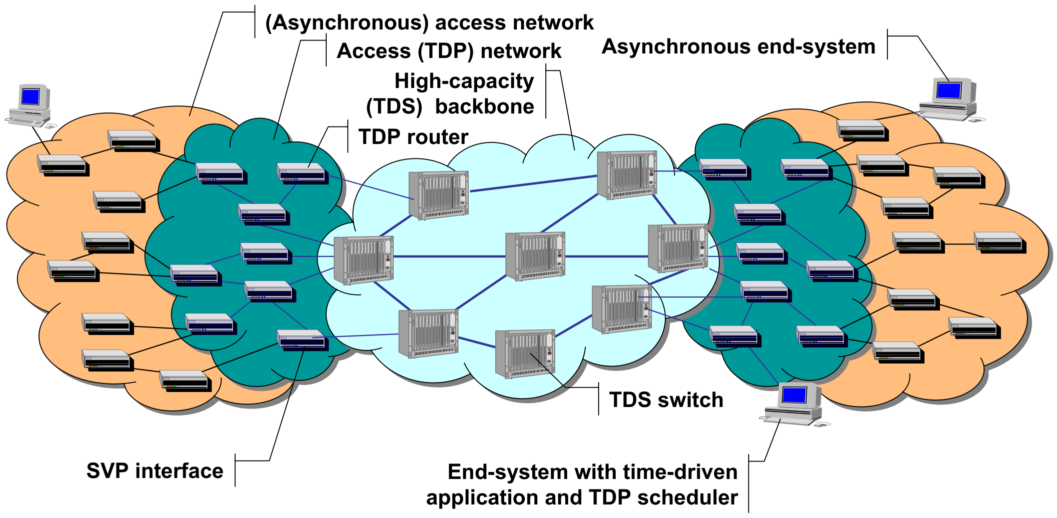

(Asynchronous) access network
Access (TDP) network
Asynchronous end-system
High-capacity (TDS) backbone
TDP router
TDS switch
SVP interface
End-system with time-driven application and TDP scheduler

Fig. 7. Network architecture for the deployment of pipeline forwarding.

mance at low cost, while the latter are being used where scalability is not a primary concern, but their flexibility is particularly beneficial. When packets reach the TDS backbone, they enter a SVP routed to an egress point suitable for reaching their destination, i.e., at resource reservation time scheduling must be performed seamlessly in TDP routers at the backbone edges and TDS switches in the backbone core.

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 [13] 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.

4.2. Scenarios for incremental introduction

Deploying UTC-based pipeline forwarding – and enjoying its benefits – does not require to immediately roll out a large number of TDP routers and/or TDS switches. A possible incremental deployment path could start from implementing a backbone of TDS switches in the network core. This would enable taking advantage of the low per-bit cost of TDS equipment, while keeping the start-up investment in network infrastructure limited.

As the TDS core is expanded towards the edges of the network, clouds of TDP routers can be deployed where very high port density and switching capacity are not fundamental. Moreover, when multiple pipeline forwarding clouds (whether including TDP routers or TDS switches, or both)

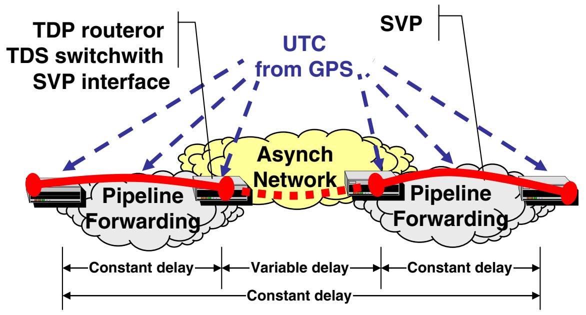


Fig. 8. Pipeline forwarding clouds interconnected through asynchronous networks.

are being deployed, SVPs can be set up across them, even though they cross asynchronous portions of the network, as shown in Fig. 8. During SVP setup, a signaling request containing the SVP availability vector is forwarded through the asynchronous domain to the first PF node at boundary of the next PF domain. The asynchronous portion of the network will not be exempt from congestion, i.e., packets might experience loss and variable delays. The ingress node to the PF domain maps the SVP availability vector with its link availability vector(s) according to an estimate of the maximum delay (or a pre-defined percentile delay) introduced by the asynchronous domain. A SVP interface installed on edge TDP routers or TDS switches re-shapes the packet flow by compensating variable delays. This is done by classifying (micro or macro) flows that are to enter an SVP and then transmitting the corresponding packets during the forwarding TFs defined at resource reservation time for their SVP. Classification of traffic flows is based on the information provided during the signaling procedure, e.g., an MPLS label, a FEC or, more generally in a non-MPLS context, an RSVP filter spec [9]. The end-to-end delay through the SVP will not be as short as it would be if pipeline forwarding was performed across the whole network and packets might get lost. However, especially if the backbone is lightly loaded (today this is the case for many large backbones), the performance might not be much worse than in the ideal case. Conversely, if the peripheral parts of the network where pipeline forwarding is deployed are heavily loaded, the performance (both in terms of quality of service and efficiency in the utilization of network resources) will be significantly higher than if traditional asynchronous packet switching were deployed.

