POLITECNICO DI TORINO
Repository ISTITUZIONALE

Time-Driven Priority Router Implementation: Analysis and Experiments

Original
Time-Driven Priority Router Implementation: Analysis and Experiments / Baldi, Mario; Marchetto, Guido. - In: IEEE TRANSACTIONS ON COMPUTERS. - ISSN 0018-9340. - STAMPA. - 62:5(2013), pp. 1017-1030. [10.1109/TC.2012.71]

Availability:
This version is available at: 11583/2488379 since:

Publisher:
IEEE

Published
DOI:10.1109/TC.2012.71

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

28 April 2024

This is an author's version of the paper

Baldi M., Marchetto G.

"***Time-Driven Priority Router Implementation: Analysis and Experiments***"

Published in

IEEE Transactions on Computers, vol. 62 n. 5, pp. 1017-1030

The final published version is accessible from here:

http://dx.doi.org/ 10.1109/TC.2012.71

©2013 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Time-Driven Priority Router Implementation: Analysis and Experiments

Mario Baldi, *Member, IEEE*, and Guido Marchetto, *Member, IEEE*

Abstract—Low complexity solutions to provide deterministic quality over packet switched networks while achieving high resource utilization have been an open research issue for many years. Service differentiation combined with resource overprovisioning has been considered an acceptable compromise and widely deployed given that the amount of traffic requiring quality guarantees has been limited. This approach is not viable, though, as new bandwidth hungry applications, such as video on demand, telepresence, and virtual reality, populate networks invalidating the rationale that made it acceptable so far. *Time-driven priority* represents a potentially interesting solution. However, the fact that the network operation is based on a time reference shared by all nodes raises concerns on the complexity of the nodes, from the point of view of both their hardware and software architecture. This work analyzes the implications that the timing requirements of time-driven priority have on network nodes and shows how proper operation can be ensured even when system components introduce timing uncertainties. Experimental results on a time-driven priority router implementation based on a personal computer both validate the analysis and demonstrate the feasibility of the technology even on an architecture that is not designed for operating under timing constraints.

Index Terms—Architecture related performance, experiments on a network testbed, packet scheduling, time-driven priority.

1 Introduction: Motivation and Related Work

A significant effort has been devoted during the last two decades to the study of effective techniques for the provision of Quality of Service (QoS) over the Internet. Several frameworks and switch architectures were proposed, with the aim of meeting the increasing QoS demand coming from real-time applications, such as voice over IP and multimedia streaming.

The *Integrated Service (IntServ)* model [1] was firstly proposed. IntServ has the potential to provide absolute QoS guarantees to single packet flows in term of end-to-end delay, jitter, and packet loss. However, it has proven not to scale due to the high complexity and processing requirements associated with packet scheduling algorithms, such as packet-by-packet generalized processor sharing (PGPS) [2], a.k.a. weighted fair queuing (WFQ), combined with the need for their per-flow deployment. Moreover, PGPS and other similar well known scheduling algorithms [3][4], such as, class based queuing, weighted round robin and others, cannot combine optimal delay and resource utilization efficiently (see detailed discussion in [5]). A survey of existing scheduling algorithms for QoS provision, including a discussion of their complexity and realization issues is available in [6].

Due to all of the above, IntServ has not gained a wide acceptance and the second half of 1990s was devoted to the definition of more scalable QoS solutions that could be deployed in the Internet. This effort essentially resulted in the Differentiated Services (DiffServ) model [7], which basically consists in mapping traffic flows to few service classes at the edge of the network and then discriminating among them in the network core by providing service differentiation. DiffServ overcame the scalability issues affecting IntServ, thus rapidly becoming the standard solution (still adopted) for QoS provisioning in packet switched networks. However, this approach cannot withstand a significant increase in the fraction of traffic with QoS requirements as it is combined with overprovisioning of resources, i.e., it assumes that differentiated traffic uses a small fraction of the network capacity.

A simple solution that relies on a more efficient utilization of network resources is needed to allow traffic with QoS requirements to use a large percentage of network capacity. *Time-Driven Priority (TDP)* [8] with *pipeline forwarding* is a packet scheduling technique that can satisfy such requirements thanks to its unique combination of simplicity and efficiency stemming from deploying a *global common time reference (CTR)* for shaping the traffic through the network. Pipeline forwarding provides guaranteed quality of service and scalability, as it has been extensively studied both analytically and through simulations [8]-[13]. In particular, [8] shows how a TDP node combines service guarantees with a buffering complexity comparable to the baseline FIFO algorithm, while [13] proposes a multimedia delivery framework that demonstrates the effectiveness of pipeline forwarding in handling current multimedia applications. These results make TDP very attractive as one of the building blocks of the future Internet. Its simplicity ensures scalability to high performance (multi-terabit) routers and switches. Its efficiency, manifested in high resource utilization, guar-

The authors are with the Dipartimento di Automatica e Informatica, Politecnico di Torino, 10129 Torino, Italy (e-mail: mario.baldi@polito.it; guido.marchetto@polito.it).

Manuscript received (insert date of submission if desired).
Some of the results provided in Section 6 have been previously presented at ICC 2006.

xxxx-xxxx/0x/$xx.00 © 200x IEEE

antees support to a wide diffusion of bandwidth hungry applications with strict QoS requirements, which promises to bring new revenue to an ailing telecom and networking market.

Nevertheless, the fact that in TDP packet forwarding is paced according to a time reference shared by all nodes raises concerns at two levels: (*i*) the complexity of TDP router implementation and (*ii*) additional deployment constraints and requirements due to the need to realize the CTR (sometimes inappropriately paralleled to the need to distribute synchronization in SONET/SDH or PDH networks). Previous work [14] addressed the realization of the CTR by analytically proving and experimentally demonstrating that it can be distributed through the network with limited complexity. On the other hand, existing literature on TDP lacks a detailed study of the impact that the CTR-based operation has on the implementation complexity of a TDP router. Due to its time-driven operation, a TDP router is intrinsically different from traditional IP routers, which naturally raises some questions. Does the synchronous operation of TDP increase the system complexity? Is the resulting cost per switched bit higher than the one characterizing solutions based on the traditional DiffServ model? Can pipeline forwarding and TDP be implemented on simple router architectures or do timing requirements lead to sophisticated architectures? Because of unanswered questions of this nature, many see the CTR as a hurdle with the potential to hinder TDP deployment by assimilating it to other technologies (see for example ATM and SONET/SDH) whose deployment failed to live up to original expectations due to their complexity. This motivates this work that analyzes the requirements that a router architecture and its synchronization signal must satisfy to guarantee proper TDP operation and identifies the most critical implementation aspects. In particular, this paper focuses on the hardware and software architecture of a TDP router and its main contribution is to show how the device can operate properly if the synchronization inaccuracy and the response time to the synchronization signal are upper bounded, without any constraints on such bounds. The synchronization requirements for TDP routers were briefly analyzed in [8], where the authors state that the CTR phase displacement at different nodes has to be smaller than the pipeline forwarding operation time unit, which is called a time frame (TF). However, this bound (which represents a strict requirement on the synchronization accuracy) was given as the maximum admitted CTR error to ensure timing consistency among nodes and considers neither the operations that nodes have to perform on such signal nor the inaccuracies introduced within a real router. This paper extends these results by showing how TDP can properly work with *any* bound on both the synchronization accuracy and the responsiveness of the router modules.

In summary, relying on analytical results, the paper shows how a PC-based implementation of a TDP router is feasible and meaningful. Experiments run on a testbed realized with such routers validate the analysis. The PC-based TDP router implementation is by itself also a significant result of this work, as it shows how TDP, thanks to its simplicity, enables a PC-based software router to offer service guarantees to traffic flows, notwithstanding the general purpose nature of the platform and its inherently unpredictable and far-from-optimized operation as a router.

The rest of the paper is organized as follows. The operating principles of TDP are presented in Section 2, while Section 3 describes the architecture of a TDP router and potential criticalities in its implementation due to system inaccuracies: determining the time at which a packet should be forwarded and actually transmitting it at such time. Section 4 and 5 are devoted to analyzing these criticalities in depth, showing under which conditions TDP operation is not compromised. A case study is presented in Section 6 demonstrating how TDP can be successfully implemented on a general purpose platform, such as the personal computer, that is well known to be unfitted for real-time applications. Conclusions are drawn in Section 7. A summary of the main symbols used throughout the paper is provided in Table I.

2 Time-Driven Priority

As the context of this work is a router architecture implementing *Time-Driven Priority (TDP)*, this section briefly introduces this technology and the concept of "pipelining" the forwarding of packets across the network, on which TDP is based. An extensive and detailed description of TDP and pipeline forwarding is outside the scope of this paper and is available in the literature [8][9].