An alternative incremental deployment path could start with deploying TDP routers in peripheral areas (e.g., metro area) where transmission capacity is not very abundant, i.e., network utilization is high, there might be occasional congestion, and very high performance and scalability are not needed. In this scenario, TDP routers offer deterministic quality of service and efficient usage of network resources – which results in users perceiving an improved overall service. Packets can travel among TDP clouds through an asynchronous backbone (see the network architecture depicted in Fig. 8). As mentioned above, SVPs can be set up from a TDP cloud to another one through the asynchronous backbone with the limitations and benefits discussed above. As the benefits of pipeline forwarding are experienced and the network infrastructure is updated, deployment of pipeline forwarding can expand towards the core and possibly lead to the deployment of TDS switches where many high-capacity links are available and high switching capacity is required.

The next step towards ubiquitous deployment of pipeline forwarding might consist in providing the CTR to end-systems. Although an end-system might still be directly connected to an asynchronous network (e.g., a wireless hot spot, a UMTS network, a DSL or cable modem access network) as shown in Fig. 9, CTR aware operating system and applications can time generated packets according to the TFs reserved to the SVP that carries them through the network. This has the potential to minimize delay and loss in the SVP interface at the ingress to the pipeline forwarding backbone. Even if delays across the asynchronous part of the network are variable, the end-to-end service might be improved; the lighter the load on the asynchronous network, the better such service approximates the one offered by end-to-end pipeline forwarding.

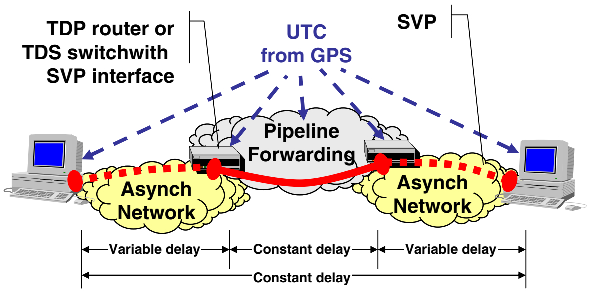


Fig. 9. End-systems with CTR connected through asynchronous access networks and pipeline forwarding backbone.

4.3. SVP persistency

As previously discussed, a SVP is a virtual channel between an ingress node and an egress node, that could be end-systems, TDP routers, or TDS switches; in the context of optical networks the SVP might be called FλP. The lifespan of established SVPs varies according to implemented policies. In general, at least two SVP persistency policies might be envisioned:

- *dynamic SVPs* are created for carrying a specific packet flow (e.g., a videoconference) and torn down when the connection ends, i.e., the corresponding application terminates;
- *provisioned SVPs* are created, possibly following a manual intervention, for carrying one or more packet flows and might survive the applications for which they were initially set up in order to possibly carry subsequent flows. (Provisioned SVPs may be viewed as analogous to *virtual pipes* (VPs) in asynchronous transfer mode – ATM.)

4.4. Traffic management

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 [9], quality of service (QoS) is guaranteed to single packet flows by reserving resources on a per-flow basis. This implies that network nodes have per-flow information and perform per-flow classification and packet handling so that each packet uses its own resources, thus getting the requested service. The IntServ model proven unable to scalable on large, high-capacity networks and consequently it is currently not deployed in any significant scenario. The reasons for IntServ's scalability issues are:

- Per-flow signaling with periodic refreshes to be processed by each node on the path of the flow.
- Per-flow state information must be kept in network nodes.
- Network nodes classify and handle packets on a per-flow basis. Per-flow packet classification and handling, i.e., queuing and scheduling, and flow isolation by means of sophisticated packet scheduling algorithms, such as weighted fair queuing, are in our view the main reason for IntServ lack of scalability.

Conversely, the architecture presented in this paper enables *guaranteed per-flow QoS* but does not have scalability issues because:

- Per-flow state information is not needed in network nodes – a limited amount of information is needed for the, possibly small, fractions of flows to which non-immediate forwarding is applied.
- Per-flow classification and handling is not required – with the exception of the possibly small, fractions of flows to which non-immediate forwarding is applied.
- A small complexity scheduling algorithm (comparable with round robin scheduling) operates on a small number of queues: *per-flow queuing is not required.*
- *Hierarchical resource reservation* is deployed where *contributing* SVPs possibly carrying a single packet flow at the edge are merged in *aggregate* (provisioned) SVPs, as shown in Fig. 10a. Consequently, nodes traversed by the larger SVP are completely unaware of single flows, i.e., they neither maintain per-flow state information, nor process per-flow signaling.

It is worth elaborating further on the latter point, i.e., pipeline forwarding guaranteeing per-flow QoS to aggregated flows. Since scheduling in each node is based on time, packets belonging to a SVP reach a particular node in well-defined moments (i.e., pre-defined TFs), even if they are mixed within TFs with packets of other SVPs, i.e., multiple SVPs have been merged in a larger one. In fact, once capacity during a set of TFs of an aggregate SVP is reserved for transmission of packets belonging to a contributing SVP, forwarding times are totally deterministic.