2.1 Pipeline Forwarding: Time-Driven Priority

In *pipeline forwarding* all packet routers utilize a basic time period called time frame (TF). The TF duration may be derived, for example, as a fraction of the UTC second received from a time-distribution system such as the global positioning system (GPS) and, in the near future, Galileo. As shown in Fig. 1, TFs are grouped into time cycles (TCs) and TCs are further grouped into super cycles, which may, for example, last and be aligned to one UTC second. 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.

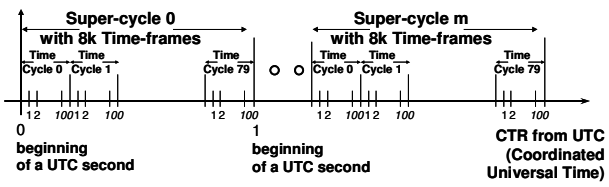


Fig. 1. Common time reference structure

During a resource reservation phase, TFs are partially or totally reserved for each flow on the links along its route. Thus, TFs can be viewed as virtual containers for multiple packets that are switched and forwarded according to the CTR. In particular, a *synchronous virtual pipe (SVP)* is a predefined schedule for forwarding a preallocated amount of bytes during one or more TFs along a path of subsequent UTC-based routers. A signaling pro-

TABLE I
SUMMARY OF FOREMOST SYMBOLS

Symbol **	Definition
d_{ij}	Forwarding delay between nodes *i* and *j*
$w \leq W$	Shaping delay at the SVP interface
T_f	TF duration
Δ	Per-TF packet jitter
J	End-to-end jitter
N_i	Forwarding TF at node *i*
H	Number of TFs per TC
t	Packet transmission time
T	Packet arrival time
P	Propagation delay
$\tilde{D}$	One-shot measurement of the propagation delay
$-\Pi \leq \pi \leq \Pi$	Variation of the propagation delay
$-\mathrm{E}_T \leq \varepsilon_T \leq \mathrm{E}_T$	CTR accuracy at the transmitter
$0 \leq \tau_T \leq \mathrm{T}_T$	Overall transmission latency
$0 \leq \tau_T^{ctr} \leq \mathrm{T}_T^{ctr}$	Transmission latency component due to the CTR
τ_T^{tx}	Transmission latency component due to the output link and interface
$\psi = \varepsilon_T + \tau_T^{ctr}$	Overall transmitter inaccuracy
$-\mathrm{E}_R \leq \varepsilon_R \leq \mathrm{E}_R$	CTR accuracy at the receiver
$0 \leq \rho_R \leq \mathrm{P}_R$	Reception latency
$\hat{g}$	Guard time band duration
T_e	Packet processing time
$0 \leq \lambda \leq \mathrm{M}$	Variation of the packet processing time
C_{TF}	TF capacity
C	Link capacity
R_{TF}	Reserved capacity in a TF
$\mathcal{T}_n^b$	Time at which TF n begins
$\Delta\mathcal{T}_n^b$	Difference between actual and nominal beginning time of TF n
$Buff$	Output buffer size

** For a given symbol, x, the plain symbol denotes the actual value of the parameter, while $\hat{x}$ denotes its nominal value, $\tilde{x}$ denotes its measured value, and x' (used in Section 4.1) indicates a value of the parameter specifically referred to the packet used for estimating the propagation delay.

tocol is needed for performing resource reservation and TF scheduling, i.e., selecting the TF(s) in which packets belonging to a given flow should be forwarded by each router on their path. Existing standard protocols and formats should be used whenever possible. Many solutions have been proposed for the distributed scheduling in pipeline forwarding networks [9] and the generalized MPLS (G-MPLS) control plane provides the protocols suitable for their implementation.

The basic pipeline forwarding operation is regulated by two simple rules:

Rule 1. 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

Rule 2. A packet p transmitted in TF t by a node i must be transmitted in TF $t + d_{i(i+1)}$ by node $i+1$, where $d_{i(i+1)} \geq 1$ is an integer constant (related to the link between nodes i and $i+1$) called *forwarding delay*, and TF t and TF $t + d_{i(i+1)}$ are referred to as the *forwarding TFs* of packet p at node i and node $i+1$, respectively.

It follows that packets are orderly moved along their paths and served at well defined instants at each node. Nodes therefore operate as they were part of a *pipeline*, from which the technology's name is derived. The value of the forwarding delay is determined at resource-reservation time and must be large enough to satisfy the abovementioned Rule 1. In particular, its evaluation has to consider all the factors affecting the time required to move a packets from the output buffer of a node to the output buffer of the next one, such as transmission, propagation, processing, and switching delays. A different value of forwarding delay might be applied:

- on a node basis, to take into account differences in the processing and switching delays of various router architectures;
- on a port basis, to take into account different transmission and propagation delays on incoming links and processing time on input interfaces;
- on a per packet flow basis, in order to achieve higher scheduling flexibility, i.e., at resource reservation time the first TF with enough available resources might be found few TFs after the earliest TF satisfying Rule 1 above.

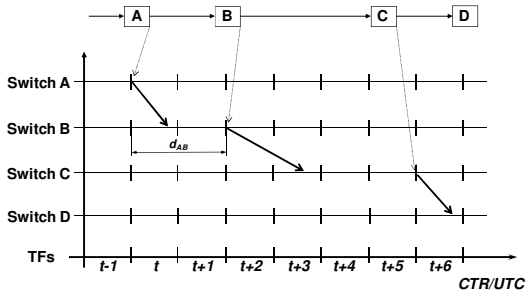


Fig. 2. Pipeline Forwarding operation: a possible schedule to move a packet from node A to node D.

In the latter case, *non-immediate forwarding* is being performed, while *immediate forwarding* is being realized in the other two cases above. Fig. 2 exemplifies a possible journey of an IP packet over a pipeline forwarding network, from a node A to a node D. Different forwarding delays can be observed at different nodes. Without loss of generality the analysis in the remainder of this paper considers a single value for the forwarding delay d to be used throughout the network in order to keep notation simpler and more readable.

On edge routers an *SVP interface* shapes asynchronous traffic entering the pipeline forwarding network. Its input module comprises mechanisms to classify incoming packets, identify the data flow they belong to, and select the proper TF(s) to forward them into the pipeline forwarding network according to the reservation (i.e., SVP)

set up for the flow.

Time-driven priority (*TDP*) is a synchronous packet scheduling technique that couples pipeline forwarding with conventional routing mechanisms to achieve high flexibility together with guaranteed service. While scheduling of packet transmission is driven by time, the output port is selected according to either conventional IP destination-address-based routing, or multi-protocol label switching (MPLS), or any other packet routing technology of choice. While the TF in which a packet is to be forwarded is determined according to the pipeline forwarding operating principles, the transmission order of packets during a TF is not predefined.

2.2 Non-pipelined Traffic

Non-pipelined (i.e., non-scheduled) IP packets — namely packets that are not part of an SVP (e.g., IP best-effort packets) — can be transmitted during any unused portion of a TF, whether not reserved or reserved but currently unused. Consequently, links can be fully utilized even if flows with reserved resources generate fewer packets than expected. Moreover, any service discipline can be applied to packets being transmitted in unused portions of TFs. For example, various traffic classes could be implemented for non-pipelined packets in accordance to the Differentiated Services (DiffServ) model.

2.3 Performance: QoS, Efficiency, and Scalability

A signaling protocol in the control plane of a pipeline forwarding network is expected to handle the reservation of resources during TFs, ensuring that the overall capacity to transmit is not exceeded for each link during each TF. Hence, as demonstrated in [9], packets belonging to flows with reservations traverse the TDP network without contending for resources. This results in pipeline forwarding guaranteeing that pipelined traffic experiences (*i*) bounded end-to-end delay, (*ii*) low delay jitter independent of the number of traversed nodes, and (*iii*) neither congestion nor resulting loss — i.e., the offered QoS service is deterministic. The effectiveness of pipeline forwarding in providing deterministic quality has been investigated in many publications [8]-[13] that demonstrate how this technology can be profitably adopted in various network scenarios including wired [8], wireless [10], and even optical [11] technologies. In particular, [8] proves that the end-to-end delay $Del(h)$ on an SVP encompassing h nodes is

$$Del(h) = w + \sum_{i=1}^{h-1} d_{i(i+1)} \cdot \hat{T}_f + \Delta \,, \qquad (1)$$

where $w \leq W$ is the shaping delay that a packet can experience at the SVP interface, $\hat{T}_f$ is the nominal TF duration, and $\Delta = [0, \hat{T}_f]$ takes into account that the packet can be transmitted at any time during its forwarding TF. Since two different packets could experience $\Delta = 0, w \approx 0$ and $\Delta = \hat{T}_f, w = W$, respectively, the upper bound for the end-to-end jitter is $J = \hat{T}_f + W$, of which only $\hat{T}_f$ is experienced within the TDP subnetwork independently of the number of nodes traversed, namely, of the subnetwork diameter.

However, the overall amount of resources that can be reserved to pipelined traffic is not known in advance and subordinated to the successful creation of SVPs. This, in turn, depends on the possibility of finding a schedule, namely, not on the mere availability of transmission and switching capacity, but also on the time (i.e., the TFs) at which they are available. When a reservation request fails although enough resources are available, but not during the proper TFs, the SVP reservation is said to be *blocked*. Both analytical [9] and simulation [15] studies showed how about 90% or more of network resources can be reserved with negligible blocking probability (i.e., at most few percentage points). This demonstrates the superiority of TDP in terms of efficiency over DiffServ, which instead relies on the assumption that only a small percentage of the link capacity is occupied by traffic with QoS requirements. Improved efficiency in utilization of network resources directly translates in higher *scalability* of the communication system as a larger amount of end-users/end-systems/applications can be accommodated on a network infrastructure with comparable capacity and complexity. From another point of view, given an expected user base and load on the network, a less powerful network infrastructure can be realized to accommodate it. Consequently, TDP can have a significant economical impact because its support for deterministic QoS enables high revenue applications, while its efficiency and scalability allow costs to be contained, thus boosting profits. One of the goals of this paper is to demonstrate that the implementation complexity of a TDP router is not significantly higher than the one of a router supporting DiffServ, thus showing TDP superiority over IntServ as well.

3 TDP Router Implementation

In order to provide the context for the analysis of the complexity of implementing a TDP router, an overview of its architecture is first provided and potential criticalities in implementing it are discussed.

3.1 Router Architecture Overview

Generically, in a packet switch data plane packets are moved from input ports to output ports going through three modules that perform *input processing*, *forwarding*, and *output processing*. The same applies to a TDP router, whose main architectural building blocks is schematically depicted in Fig. 3.

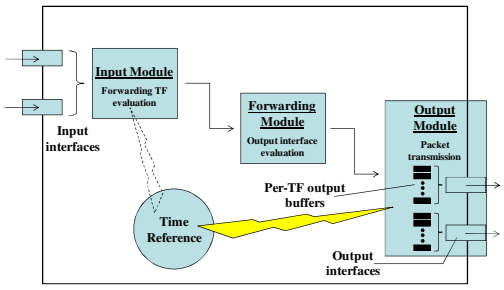


Fig. 3. TDP router architecture.

The input module comprises mechanisms to select the correct TF in which packets will be forwarded according to the current resource reservation setup, i.e., the forwarding TF. The evaluated forwarding TF determines the output buffer where packets will be stored by the output module.

The forwarding module processes packets according to the specific network technology (e.g., IP, MPLS, etc.). A TDP router requires no modification with respect to a traditional packet router as far as the forwarding module is concerned.

The output module implements a per-TF, per-output queuing system, where packets to be forwarded during the same TF through the same interface are buffered in the same queue. The queue in which each packet is stored is determined by both the input module, which decides the forwarding TF, and the forwarding module, which selects the output interface. Finally, the output module is responsible for the timely transmission of all the packets stored in the queues corresponding to the current TF in accordance with a time reference common to all nodes of the network.

As discussed in Section 4, the time reference may be used also in the input module for the evaluation of the forwarding TF. However, more effective methods, not making use of the time reference, will be presented..

3.2 Potential Implementation Criticalities

In summary, two actions, not performed by traditional routers, must be properly executed by a router to implement TDP:

1. Determine the forwarding TF, i.e., the TF during which resources were allocated for transmission of a packet — performed in the input module;
2. Transmit a packet during its forwarding TF — performed in the output module.

If not properly accounted for, low CTR accuracy and non-zero latencies, hereafter collectively referred to as *system inaccuracy*, result in malfunctioning, i.e., packets not being transmitted in their forwarding TF. Consequently, pipeline forwarding is disrupted and its properties cannot be enjoyed. In particular, packets possibly experience a delay longer than expected and network congestion, thus running the risk of being dropped. Section 4 and Section 5 analyze each of the above listed two actions, respectively, and show how their proper implementation ensures *correct TDP operation independently of system inaccuracy*.

4 Forwarding Time Frame Evaluation

Two approaches can be used for computing the forwarding TF of a packet:

1. The input module of a node adds the forwarding delay to the forwarding TF at the previous node, which has therefore to be determined in some way. Thus, if N_{i-1} and N_i are the forwarding TFs at two subsequent nodes $i-1$ and i, respectively, and H is the number of TFs per TC, we have
$$N_i = \left[N_{i-1} + d_{(i-1)i} \right] \bmod H \,. \tag{2}$$
2. The input module of each node classifies and associates incoming packets to their forwarding TF based on the reservation information related to their flow.

Concerning the first approach, there are various, non mutually exclusive ways, to determine the forwarding TF at the previous node, among which (*i*) precisely measuring both the propagation delay and the arrival time of each packet, (*ii*) attaching a time stamp to each packet, and (*iii*) including a TF delimiter within the data stream.

These alternatives are analyzed in this section discussing their strengths, drawbacks, and criticalities, which led to the implementation choices presented in Section 4.5.

4.1 Propagation Delay and Arrival Time Measurement

In order to analyze how inaccuracies affect packet forwarding time evaluation, we first model how the forwarding TF is devised from a propagation delay measurement and the conditions under which the outcome is correct. Then, the various elements in the model are expressed in terms of the system inaccuracies. Packets are time stamped as they are received by an input interface and the TF in which they were sent out by the previous node N_{i-1} is determined through the knowledge of the propagation delay on the incoming link. In particular, if the time origin corresponds to TF 0 of a certain TC and super cycle[1],

$$N_{i-1} = \left\lfloor \frac{T-P}{\hat{T}_f} \right\rfloor \bmod H \,, \tag{3}$$

where T is the packet arrival time (the time elapsed from the system time origin until the reception of the first bit of a packet), $\hat{T}_f$ is the nominal TF duration, and P is the time spent by the packet on the incoming link. In order not to require any manual configuration of topology dependent information, as required for effective deployment in production networks, a router must be capable of measuring the link propagation delay P on its incoming links. This could be done by having the sending end of the link including a time stamp in a packet just before transmitting it and the receiving end comparing this timestamp with the time at which the packet is received. However, as we have previously mentioned, an inaccuracy on the CTR and a non null CTR response latency can cause a delay in transmitting packets. The same issues, together with a non zero latency in the reception of packets, affect also packet time stamping at the receiving end of a link — i.e., the determination of the time T at which a packet is received. All of these inaccuracies result in an error in the measurement of the link propagation delay P. In order to avoid excessive burden on network nodes, a delay measurement is not taken for each packet, but a measurement of the link propagation delay $\tilde{D}$ is taken either only when the link becomes operational, or period-

[1] In order to avoid to unnecessarily complicate equations, the identification of the TC within the super cycle of a given TF has been omitted. The validity of the obtained results is not limited in any way as they relate to the critical issue of identifying the boundaries of a TF, independently of the TC it belongs to. The analysis presented in the paper can be easily extended by identifying a TF as a tuple (N_i,C_i), where N_i is the TF number within TC C_i within a super cycle.

ically through a special purpose link level protocol and used for all subsequent packets. Consequently, the calculation of the TF in which a packet was transmitted is further affected by the inaccuracy of the link propagation delay measurement and is performed as

$$\tilde{N}_{i-1} = \left\lfloor \frac{\tilde{T} - \tilde{D}}{\hat{T}_f} \right\rfloor \bmod H \,, \tag{4}$$

where $\tilde{T}$ is the measured arrival time of the packet.

The transmission TF is calculated correctly as long as $N_{i-1} = \tilde{N}_{i-1}$. Considering that $N_{i-1} = \tilde{N}_{i-1}$ only if they belong to the same TC, we do not compromise on generality by saying that the transmission TF is calculated correctly only if:

$$\left\lfloor \frac{T - P}{\hat{T}_f} \right\rfloor = \left\lfloor \frac{\tilde{T} - \tilde{D}}{\hat{T}_f} \right\rfloor \tag{5}$$

The remainder of this section devises the conditions on the system parameters, specifically the system inaccuracy, under which (5) holds.