Resource reservation for a contributing SVP during the TFs of an aggregate SVP reservation can be performed independently by the ingress node to the aggregate SVP that acts as an access bandwidth broker [14] and, as shown in Fig. 11, nodes traversed by the aggregate SVP are not involved in set up of contributing flows. Obviously, all the above holds true when aggregate SVPs are merged to a larger aggregate SVP (see Fig. 10a), i.e., flow aggregation can be iterated while still guaranteeing per-flow QoS. This is pictorially shown in Fig. 10b. Let's consider SVP-1 that starts from S1 and has been reserved transmission capacity during TF 5 (the sender is supposed to be UTC-aware). R1 aggregates the SVP with two other SVPs, as shown in both Fig. 10a and b within an aggregate SVP named SVP-R1. As shown in Fig. 10b, TFs 4, 5, and 6 are reserved to the SVP-R1 and node R1 performs immediate forwarding on packets of the SVP-1[3] thus transmitting them towards R3 with forwarding delay equal to 1 TF, i.e., during TF 6. SVP-1 reaches R3 together with the other flows that enter SVP-R1. However, packets of SVP-1 will leave node R1 always in TF 6 (TF 5 + forwarding delay). Thus, supposing that also node R3 implements immediate forwarding with 1 TF forwarding delay, SVP-1 packets will be forwarded by R3 always in TF 7 even though node R3 does not distinguish them as belonging to SVP-1, but just as part of the packets of SVP-R1.

In summary, in order to achieve maximum scalability, traffic management and engineering on a pipeline forwarding network can be based on

- (provisioned and dynamic) SVPs possibly carrying a single packet flow at the edge of the network, where TDP routers are likely to be deployed, that are merged into
- aggregate, possibly provisioned, SVPs in the backbone, where, as discussed in Section 4.1, TDS switches are likely to be deployed.

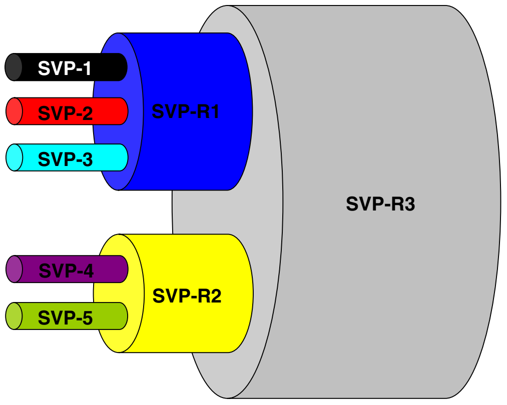
This model allows applications to reserve resources for each traffic flow and receive per-flow guaranteed service, while providing a high level of flexibility and scalability to the network.

4.5. Resource reservation origin

The above traffic management model is compatible with various alternatives for initiating resource reservation. A number of options – listed below in order of decreasing service quality and increasing generality – result from sensible combinations of end-systems being (i) UTC-aware or not (ii) QoS-capable or not.

1. UTC-aware end-systems are able to initiate SVP set up and transmit packets during TFs reserved to it. Consequently, especially if applications themselves are UTC-aware, end-to-end service

[3] For the sake of simplicity and without loss of generality, propagation delay is considered null.



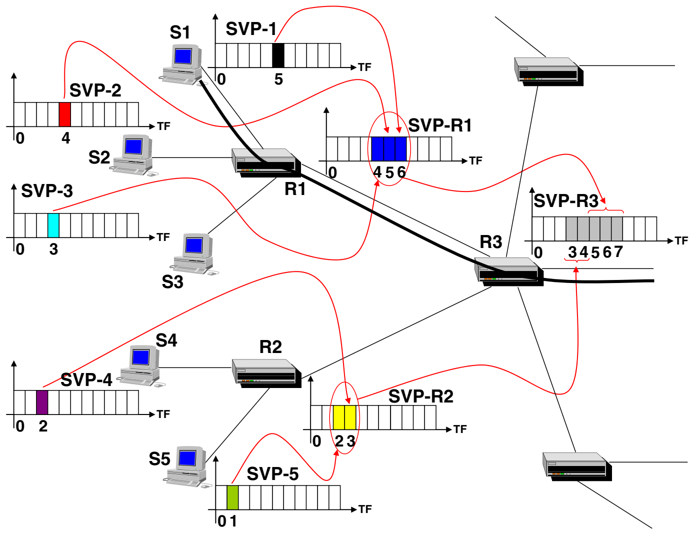


Fig. 10. Resource reservation model: (a) hierarchical resource reservation and (b) timely forwarding in aggregate SVPs.

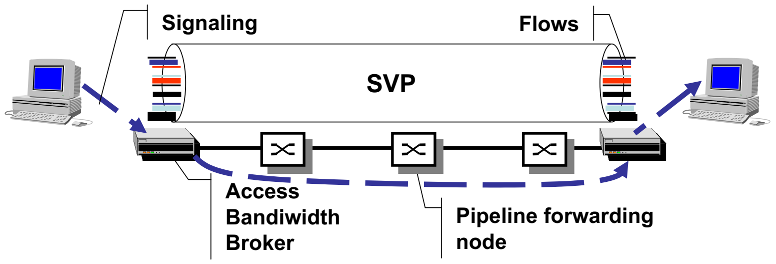


Fig. 11. Hierarchical bandwidth reservation and access bandwidth broker.