Considering the notation described in Table I, the measured time of arrival $\tilde{T}$ for a packet is related to the actual time of arrival T as $\tilde{T} = T + \varepsilon_R + \rho_R$, while the delay experienced by packets over the link, possibly changing for each packet, is $P = \hat{P} + \pi$. An estimate of the link propagation delay $\tilde{D}$ must be devised in order to apply (4). Although different and possibly more effective approaches can be used, for the sake of this analysis the link propagation delay is assumed to be measured as follows: the transmitting end includes in a packet its (measured) transmission time $\tilde{t}'$ and the receiving end calculates the difference with the packet's (measured) time of arrival $\tilde{T}'$. In essence, $\tilde{D} = \tilde{T}' - \tilde{t}'$, where the prime indicates values specifically referred to the packet used for estimating the propagation delay. However, the real propagation delay P of a generic packet is given by the difference of the actual transmission time and arrival time, i.e., $P' = \hat{P} + \pi' = T' - t'$. Considering the system inaccuracy at the transmitter, the value of the time stamp in the packet can be expressed as $\tilde{t}' = t' + \varepsilon'_T - \tau'_T$. By performing some substitutions we derive

$$\tilde{T} - \tilde{D} = (T - P) + \varepsilon_R + \rho_R - \varepsilon'_R - \rho'_R + \varepsilon'_T - \tau'_T - \pi' + \pi \,. \tag{6}$$

For the sake of readability, the term a is defined as follows and used in the rest of the paper:

$$a = \varepsilon_R + \rho_R - \varepsilon'_R - \rho'_R + \varepsilon'_T - \tau'_T - \pi' + \pi \,.$$

The conditions under which the evaluation of the forwarding TF in a TDP router is correct are identified by the following theorem.

Theorem 1. *Sufficient and necessary condition for $N_{i-1} = \tilde{N}_{i-1}$ for any packet transmitted in TF N_{i-1} is that a is an arbitrarily small time interval.*

A formal proof is provided in Appendix A.1. However, this condition is quite intuitive as it formalizes the fact that the estimate of the packet transmission time $\tilde{T} - \tilde{D}$ is going to be close enough to the actual transmission time $(T - P)$ if the compound effect of all inaccuracies is small enough. Put in these terms, Theorem 1 seems trivial and its conditions impossible to satisfy by any real system — which would imply that pipeline forwarding cannot be implemented. However, this stringent condition can be relaxed by imposing a *guard time band* of duration $\hat{g}$ at the beginning and at the end of each TF, so that no transmission shall take place during the guard time band. Deploying guard time bands implies wasting a fraction of the transmission link capacity (specifically, $2 \cdot \hat{g}/\hat{T}_f$), hence it is desirable that $\hat{g} << \hat{T}_f$. When guard time bands are used, the following can be stated.

Theorem 2. *Given a guard time band of duration $\hat{g}$, the necessary and sufficient condition for $N_{i-1} = \tilde{N}_{i-1}$ for any packet transmitted in TF N_{i-1} is that $-\hat{g} \le a < \hat{g}$.*

Theorem 2 (see Appendix A.2 for its proof) can be intuitively explained by considering that a packet transmitted close to the boundaries of a TF is assigned to the wrong TF at the receiving end when its time distance from either beginning or end of a TF is larger than the difference between its actual time of transmission $(T - P)$ and the measured one $\tilde{T} - \tilde{D}$. The theorem provides a way of dimensioning guard time bands based on the knowledge of system accuracies and latencies. Alternatively, the duration of guard time bands can be chosen according to a target efficiency in the usage of transmission link capacity, in which case Theorem 2 is deployed to devise corresponding bounds on system inaccuracy that will drive the design and engineering of network nodes.

4.2 Time Stamp and Time Frame Delimiter

Other proposed solutions to determine the forwarding TF at the previous node — i.e., (*i*) attaching a time stamp to each packet, or (*ii*) including a TF delimiter within the data stream — are not subject to the strict requirements on system accuracy expressed by the theorems in the previous section. According to these methods, the forwarding TF at the previous node is derived directly from specific information carried by packets, i.e., a time stamp according to (*i*) and a specific packet structure/field according to (*ii*). The value of N_{i-1} can be devised from this information, whether explicitly or implicitly coded, independently of the system inaccuracies. This should not give the wrong impression that by using the methods described in this section system inaccuracies do not impact the forwarding TF. In fact, as it will be discussed in Section 4.6, they affect the minimum forwarding delay to be applied, and consequently the overall network performance in terms of end-to-end delay.

The Real-time Transport Protocol (RTP) [16] is likely the most widely deployed solution for carrying time stamps. It is normally adopted in multimedia communications where the receiver uses the time stamps to reconstruct the time profile of traffic at the sender in order to properly reproduce media. Being an application layer protocol, it is used in host-to-host communications where it adds specific information (among which, a 32-bit time stamp) between transport (UDP) header and application data. Consequently, the adoption of RTP for the purpose of time stamping IP packets between TDP routers would

be far beyond the context for which it was designed and require major changes to typical router operations and deployment having them handling application layer protocol information by either tunnel IP packets through RTP sessions between routers or change time stamps in RTP headers of the packets being forwarded. On the other hand, solutions relying on data-link layer protocols (e.g., Real-Time Ethernet [17], which allows transmitting precise time stamps within data frames) would limit TDP to be deployed only on to specific data-link and physical layer technologies. Hence, the time stamp should be preferably inserted at network layer to minimize the impact on the router while ensuring independence from lower level technology. A TF delimiter can be implemented in various ways ranging from defining a control packet to be inserted at the beginning of each TF, to setting a 1 bit field in the first packet transmitted during a TF. The latter is likable as it introduces a very limited transmission and processing overhead, but it requires to either modify the packet header or overload/change the semantics of (a portion of) a field.

4.3 Packet Classification

The forwarding TF of a packet can be determined based on the reservation previously performed for the traffic flow it belongs to. In essence, classification rules based on specific values of some header fields (e.g., source and/or destination address, protocol type, etc.) enable identifying the flow the packet belongs to and a reservation table for the flow indicates the TFs in which packets can be transmitted with deterministic service. The first upcoming TFs listed in the table is used as the forwarding TF for the packet.

Both hardware and software based solutions exist for packet classification. The latter are less expensive and more flexible (i.e., they better adapt to rule updates) than the former, but offer a reduced throughput due to the computational burden related to parsing and matching transport layer information. Examples of hardware-based classifiers are presented in [18][19], while [20][21] describe two software-based solutions.

As with the time stamp and delimiter based methods, no absolute accuracy bounds are required, but system inaccuracies affect the minimum forwarding delay and consequently the overall network performance in terms of end-to-end delay, as discussed in Section 4.6.

4.4 Comparison

Each of the presented solutions for devising the forwarding time frame of a packet features strengths and drawbacks, as summarized in Table II. Here, we provide a detailed comparison that leads to the implementation choices described in the next subsection.

The delay measurement method, analyzed in Section 4.1, requires high accuracy throughout the system (ideally inaccuracy free). This results in high complexity and costs, which make this solution impractical for deployment in commercial devices. The use of larger guard time bands can relax these strict requirements, but at expense of resource utilization and hence of network efficiency. Furthermore, the need of precisely UTC time stamping all incoming packets involves the deployment of special software/hardware to interface the CTR source (e.g., a GPS receiver) with the network card. This further increases the resulting complexity, thus also affecting the node scalability. In fact, this solution could not be deployed in high capacity networks where a large number of packets per second have to be handled at the interfaces. On the positive side, the solution is resilient to packet loss, which does not affect in any way correct pipeline forwarding operation. No complex data structures are required: a 16 bit integer and a 64 bit integer are sufficient to contain the estimated propagation delay and the arrival time of each packet, respectively.

Time stamp based and delimiter based methods (Section 4.2) are similar as far as accuracy requirements, but feature different packet loss resilience and complexity. The former is not affected by lost packets as each packet carries its own time stamp required to properly devise its forwarding TF. Even if carried at the network layer to avoid the extra burden of processing higher layer protocol headers as discussed in Section 4.2, transmitting, parsing, and processing the 16 bit (or more) integer required to represents the time stamp results in some overhead. Furthermore, modifications to existing protocol headers are required as common network layer protocols (e.g., IPv4, IPv6, MPLS) do not feature any field suitable for carrying such time stamp.

TABLE II
FORWARDING TIMEFRAME EVALUATION

Solution	Measurement	Time stamp	TF delimiter	Classification
Hardware accuracy	High	Low	Low	Low
Specific HW/SW	Yes	No	No	No
Scalability	Low	Quite high	Very high	Quite high
Resilience to packet loss	Yes	Yes	No	Yes
Data structure	~80 bits	~16 bits	Few bytes	Several bytes
Protocol modifications	No	Yes	No	No

The TF delimiter method introduces a very limited transmission and processing overhead (thus not compromising scalability) and it is not unlikely that an unused bit be available in existing protocol headers to be used for this purpose (thus not requiring major modifications to the standards). For example, a non reserved codepoint of the DiffServ (DS) field could be used in the header of IP packets.

The drawback of using TF delimiters is sensitivity to the loss of the packets delimiting TFs. As part of the solution, a TF counter (possibly a 16 bit integer) for each input interface is increased whenever a delimiter is received. When a packet arrives, the current value of the counter is the number of the TF during which the packet had been transmitted by the upstream node. If one delimiter is lost, the node is thereafter unable to correctly evaluate the

forwarding TF at the previous node and consequently to correctly realize pipeline forwarding.

Given the memory capacity of modern computer architectures, the three methods can be considered equivalent from the point of view of the complexity and size of the required data structures.

Packet classification can combine the strengths of time stamp-based and delimiter-based methods as it can provide full resilience to packet loss without requiring any protocol header modification. The drawback of packet classification is the more complex data structure (a per flow resource allocation table) and extra processing required to classify packets to their respective flows and look up the corresponding TFs in the reservation table. However, the overhead strictly due to the identification of the forwarding TF can be negligible. In fact, a reservation table might anyway be needed to ensure maximum flexibility in resource bookkeeping (i.e., in the control plane), e.g., to implement non-immediate forwarding. Hence, no additional data structure is required. On the other hand, additional processing is required to classify packets and look up their forwarding TF in the reservation table. This can be limited by reserving resources to flow aggregates rather than single flows [8][9] and using hierarchical resource reservation in the network core in order to enhance scalability [22], while still ensuring pipeline forwarding properties and benefits to each single flow.

4.5 A Best-of-breed, Hybrid Method

Based on the comparison in the previous section, packet classification seems to be the most attractive method for forwarding time frame evaluation as it is robust, scalable, and relatively simple. Here we propose a TF delineation protocol based on the combination of a *robust* TF delimiter and a compressed time stamp that avoids the extra burden of packet classification in situations where only immediate forwarding is implemented. This hybrid method combines the strengths of the two above methods, while avoiding their drawbacks and can be realized using three bits in each packet to carry the combined delimiter/compressed time stamp . One bit toggles each TF, the other one each TC, and the third one each super cycle, which results in an alternating-bit protocol for TF and TC identification[2]. A TDP router keeps track, for each input interface, of the number of the TF and TC during which the last received packet was transmitted by each neighboring node. This information is updated every time the value of the bits in a packet received through an interface is different from the previously received one. TF and TC counter initialization is performed by setting the TF and TC number to zero the first time the bit corresponding to the super cycle toggles. Forwarding TF evaluation is faulty only if all packets sent during one TF are lost, which results in incorrect pipeline forwarding operation, hence risk of congestion. However, disruption is temporary as correct TF evaluation resumes at the beginning of a new TC.

Section 6.2 details how the method was implemented in our TDP router prototype using 8 unreserved DS (DiffServ) codepoints of the DS field which does not require any changes to the standard IP header. The hybrid method can be similarly implemented with other protocols; for example, the EXP field of the shim header can be analogously used for an MPLS-based implementation.

The hybrid method is comparable in implementation complexity and scalability to DiffServ. In our TDP router, the DS field must be processed for each packet to determine the forwarding TF of the packet, i.e., the output queue in which the packet shall be stored. Similarly to DiffServ, a few queues are required on the output interface and packets are scheduled according to a simple algorithm as discussed in Section 5.3. Non-immediate forwarding requires packet classification to identify the forwarding delay associated with the (aggregated) flow to which a packet belongs. Another codepoint in the DS field could be used to identify packets that require non-immediate forwarding, so that only those are processed by the classifier. In this case, the overall processing overhead and scalability of the approach depends on the amount of traffic for which non-immediate forwarding is required. Such amount is small if non-immediate forwarding is limited to flows that are blocked with immediate forwarding. In fact, previous results discussed in Section 2.3 show that finding a schedule for a new flow with immediate forwarding operation is possible when link utilization is below a quite high threshold (e.g., 80-90%).

4.6 Minimum Forwarding Delay

Whatever method is selected to determine the forwarding TF (including the hybrid one), the evaluation of the minimum forwarding delay is critical to the correct operation of the pipeline forwarding network. This value has to be selected at reservation time (in particular, when determining the set of TFs in which capacity should be booked) so that Rule 1 introduced in Section 2.1 is respected. This section analyses how system inaccuracies influence the minimum forwarding delay.

Let us consider a forwarding delay $d_{(i-1)i}$ between two subsequent nodes $i-1$ and i and let $Te = \hat{T}e + \lambda,\ \ 0 \le \lambda \le \mathrm{M}$ denote the (variable) time that node i spends processing the packet (i.e., to perform header processing, routing, etc.).

Theorem 3. *Necessary and sufficient condition on the forwarding delay $d_{(i-1)i}$ (measured in TFs) to guarantee correct pipeline forwarding operation is:*

$$d_{(i-1)i} > \left\lceil \frac{\mathrm{E}_R + \mathrm{E}_T + \mathrm{T}_T + \hat{P} + \Pi + \mathrm{P}_R + \hat{T}e + \mathrm{M}}{\hat{T}_f} \right\rceil + 1 . \quad (7)$$

In essence, as per Rule 1, packets must be in node i output buffer by the TF preceding their forwarding TF,

[2] Such mechanism can be seen as the transmission of a time stamp composed of the TC and TF number where, in order to reduce the amount of information transmitted, the numbers are compressed by sending only the least significant bit. Alternatively, the mechanism can be seen as delimiting the beginning of each TF by changing the value of 3 bits in a way that is tolerant to some level of loss. Notice that if a reservation period longer than a time cycle is not needed, as in many practical deployments, two bits suffice as the identification of TC within super cycle is not required.

which is expressed by the +1 term in the theorem. By the end of that TF packets must have gone through the link between the nodes (worst case propagation delay Π) and undergone the processing at node *i*, (taking at most $\hat{T}e + \mathrm{M}$), while taking into account the maximum uncertainty due to system inaccuracies. This provides a latency figure that needs to be rounded up to an integer number of TFs. Proof of the theorem is available in Appendix A.3.

5 Packet Transmission

5.1 Criticalities

In principle, the transmission of packets scheduled during a given TF must start as soon as the TF begins. In this way, a total of $\hat{C}_{TF} = \hat{T}_f \cdot C$ bits can be transmitted during a TF, where $\hat{T}_f$ is the nominal TF duration (in seconds), and C is the link capacity (in bit/sec). Poor system accuracy possibly results in a non deterministic variation of the duration of each TF T_f, hence of the actual TF capacity that in reality is $C_{TF} = T_f \cdot C$ bits. The system is not able to honor a resource reservation R_{TF} for a TF, thus not performing proper TDP operation, when the following applies for at least one TF:

$$C_{TF} < R_{TF} \leq \hat{C}_{TF} \tag{8}$$

When the TDP traffic load is low, it is likely that $R_{TF} << \hat{C}_{TF}$ for any TF[3] and system inaccuracies do not have any consequence. However, at high reserved traffic levels, system inaccuracies might result in $R_{TF} > C_{TF}$ and possibly in TF "overflows": the transmission of the R_{TF} bits scheduled during a certain TF begins late and consequently does not end before the TF is over. Consequently, some packets can remain into the node, building up a backlog of packets, with consequent delay, jitter and buffer overflows. Alternatively, a preemption mechanism can be implemented so that backlogged packets are discarded in order to start the following TF and avoid penalizing the service provided to subsequent packets. In both cases, the scheduling and the guaranteed service are disrupted.

5.2 Discussion

The abovementioned delay in beginning the transmission of the packets scheduled during a TF is due to both the inaccuracy of the CTR at the transmitter ε_T and the TF transmission latency τ_T. It is worth noticing that τ_T may include two different components: a *CTR response latency* τ_T^{ctr} — the time between the nominal beginning of a given TF and the instant at which the system actually schedules the transmission of the first packet of the TF — and a *transmission latency* τ_T^{tx} — the time between when a packet is scheduled for transmission and the actual transmission of the first bit of the packet. Hence, $\tau_T = \tau_T^{ctr} + \tau_T^{tx}$, where τ_T^{tx} depends on the specific output link technology, as well as the specific hardware and software implementation of the network interface. For example, the transmission of a preamble before the beginning of a packet and an Inter Frame Gap after its transmission introduce a transmission latency in Ethernet links. Further latencies may be introduced by the network port controller: for example, in simple system architectures an interrupt for the main processor may be generated after the transmission of each packet to notify that the next packet can be transmitted, which requires some time to be served. Since a non null τ_T^{tx} prevents the usage of part of the TF resources before the transmission of each packet, τ_T^{tx} must be taken into account at resource reservation time. Note that this bandwidth waste is TDP-specific: since it is related to the specific technology adopted to interconnect routers and the implementation of their network interfaces, it is experienced independently of the scheduling algorithm deployed.