can be optimized, i.e., applications experience deterministic end-to-end delay and strictly bound jitter (e.g., 2 TFs).
2. QoS-capable end-systems, even though UTC-unaware, are able initiate a resource reservation procedure that can be properly processed by an access node to the pipeline forwarding network (see the TDP router or TDS switch with SVP interface in Fig. 12). As a result the access node (1) sets up a SVP with proper capacity, and (2)

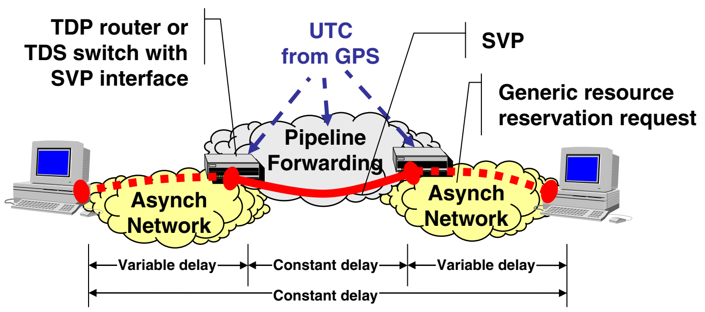


Fig. 12. Resource reservation across asynchronous and pipeline forwarding networks.

presets the SVP interface to forward through the SVP packets of the flow making the reservation. The latter requires (2.a) configuring a suitable filter to classify packets belonging to the flow and (2.b) associating to it the TFs reserved for the SVP. For example, an end-system supporting IntServ [9] is capable of setting up a reservation with the Resource reSerVation Protocol (RSVP). As discussed in Section 5, a variant of the same protocol can be used for setting up a SVP, thus simplifying end-to-end signaling. In general, the resource reservation performed by the station provides network nodes, and specifically the ingress node to the pipeline forwarding cloud, with a description of the traffic generated and the QoS ideally needed by the application. This information is used to drive the reservation of transmission capacity for the SVP, while if the application complies with the former packets will not accumulate at the SVP interface. Consequently, although the end-to-end service will not be optimal (as in case 1 above), it can approximate it. Actually, if proper QoS support is provided in the asynchronous parts of the network, the end-to-end service perceived by the applications might be basically the same as in case 1 above, the difference laying in the complexity and resource utilization efficiency of the QoS solution deployed in the asynchronous parts of the network.[4]

3. Today, virtually all end-systems deployed over the Internet are neither QoS-capable nor UTC-aware. In this case resource reservation is responsibility of the ingress node to the pipeline forwarding network. At least two approaches can be foreseen.
   - SVPs are provisioned and traffic associated to them (by proper configuration of the SVP interface of ingress nodes) as part of the usual traffic engineering effort by the network administrator.
   - Ingress nodes implement traffic classification algorithms and policies to identify and characterize traffic flows that should receive guaranteed service. SVPs with proper capacity are automatically set up, associated to corresponding flows, and torn down when the flow is deemed to have ceased.

[4] Nevertheless, given that resources might not be scarce and high-capacity network gear might not be required at the periphery, the network scenario presented in Fig. 11 with asynchronous packet switches and UTC-unaware, QoS-capable end-systems might be sensible and possibly represent the best trade-off.

4.6. SVP interface

The ingress node of a SVP is responsible for (i) setting it up by initiating the procedure to reserve transmission capacity on each traversed link during the respective forwarding TFs and (ii) *UTC shaping* the incoming flow thus ensuring that packets are transmitted during the reserved TFs and without exceeding the reserved capacity on the first link.

Assuming a signaling protocol is used by asynchronous end-systems, as in case 2 in the previous section, the SVP interface on the ingress node reserves transmission capacity during a number of TFs according to the traffic specification carried by the signaling packets. Once the SVP is set up, packets with reserved resources arrive asynchronously, i.e., at unpredictable times distributed across each TC. These packets are stored in a buffer until they can be transmitted using the reserved capacity during forwarding TFs – i.e., they are *UTC shaped*.

Consequently, the SVP interface is a cause of delay and possibly packet loss. Although a detailed analysis of the SVP interface contribution in terms of delay, jitter, and loss is outside the scope of this paper and left to future works, it is worth considering here at a high level what affects such contribution. Broadly speaking, the time spent by packets in the SVP interface buffer depends on (i) the burstiness of the flow, (ii) the distribution of the allocated TFs across the time cycle. The smaller the flow burstiness, the higher the reserved capacity (i.e., the number of reserved TFs), and the more evenly reserved TF are distributed across the time cycle, the smaller the delay packets experience in the SVP interface. Notice that the flow burstiness is due to both the source burstiness and the service received within the asynchronous network. Consequently, it can be reduced by shaping traffic at the source and by controlling jitter within the asynchronous network, e.g., by implementing proper packet scheduling algorithms or by keeping network utilization low (i.e., by performing over-provisioning).

A source of potentially large delay and loss is the difference in the frequency of the source clock with respect to UTC. If the data rate of the generated flow is larger than the nominal one, i.e., it exceeds its reservation, packets accumulate in the SVP interface buffer possibly overflowing it. This can be avoided in various ways; for example,

- a slightly larger capacity than the nominal flow rate is reserved to the SVP,[5] or
- when the SVP interface buffer exceeds a given threshold, extra packets are sent as best-effort traffic.