In summary, considering τ_T^{tx} a per-packet overhead, the overall transmitter inaccuracy delaying transmission at the beginning of each TF can be expressed as: $\psi = \varepsilon_T + \tau_T^{ctr}$. One way to ensure correct TDP operation notwithstanding ψ consists in reducing the overall amount of traffic planned to be transmitted during a TF. In other words, poor system accuracy translates in bandwidth waste that must be taken into account at reservation time (i.e., some extra bandwidth must be set aside, like for τ_T^{tx}) in order to avoid uncontrolled delay, jitter, and possibly packet loss.

However, unlike for τ_T^{tx}, the bandwidth waste related to ψ can be avoided with a proper operating mode. In particular, guaranteeing deterministic quality of service, i.e., avoiding losses and unpredictable delay and jitter due to network congestion, is possible by simply forwarding all packets that match the predefined schedule for TF N_i, i.e., that have been reserved resources during TF N_i, even if this requires extending the transmission beyond the end of TF N_i. According to this new operating mode, transmission of packets scheduled during a TF N_i may end at different times on different output interfaces of the same node. This leads to a new definition for the TF beginning, which is no longer specific only to a node i, but also to a particular output interface:

Definition: ***Inaccuracy-tolerant pipeline forwarding operating mode.*** The beginning of a new TF on an output interface is identified by the latest of the following events:

1. the TF beginning signal is provided by the CTR,
2. all the packets scheduled for transmission during the current TF becomes empty.

Thus, a generic TF (which for the sake of notation simplicity we denote as n in the rest of this section) at a generic TDP node begins at a time $\mathcal{T}_n^b$ which differs from the nominal beginning time $\hat{\mathcal{T}}_n^b$ on an ideal (zero-latency, zero-inaccuracy) node implementing the original pipeline forwarding operating mode [8][9]. Furthermore, a delay in the beginning of a TF may result in a delay in the beginning of the following one (unless the amount of data to be transmitted during the first TF is small). Necessary condition for the delay tolerant operating mode to maintain the properties of the original is that the time difference between actual TF beginning and ideal TF beginning are bounded and non-additive, i.e., an upper bound for

[3] At least, this is the case if even distribution across all TFs is among the resource allocation objectives.

$\Delta\mathcal{T}_n^b = \left|\mathcal{T}_n^b - \hat{\mathcal{T}}_n^b\right|$ does exist. This is necessary to ensure that the latest time at which a packet is forwarded at any node, and consequently the total end-to-end delay experienced by the packet through the network, be deterministically known in advance.

Theorem 4. *In a TDP node deploying the inaccuracy-tolerant pipeline forwarding operating mode where* $-\mathrm{E}_T \le \varepsilon_T \le \mathrm{E}_T$ *and* $0 \le \tau_T^{ctr} \le \mathrm{T}_T^{ctr}$ *(i.e.,* $-\mathrm{E}_T \le \psi \le \mathrm{E}_T + \mathrm{T}_T^{ctr}$*), the time difference between the actual TF beginning and ideal TF beginning is bounded as:*

$$\Delta\mathcal{T}_n^b = \left|\mathcal{T}_n^b - \hat{\mathcal{T}}_n^b\right| \le \mathrm{E}_T + \mathrm{T}_T^{ctr} \ \forall n \tag{9}$$

In addition to ensuring that pipeline forwarding properties are maintained, Theorem 4 (see Appendix A.4 for a formal proof) provides the maximum (worst case) latency in beginning packet transmission during any TF as $\mathrm{E}_T + \mathrm{T}_T^{ctr}$. The forwarding delay evaluation according to (7) includes such latency.

5.3 Implementation

The inaccuracy-tolerant operating mode is implemented by continuing retrieving packets from the buffer related to TF n until it is empty even after the CTR signal marking the beginning of TF $n+1$. Once such buffer is empty, retrieval from the buffer related to TF $n+1$ can start.

When pipeline forwarding is deployed within the network layer of a router, the inaccuracy-tolerant operating mode is most likely indirectly ensured by the presence of a transmission buffer in the data-link layer protocol implementation (e.g., in the Ethernet driver or network interface card). In fact, packets that have to be transmitted during TF n are retrieved by the pipeline forwarding scheduler at the network layer and transferred to the data-link layer as soon as TF n begins. Since the time required by this operation — consisting in a pointer adjustment, a memory copy, or a transfer through the system bus, depending on the router architecture — is negligible compared to the time required to transmit the packets on the output link (i.e., the total TF duration) the buffer related to TF n has been emptied when TF $n+1$ begins. Instead, the data-link buffer is not empty when TF $n+1$ begins if $\psi_{n+1} < \psi_n$ and it has to be dimensioned properly in order to avoid buffer overflow with consequent packet loss. From Theorem 4, the worst case to handle is

$$\psi_n^{\max} = \mathrm{E}_T + \mathrm{T}_T^{ctr}\ ,\ \psi_{n+1}^{\min} = -\mathrm{E}_T\,. \tag{10}$$

Thus, if C is the output link capacity, the data-link buffer shall be dimensioned as

$$Buff = \left[\hat{T}_f + \left(\psi_n^{\max} - \psi_{n+1}^{\min}\right)\right]\cdot C = \left[\hat{T}_f + \left(2\mathrm{E}_T + \mathrm{T}_T^{ctr}\right)\right]\cdot C\ . \tag{11}$$

6 Case Study: A PC-Based Implementation

The guidelines presented in Section 4.5 and Section 5.3 have been used to develop a PC based TDP router [23] as described in Section 6.1. Section 6.2 specifically describes the implementation of the hybrid method presented in Section 4.5. The timing inaccuracies and limitations of a PC are analyzed in Section 6.3 as they have to be taken into account when implementing a TDP router. Section 6.4 reports on experiments that demonstrate that the guidelines discussed and proven in previous sections actually do ensure correct pipeline forwarding operation notwithstanding system inaccuracy. The router prototype considered throughout this section is a perfect environment for validating the concepts and theorems presented in Sections 4 and Section 5 because it is based on the general purpose architecture of the PC, which has not been designed for operating as a router, even less a TDP one. In fact, the PC architecture is well known to be particularly unfitted for real-time applications as control on operation timing is not one of its design objectives.

6.1 TDP Router Implementation

The developed TDP router is based on the routing software of the FreeBSD 4.8 operating system running on a 2.4 GHz Pentium IV PC equipped with Intel PRO/1000 MT server adapter Gigabit Ethernet cards; the TDP scheduling algorithm is implemented in the FreeBSD kernel.

The input module determines the forwarding TF of each TDP packet by implementing the DS field-based solution presented in Section 4.5. Our input module implementation also includes SVP interface features, which enables the TDP router to be used at the edge of a pipeline forwarding network connected to nodes that do not perform pipeline forwarding. As described in Section 2.1, an SVP interface

- Classifies each incoming packet to identify the data flow it belongs to, and
- Determines the TF during which the packet should be forwarded by the output module (i.e., its forwarding TF) based on the resource reservation of its flow.

The forwarding module performs conventional IP routing as implemented in the FreeBSD kernel. Switching relies on the shared (by input and output ports) PC memory and bus.

In the output module we implemented the per-TF, per-output queuing system described in Section 3.1 and the inaccuracy-tolerant operating mode presented in Section 5.2. Section 6.4 shows that the output buffer of Intel Gigabit Ethernet cards (implementing the data-link buffer discussed in Section 5.3) satisfies the requirements to support such operating mode as expressed by (11).

UTC is provided to our prototypal router by a Symmetricom GPS receiver PCI card that can generate interrupts at a programmable rate ranging between 1 Hz (1PPS — pulse per second) and 250 kHz (every 4 µs). Such interrupts are used to pace the beginning of TFs; whenever an interrupt occurs, the values of the current TF and TC are updated.

6.2 Forwarding Time Frame Evaluation Method

In our prototype router, forwarding TF evaluation relies on the hybrid method presented in Section 4.5 that combines a (compressed) time stamp and TF delimiter. The DS field in the IP header is used to carry the combined time stamp and delimiter: bits 0x0c are set in all TDP packets to distinguish them from those not receiving TDP service (e.g., best-effort or differentiated service packets),

bit 0x10 is set to 1 (0) in packets transmitted during odd (even) TFs, and bits 0x20 and 0x40 toggle their value every TC and every super cycle, respectively. Whenever the DS codepoint of a packet received through an interface is different from the previous one, the latest transmission TF and TC counter kept for the upstream node is updated. Counter initialization is performed by setting the TF and TC number to zero the first time bit 0x40 toggles. Consequently, system initialization lasts up to the super cycle duration (typically 1 s), but happens only when a link first becomes operational and does not require transmission of additional information. When a node has no packets (including non-TDP packets) to transmit on a given link, it sends sequences of *padding* IP packets with proper TF and TC marking for keeping the TF and TC counters on the router at the other end synchronized[4]. This solution is elegant and effective since (*i*) it does not require any new standard or protocol on IP networks, (*ii*) introduces very-limited computational overhead and no transmission overhead, and (*iii*) is resilient to packet losses.

6.3 PC Architecture Inaccuracies and Limitations

Errors and inaccuracies discussed in Section 4 and Section 5 are particularly significant in a PC-based TDP router due to the general purpose nature of the underlying hardware and software architecture. In particular, interrupts are serviced a variable amount of time later than they are triggered by peripheral devices. This latency depends on several factors — among which the priority of the incoming interrupt, the current CPU load, and the interrupt service policies of the employed operating system — that make interrupt timing heavily non-deterministic. Since TF beginning and end are determined by the periodic PCI interrupt generated by the GPS receiver, each TF begins with a variable unpredictable delay with respect to ideal operation. Similar uncertainties affect the packet receiving procedure that is triggered by interrupts generated by network cards.

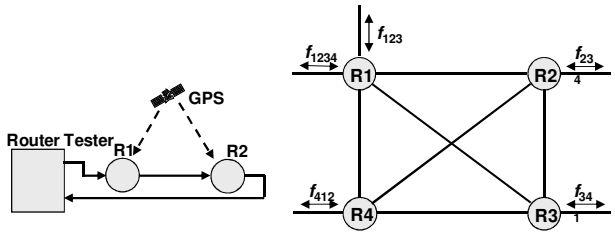
Additional latencies in transmitting packets stem from the mono-processor and mono-bus architecture of the PC. In fact, TDP requires that all interfaces start transmitting packets scheduled during a given TF at the beginning of such TF. The TDP router prototype, instead, due to the above mentioned architectural characteristics, handles interfaces sequentially, delaying the beginning of transmission on all output links except one. Although the additional delay has a minor impact since it is equivalent to a link being longer (i.e., a greater $\hat{P}$), its variations contribute to the system inaccuracies that could affect TDP operation. Such variations are essentially due to the non-deterministic bus-acquisition time and are outside the control of the operating system. The additional delay resulting from the above mentioned characteristics of the PC architecture can be taken into account within π as defined in Section 4.1.

4 Notice that this does not represent a bandwidth waste since the transmission link would anyway be idle.

6.4 Experiments

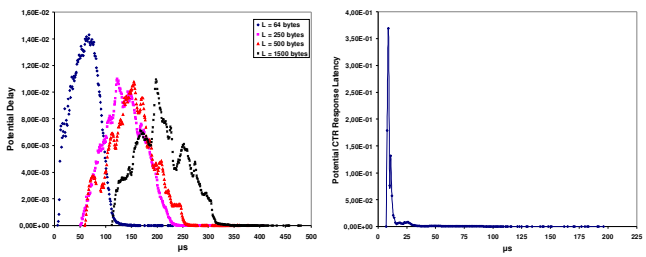
A first set of experiments is run to measure the various system inaccuracies that affect the forwarding delay evaluation and the data-link buffer dimensioning.

As defined by Theorem 3, the forwarding delay must take into account inaccuracies related to several parts of the system, i.e., the source of the CTR, the transmitter, the receiver, and the link. Since our lab is not equipped to measure each latency component separately, the testbed shown in Fig. 4(a) is deployed to measure the time interval from the nominal beginning of a TF to the arrival of the first packet transmitted during the TF to the output buffer of the next node. Such interval includes the terms at the numerator of the fraction at the right member of (7) introduced by Theorem 3: its highest measured value can be used to derive a lower bound for the forwarding delay. In order to perform the measurement, an Agilent N2X Router Tester is used to generate a traffic flow that enters TDP router R1, is forwarded to router R2, and then is routed back to the router tester. Since the TF duration is 250 µs, the beginning of the current TF is calculated as the largest integer multiple of 250 µs smaller than the time of day devised from the GPS receiver. Time from the GPS receiver is used on R2 to measure the instant at which the first packet of each TF reaches the output buffer. Note that the CTR source inaccuracy can be neglected as its upper bound E_T is 340 ns for the deployed GPS receiver [24]. In addition, the propagation delay is negligible for all purposes since a short cable is used between the testbed routers stacked one on top of the other.



(a) (b)

Fig. 4. (a) Synchronization error evaluation testbed; (b) Full experiment testbed.



(a) (b)

Fig. 5. (a) Distribution of buffer-to-buffer time; (b) Measured CTR response latency distribution.

Fig. 5(a) plots the distribution of the time interval measured over several test runs with fully loaded links and various packet lengths. The lower bound for the forwarding delay as derived from the measurements is:

$$d > \left\lceil \frac{480\,\mu s}{250\,\mu s} \right\rceil = 2 \text{ TF} \Rightarrow d = 3\,\text{TF}. \quad (12)$$

Notice that, being the propagation delay negligible, the forwarding delay (750 µs) basically represents the latency introduced by each node independently of traffic load condition. This is sensibly lower than the one a packet could experience in output buffers of a traditional asynchronous router (several milliseconds), especially under high load conditions.

The described testbed is also used to measure the CTR response latency τ_T for the PC-based router R2, whose distribution is plotted in Fig. 5(b), as the difference between the GPS time when the transfer of the first packet of a TF to the data-link output buffer is initiated and the nominal beginning of the TF. This is used for dimensioning the data-link output buffer according to (11). From Fig. 5(b) we can derive

$$\mathrm{T}_T \approx 195\ \mu s\,.$$

Being E_T negligible for our purposes, it can be concluded that a suitable size for the data-link buffer is

$$Buff = \left[\hat{T}_f + \left(2\mathrm{E}_T + \mathrm{T}_T\right)\right] \cdot C \approx 43\,\mathrm{KB}\,. \quad (13)$$

This value is comparable with the size of data-link buffers currently deployed in network nodes, which result therefore eligible for properly handling CTR inaccuracies. For example, our Intel PRO/1000 MT server adapters provide a 64 KB onboard output buffer, which satisfies the buffer size requirement devised according to (13).

These results are used to tune the system parameters for a second set of experiments on the network testbed showed in Fig. 4(b) aimed at validating the inaccuracy-tolerant operating mode presented in Section 5.2. The goal of the experiments is to verify that packets traveling across a large number of TDP routers implementing inaccuracy-tolerant pipeline forwarding receive deterministic service with end-to-end delay and jitter within the theoretical TDP bounds. Not being able to setup a large network in our labs, we used a testbed composed of 4 TDP routers connected by 100 Mb/s Ethernet links with 250 µs TFs and routed packets along long paths by having them traversing multiple times the same node. Specifically, five traffic flows are injected in the testbed network by the Agilent Router Tester. Each injected packet loops several times along a circular route before being routed back to the Router Tester, which has the combined effect of traffic traversing long paths and high traffic load resulting on network links. This is achieved by modifying the forwarding module to make IP routing decisions based on both the destination address and the time-to-live (TTL) field. The flow names subscripts in Fig. 4(b) indicate the list of routers the corresponding flow traverses. For example, flow f_{1234} enters the network through router R1, loops along the 4-hop path R1→R2→R3→R4→R1, and then is routed back to the Router Tester through the same interface on R1 it came from. Each flow contributes 6.4 Mb/s and loops 5 times through the network, resulting in a 21 hop route (considering that R1 is traversed both entering and exiting the loop) for f_{1234} and in a 16 hop route for the other flows. This produces an overall load of 96 Mb/s on each link traversed by flow f_{1234}, thus achieving 96% link utilization. Such utilization level is in line with the analytical and simulation results concerning blocking in TDP networks presented in Section 2.3. The overall load is 32 Mb/s on links R1→R3 and R2→R4 as traversed by less traffic in the network configuration presented in Fig. 4(b). Therefore, each router forwards about 140 Mb/s corresponding to about 17,000 packets per second.

TABLE III
END TO END DELAY AND JITTER

Flow	TDP Delay [ms]	TDP Jitter [ms]	FIFO Jitter [ms]
f_{1234}	16.60	0.18	3.10
f_{123}	12.29	0.32	1.98
f_{234}	12.27	0.35	2.28
f_{341}	12.29	0.37	2.28
f_{412}	12.31	0.45	2.32

Deterministic service is obtained by properly reserving resources to each flow. As described above, we need to accommodate 6.4 Mb/s flows. First, incoming traffic needs to be properly time-shaped before being injected in the TDP network. In our testbed, SVP interfaces are given an opportunity to transmit packets of each flow every 5 TFs. Second, several flows might share the capacity of each TF, i.e., the total number of bytes that can be transmitted within its duration. Considering for example link R1→R2, three flows, namely f_{1234}, f_{123}, and f_{412}, contend for its capacity; the deployed resource allocation policy has packets from each of the three flows transmitted during the same TF. Consequently, 1/3 of each TF is reserved to each of the three flows. Notice how, from the point of view of each flow, being scheduled to transmit for 1/3 of the TF capacity every 5 TFs corresponds to a maximum traffic rate of 6.4 Mb/s, thus satisfying our requirements. Scheduling of the TF to be used on each link (at a specific loop iteration) by packets belonging to a given flow is based on the fact that packets transmitted in TF n at node Ra, are scheduled for transmission at the subsequent node Rb in TF $n+3$ since, from (12), the forwarding delay is set to 3 TFs. Since the implementation of a control plane for TDP is not the purpose of this work, resource reservation is performed manually for simplicity. However, notice that this process can be automated by deploying one of the distributed algorithms presented in [9], as mentioned in Section 2.1.