The delay introduced by the SVP interface is anyway not longer than the one any scheduler would introduce with a fully loaded output link. However, packets traversing a pipeline forwarding network experience such delay only at the SVP interface, while in an asynchronous network each fully loaded node potentially introduces such delay. The other nodes traversed by the SVP introduce a fixed forwarding delay determined at SVP setup and depending on (i) the node performance (specifically its maximum processing and switching time) and (ii) possibly an additional delay in case non-immediate forwarding is required in order to avoid call blocking. However, results in [15] and [4] show that both TDP and TDS can achieve very high utilization with immediate forwarding; hence, only a small fraction of flows possibly needs to be subjected to the extra non-immediate forwarding delay. Buffer size requirements for non-immediate forwarding have been studied in [12].

[5] Reserving a larger capacity to a SVP can be used as a general measure to reduce the delay introduced by the SVP interface, also due to flow burstiness.

5. Synergies with existing technologies and protocols

Pipeline forwarding enables a scalable multi-service network able to provide guaranteed QoS and support to efficient traffic engineering, while providing full statistical multiplexing of elastic traffic. Various technologies and protocols have been proposed to reach these goals, some of them having an important role in present-day networks due to their effectiveness in solving part of these issues, especially scalability and traffic engineering support. The main still open issue is obtaining a network architecture being able to achieve all the above mentioned goals. Pipeline forwarding has the potential to fulfill this task and can leverage on existing technologies for this purpose. This section describes possible synergies.

5.1. DiffServ

A large part of Internet traffic today is generated by TCP-based elastic applications (e.g., file transfer, e-mail, WWW) that do not require a guaranteed service in term of end-to-end delay and jitter. Such traffic benefits from statistical multiplexing. As discussed in Section 2.4, pipeline forwarding provides (UDP-based, real-time) streaming applications with guaranteed QoS, but it is completely transparent to any other kind of traffic that enters the network and not requiring this guaranteed service. Any unused portion of each TF could be used to forward flows according to a totally arbitrary policy. Besides enabling traditional best-effort service, *Differentiated Services (DiffServ)* [11] can be implemented. This allows network administrators to handle non-streaming traffic with the provisioning, dimensioning, and traffic engineering procedures currently deployed (and consequently well known).

Moreover, DiffServ takes a major role in current TDP router implementation. Firstly, the DiffServ field is deployed to distinguish packets that must be handled with pipeline forwarding from the others. In other words, TDP packets are seen as a specific differentiated class. Secondly, TF delineation is realized by changing the value of two bits of the

DiffServ field in each TF (see [8] for details), thus realizing a sort of alternating bit protocol that enables a TDP router on the receiving end of a link to devise the TF during which each packet has been transmitted.

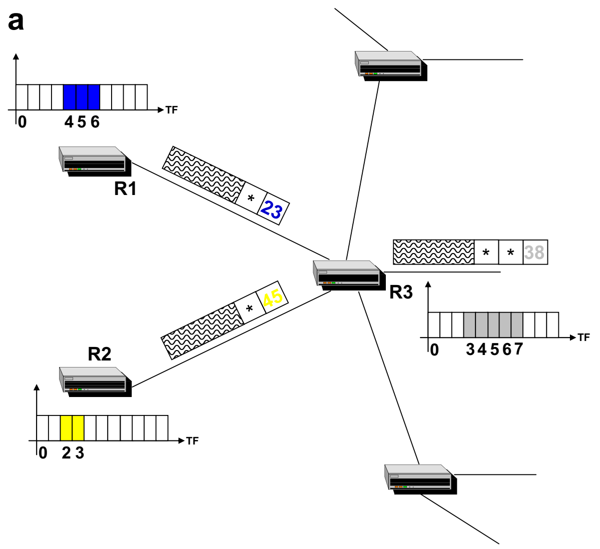
5.2. MPLS

Multi-protocol Label Switching (MPLS) [16] provides a scalable network and protocol architecture where hierarchical *Label Switched Paths* (LSPs) ensure that packets of a flow follow the same path. This fits to pipeline forwarding and its deployment scenarios described in Section 4, where packet delivery is based on hierarchical virtual channels (SVPs). The clear parallel between an LSP and a SVP is reflected in a mapping between the LSP label and the set of TFs reserved to a SVP, as exemplified by Fig. 13.

A TDP LSR (*Label Switch Router*) routes packets according to their MPLS label and schedules them according to TDP. Non-immediate forwarding can be implemented by modifying the *Incoming Label Map* (ILM) table; in particular, the forwarding TFs reserved to the SVP associated to each label must be included in each entry, as exemplified in Fig. 13b. Fig. 13 also shows a sample mapping of LSP hierarchy onto SVPs hierarchy (see Section 4.4). An aggregate LSP is associated to an aggregated SVP originating in R3. A new label is pushed on the label stack of packets entering the aggregate LSP; such packets are forwarded during the TFs reserved to the aggregate SVP. Note that although Fig. 13b shows a general case, when immediate forwarding is implemented for a particular LSP, the forwarding TF information is redundant and could be omitted from the ILM table.

A TDP LSR can implement the above mentioned alternating bit protocol to devise the forwarding TF for incoming packets using two bits of the DiffServ field. Nevertheless, the EXP within the MPLS shim header can alternatively be used. A discussion of the relative advantages and disadvantages of these two solutions is beyond the scope of this work.

The role of the label edge router (responsible in the MPLS architecture of labeling incoming packets) maps well on the role of an ingress node to a pipeline forwarding network. Specifically, the *FEC-to-NHLFE* (FTN) table can be modified to list the TFs reserved to a specific FEC (forwarding equivalence class). This information is used by the SVP interface on the TDP label edge router.



b

Label	NHLFE
23	Label: PUSH 38 Interface: s0 ... TF: 5, 6, 7, ...
45	Label: PUSH 38 Interface: s0 ... TF: 3, 4, ...
...	...

Fig. 13. Pipeline forwarding and MPLS: (a) label switching in R3 and (b) ILM in R3.

5.3. GMPLS

Generalized MPLS (GMPLS) [17] extends MPLS to provide a unifying control plane providing signaling and routing functionalities to devices that use different physical layers and forwarding planes, i.e., IP routers, ATM switches, TDM switches, DWDM systems, OXCs, etc. Consequently, GMPLS is a natural choice for the control plane of pipeline forwarding nodes as it is apparent that the general GMPLS framework fits to the hierarchical resource reservation model with dynamic and provisioned SVPs. In addition, GMPLS provides the pipeline forwarding network with a scalable traffic engineering framework, fast fault protection, and fault recovery.

In the context of MPLS, traditional IP routing protocols (OSPF, IS-IS) and signaling protocols (RSVP, LDP) were extended for supporting traffic engineering (OSPF-TE, IS-IS-TE, RSVP-TE, CR-LDP). GMPLS further extends them to be suitable for handling different network technologies. Additional extensions are required to support pipeline forwarding; although these are the subject of ongoing work, the basic issues can be discussed here at a high level. Routing protocols and signaling protocols should be extended for handling TFs as bandwidth reservation units, i.e., they should be able to carry information related to TF availability and handle availability vectors, respectively.

6. Experimentation

The purpose of this section is to demonstrate that the architecture proposed and described in the paper is not just an academic exercise, but can actually be implemented, by briefly presenting a testbed and some experiments run on it. The prototypes and network setup deployed in the presented experiments have been already presented and discussed in several publications [13,19,20,23], thus an extensive and detailed description is omitted here.

6.1. Network testbed

The presented network architecture has recently been demonstrated [18,19] by means of a testbed based on the prototypal implementations of a PC-based TDP router [13] and an opto-electronic TDS fractional lambda switch [19]. As shown in Fig. 14, the testbed implements a network architecture with UTC-unaware end-systems, TDP routers at the edge, and TDS switches in the core.

Four TDP routers are connected in a full mesh topology and two of them have SVP interfaces connected to asynchronous end-systems. Two Pentium IV based sources generate several flows, some – UDP-based video streaming – requiring pipeline forwarding guaranteed service and others – file transfers – receiving best-effort treatment by TDP routers. One hundred Mb/s links are deployed in the TDP access network, while the TDS backbone is realized with 1 Gb/s optical links. TF duration is set to 200 μs in the TDP routers and to 100 μs in the TDS switches, with 100 TF per TC in both types of nodes. Ten Mb/s access SVPs and 100 Mb/s core SVPs are set up through the access and backbone parts of the testbed network, respectively. A router tester is used in some tests to generate background traffic and fully load the network.

The TDP routers deployed within the testbed are based on a personal computer running a FreeBSD distribution modified to include the TDP packet scheduling algorithm. As discussed in [13] such modifications are negligible as compared to the overall implementation of the router.

Two slightly different TDS switch prototypes are deployed in the testbed. The leftmost switch in Fig. 14 features a two stage switching fabric implemented by connecting two Mindspeed M21151V-EVM switch boards [21]. The two switch boards are controlled and configured by a single FPGA-based switch controller. The FPGA-based switch

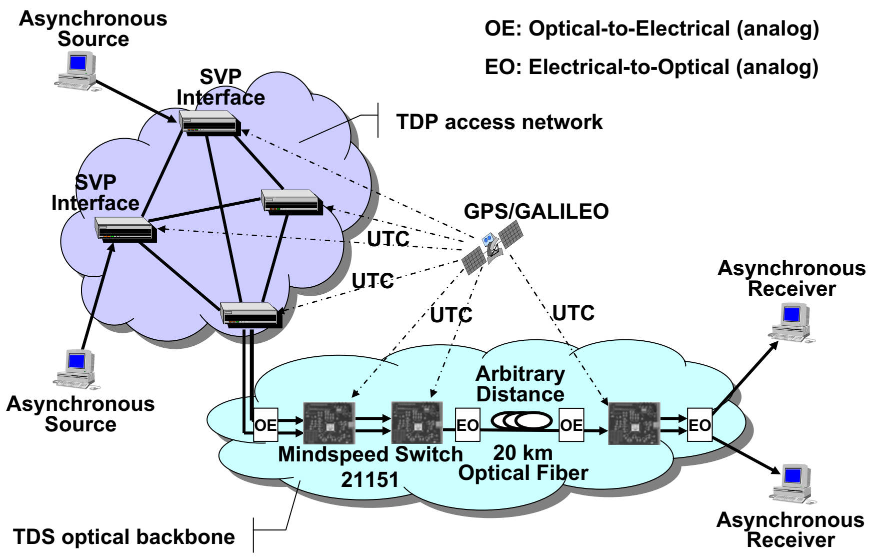


Fig. 14. Pipeline forwarding testbed.

controller has been implemented on an Opal Kelly (XEM3001) [22] module, that has a built-in Spartan-3 FPGA sub module, using Very High Speed Integrated Circuit Hardware Description Language (VHDL). The switching fabric of the rightmost node in Fig. 14 consists of one Mindspeed M21151V-EVM switch board controlled by an identical FPGA-based switch controller. It is possible to control several switch boards using a single switch controller (which embeds a finite state machine (FSM) for each board), thus supporting the realization of a high-capacity switch. Standard single mode and multimode transceivers have been used for optical interconnections between various Electrical to Optical and vice versa conversion points. Bi-directional single mode transceivers have been used for interconnecting the two TDS switches.

Signaling and dynamic creation of SVPs is not supported by the current version of the prototypes. Consequently, in the network testbed, SVPs are statically set up by manual configuration of the devices. TDP routers route packets according to conventional IP address-based routing, not supporting MPLS label-based routing. Finally, only immediate forwarding is currently implemented and no per-flow state information is kept by network nodes.

Extensive testing performed over this network testbed [13,20] confirmed the proper operation of both the TDP router prototype and the TDS switch prototype and the effectiveness of PF to provide guaranteed QoS to streaming media, while handling TCP-based elastic applications the same way as conventional asynchronous IP switching. Consequently, the performance of TCP flow and congestion control algorithms it is not affected, while real-time, possibly interactive, multimedia applications receive optimal service. In particular, Table 1 [13] compares the delay jitters experienced by packets over the previously mentioned full mesh network under link-saturation conditions with PF and FIFO (first in first out) queuing regarding. Five packet flows are looped in the network in order to emulate transport through a large number of hops and their routes are chosen to maximize the number of contentions for the output links. The jitter is measured as the difference between the maximum and the minimum end-to-end delay experienced by packets belonging to the same flow. Although aware of the simplicity of the testbed topology (a toy-network with the main purpose of proving the feasibility of PF), we consider the performed tests significant as they show that PF guarantees a deterministically bounded jitter that is an order of magnitude smaller (a few hundreds microseconds) than the one obtained with asynchronous operation (a few milliseconds).

Table 1
End-to-end delay jitter

Traffic flow	FIFO [ms]	PF [ms]
A	3.58	0.15
B	4.54	0.46
C	4.05	0.40
D	5.10	0.45
E	3.98	0.42

6.2. Concluding remarks

The pipeline forwarding experimentation represents a major contribution in demonstrating the *feasibility* of both pipeline forwarding flavors (i.e., TDP and TDS) and its *effectiveness* in providing optimal support to UDP-based streaming, possibly interactive, traffic, while leaving handling of elastic TCP-based traffic unchanged. The foremost contribution of this work derives from the experience with the prototype realizations as the limited effort they required proves the *very low implementation complexity* of both TDP and TDS. Specifically, the TDP router was basically fully realized as a master degree graduation project in about 7 months, while the TDS switch prototype was developed in less than 10 months by the joint (part time) work of a few Ph.D. students.

Future work will be done along broadly two lines: (1) improving the prototypes, currently at an early stage, (2) conducting more complex experimentation, and (3) all-optical implementation of TDS. Specifically, along the first line the TDP router data plane will be augmented with MPLS label-based routing and non-immediate forwarding capabilities, and a GMPLS control plane for automatic SVP creation will be added to both the TDP router and TDS switch. Finally, the capability of distributing the CTR through the network will be added to TDP routers. Along the second line, experiments and measurements will be conducted on a nation-wide testbed network realized by interconnecting remote sets of TDS switches and TDP routers. This activity will go through a number of steps, each characterized by the deployment of a different interconnection solution: the Internet, a high-speed IP network, and private links.

References

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

[2] M. Baldi, Y. Ofek, Fractional lambda switching, in: Proc. of ICC 2002, New York, vol. 5, pp. 2692–2696.

[3] A. Pattavina, M. Bonomi, Y. Ofek, Performance evaluation of time driven switching for flexible bandwidth provisioning in WDM networks, in: Proc. of Globecom 2004, Dallas, Texas, vol. 3, pp. 1930–1935.

[4] M. Baldi, Y. Ofek, Fractional lambda switching – principles of operation and performance issues, SIMULATION: Transactions of The Society for Modeling and Simulation International 80 (10) (2004) 527–544.

[5] D. Grieco, A. Pattavina, Y. Ofek, Fractional lambda switching for flexible bandwidth provisioning in WDM networks: principles and performance, Photonic Network Communications 9 (3) (2005) 281–296.

[6] M. Baldi, Y. Ofek, Realizing dynamic optical networking, Optical Networks Magazine, Special Issue Dynamic Optical Networking: around the Corner or Light Years Away? 4 (5) (2003) 100–111.

[7] V.-T. Nguyen, M. Baldi, R. Lo Cigno, Y. Ofek, Wavelength swapping using tunable lasers for fractional λ switching, in: 14th IEEE Workshop on Local and Metropolitan Area Networks, Chania, Crete (Greece), September 2005.

[8] H. Zhang, Service disciplines for guaranteed performance service in packet-switching networks, Proceedings of the IEEE 83 (10) (1995).

[9] R. Braden, D. Clark, S. Shenker, Integrated services in the Internet architecture: an overview, IETF Std. RFC 1633, July 1994.

[10] M. Baldi, F. Risso, Efficiency of packet voice with deterministic delay, IEEE Communications Magazine 38 (5) (2000) 170–177.

[11] S. Blake et al., An architecture for differentiated services, IETF Std. RFC 2475, December 1998.

[12] C-S. Li, Y. Ofek, A. Segall, K. Sohraby, Pseudo-isochronous cell forwarding, Computer Networks and ISDN Systems 30 (1998) 2359–2372.

[13] M. Baldi, G. Marchetto, G. Galante, F. Risso, R. Scopigno, F. Stirano, Time driven priority router implementation and first experiments, in: IEEE International Conference on Communications (ICC 2006), Symposium on Communications QoS, Reliability and Performance Modeling, Istanbul (Turkey), June 2006.

[14] K. Nichols, V. Jacobson, L. Zhang, A two-bit differentiated services architecture for the Internet, IETF Std. RFC 2638, July 1999.

[15] M. Baldi, Y. Ofek, Blocking probability with time-driven priority scheduling, in: SCS Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2000), Vancouver, BC, Canada, July 2000.

[16] E. Rosen, A. Viswanathan, R. Callon, Multiprotocol label switching architecture, IETF Std. RFC 3031, January 2001.

[17] E. Mannie et al., Generalized multi-protocol label switching (GMPLS) architecture, IETF Std. RFC 3945, October 2004.

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

[19] D. Agrawal, M. Baldi, M. Corrà, G. Fontana, G. Marchetto, V.T. Nguyen, Y. Ofek, D. Severina, H.T. Truong, O. Zadedyurina, Ultra scalable UTC-based pipeline forwarding switch for streaming IP traffic, in: 25th IEEE Conference on Computer Communications (INFOCOM 2006), Barcelona (Spain), April 2006.

[20] M. Baldi, G. Marchetto, First video streaming experiments on a time driven priority network, in: 1st IEEE Multimedia Communications Workshop, Istanbul (Turkey), June 2006.

[21] Mindspeed Technologies, M21151- 144x144 3.2 Gbps Crosspoint Switch with Input Equalization & Pre-Emphasis, Mindspeed web site at http://www.mindspeed.com/web/products/index.jsp?catalog_id=590&cookietrail=0.

[22] Opal Kelly, Inc., "XEM3001 – Xilinx Spartan-3 Experimentation Module," Opal Kelly web site at http://www.opalkelly.com/products/xem3001/.

[23] D. Agrawal, M. Baldi, M. Corrà, G. Fontana, G. Marchetto, V.T. Nguyen, Y. Ofek, D. Severina, H.T. Truong, O. Zadedyurina, Scalable switching testbed not "Stopping" the serial bit stream, in: IEEE International Conference on Communications (ICC 2007), Symposium on Optical Networks and Systems, Glasgow (Scotland), June 2007.

Mario Baldi is Associate Professor of Computer Networks and head of the Computer Networks Group (NetGroup) at the Department of Computer Engineering of Politecnico di Torino (Technical University of Turin), Italy. He received his M.S. Degree Summa Cum Laude in Electrical Engineering in 1993, and his Ph.D. in Computer and System Engineering in 1998 both from Politecnico di Torino. He co-authored over 70 papers on various networking related topics and two books, one (currently at the second edition) on internetworking and one on switched local area networks. He is co-inventor in five patents issued by the United States Patent Office in the field of optical networking, in 14 applications to the United States Patent Office in the fields of high performance networking and security, and one applications to the European Patent Office in the field of high performance networking. His research interests include internetworking, high performance switching, optical networking, quality of service, multimedia over packet networks, voice over IP, and computer networks in general.

Guido Marchetto (born in 1979) is a Ph.D. student at the Department of Control and Computer Engineering of Politecnico di Torino. He got his laurea degree in Telecommunications Engineering from Politecnico di Torino in April 2004. His research topics are packet scheduling and Quality of Service in packet switched network – Common Time Reference based technologies for packet forwarding (Pipeline forwarding, Time-Driven Priority, Time-Driven Switching). His interests include network protocols and network architectures.

Yoram Ofek has been awarded the Marie Curie Chair Professor in Trento (Italy) by the European Commission. He received his B.Sc. degree in electrical engineering from the Technion-Israel Institute of Technology, and his M.Sc. and Ph.D. degrees in electrical engineering from the University of Illinois-Urbana. He expanded his research while at IBM T.J. Watson Research Center, and for his invention of the MetaRing and his contributions to the SSA storage products, he was awarded the IBM Outstanding Innovation Award. He has written 42 US patents and more than 100 journal and conference papers. He is a fellow of the IEEE.