Delay and jitter measurements on this TDP network are performed at the Router Tester for all involved traffic flows and the maximum values observed are shown in Table III. The measured values of end-to-end delay are below the analytical bound for the corresponding flow. The latter can be computed from Equation (1) considering that the defined SVPs traverse either 21 (the one carrying f_{1234}) or 16 nodes (the others) and that $W = 0.28 + 1.25$ ms is the maximum time taken by the ingress router to process packets and move them to the output (roughly estimated from the first set of experiments as 480-195 µs) plus the time packets spend waiting for the first reserved time frame ($5 \cdot \hat{T}_f$). The theoretical bound on the end-to-end delay contribution due to buf-

fering within network nodes (i.e., excluding the shaping delay w at the SVP interface) is 750 µs multiplied by the number of traversed hops minus 1, as expressed by (1). Furthermore, also the jitter is under the theoretical bound ($\hat{T}_f + W$) and no losses are observed.

The measurement experiments (*i*) validate the inaccuracy-tolerant operating mode and (*ii*) demonstrate that TDP can be easily and properly implemented in a network device, even if based on low cost general purpose architectures like the PC.

For the sake of completeness, Table III also presents measurements of the delay jitter experienced by packets in the network scenario depicted in Fig. 4(b) using traditional asynchronous routers, i.e., when FIFO (first in first out) queuing policy is enabled instead of TDP. These results are significant as they show that the jitter obtained with asynchronous operation is an order of magnitude greater (a few milliseconds) than the one guaranteed by TDP (a few hundreds microseconds) even on the simple network topology of the testbed.

Although the setup seems simple, the experiment is creating a scenario in which only a correctly implemented pipeline forwarding can avoid congestion: a heavily loaded large scale network where a packet flow traverses several highly loaded links multiplexing cross traffic from different input links. More sophisticated traffic patterns (different than a constant packet flow as used in the experiments) would certainly create interesting challenges to the SVP interfaces, but once in the pipeline forwarding domain, packets anyway proceed through the network according to the regularly paced forwarding independently of the profile with which they were generated. Hence, when looking at the links in the pipeline forwarding domain, the only difference that can be observed when the traffic is offered according to more sophisticated patterns, is that a fraction of the time frames might not be fully occupied. Finally, also the simplicity of the network topology does not impact the generality of the experimental results: a more sophisticated topology would not affect the operation of the data plane, but just challenge the control plane.

However, we include for the sake of completeness a brief overview of some experiments run on a realistic, although simple, network scenario. The router described in this paper is used as part of a wider testbed also including some optoelectronic switches operating pipeline forwarding, specifically Time-Driven Switching (TDS) [11]. The switch architecture and implementation is described in [25], while the testbed (shown in Fig. 6) is covered in several different publications [12][22][26][27]. 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, two — UDP-based video streaming — requiring pipeline forwarding guaranteed service and others — file transfers — receiving best-effort treatment by TDP routers. 100 Mb/s links are deployed in the TDP access network, while the 20 km 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 backbone switches, with 100 TF per TC in both types of nodes. 10 Mb/s access SVPs and 100 Mb/s core SVPs are set-up through the access and backbone parts of the testbed network, respectively. The router tester is still used in some tests to generate background traffic and fully load the network.

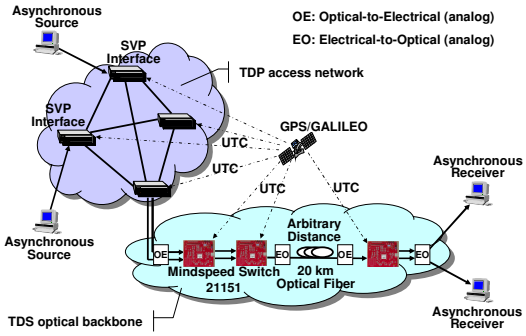


Fig. 6. Wide-area testbed.

TABLE IV
END TO END JITTER

Video Flow	PF [ms]
1	0.14
2	0.16

This testbed provided both qualitative and quantitative results concerning the goodness of the technology. First, video reproduction was perfectly fluent and without interruptions at the receivers with a replay buffer of about 1 KB. Second, the jitter observed for the two video flows at their receivers was within the theoretical bound, as shown in Table IV [27]. This brief discussion, besides further validating our implementation, gives an idea of the real potential of the pipeline forwarding technology in supporting real streaming media applications. An extensive validation of the technology in this perspective is however provided, as said, in the publications mentioned throughout the paper.

7 CONCLUSIONS

This work focuses on the requirements on the hardware and software architecture of TDP routers stemming from their time-driven operation. The analysis in Section 4 and Section 5 shows that timing inaccuracies can be properly taken into account in the dimensioning of the system (e.g., the buffers). This ensures deterministic operation of network nodes with properties comparable to ideal ones, i.e., nodes not introducing any inaccuracy.

Experiments conducted on a testbed composed of TDP routers implemented on commercial personal computers running the FreeBSD routing software (Section 6) validate the analytical results. In essence this work demonstrates that

- Even a general purpose hardware architecture such as the personal computer that has not been designed to operate with predictable timing can support proper TDP operation;
- A traditional asynchronous router software can be easily modified to include TDP queuing.

A hardware architecture designed to reduce inaccuracies possibly coupled with a specialized routing software (e.g., not running within a general purpose operating system such as FreeBSD) would limit the inaccuracies, and consequently the buffering delay introduced by each node. However, it is worth noticing that even the buffering delay introduced by the prototypal TDP router deployed in the experiments is significantly lower than the delay (several milliseconds) that a packet could experience in the output buffer of a traditional asynchronous router, especially in high load conditions.

REFERENCES

[1] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet architecture: an overview," *IETF Std. RFC 1633*, July 1994.

[2] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control – the multiple node case," *IEEE/ACM Trans. Networking*, vol. 2, no. 2, pp.137–150, 1994.

[3] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proc. of the IEEE*, Vol. 83, No. 10, 1995.

[4] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Trans. Networking*, Vol. 3, No. 4, 1995.

[5] M. Baldi and Y. Ofek, "End-to-end delay analysis of videoconferencing over packet-switched networks," *IEEE/ACM Trans. Networking*, Vol. 8, No. 4, pp. 479-492, Aug. 2000.

[6] G. Nong and M. Hamdi, "On the Provision of Quality-of-Service Guarantees for Input-Queued Switches," *IEEE Communications Magazine*, Vol. 38, No. 12 , 62-69, Dec. 2000.

[7] S. Blake et al., "An architecture for Differentiated Services," *IETF Std. RFC 2475*, Dec. 1998.

[8] C.-S. Li, Y. Ofek, and M. Yung, "Time-driven priority flow control for real-time heterogeneous internetworking," *IEEE Int. Conf. on Computer Communications (INFOCOM 1996)*, San Francisco, CA, Mar. 1996.

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

[10] M. Baldi, R. Giacomelli, G. Marchetto, "Time-Driven Access and Forwarding for Industrial Wireless Multihop Networks," *IEEE Transactions on Industrial Informatics*, vol.5, no.2, pp.99-112, May 2009.

[11] D. Grieco, A. Pattavina and Y. Ofek, "Fractional Lambda Switching for Flexible Bandwidth Provisioning in WDM Networks: Principles and Performance," *Photonic Network Communications*, Vol. 9, No 3, May 2005, pp. 281-296.

[12] M. Baldi, G. Marchetto, "First Video Streaming Experiments on a Time Driven Priority Network," 1st IEEE Multimedia Communications Workshop, Istanbul, Turkey, June 2006.

[13] M. Baldi, J. C. De Martin, E. Masala, A. Vesco, "Quality-Oriented Video Transmission With Pipeline Forwarding," *IEEE Trans. Broadcasting*, Vol. 54, No. 3, pp. 542-556, Sep. 2008.

[14] M. Baldi, G. Marchetto, "Pipeline Forwarding of Packets based on a Low Accuracy Network-distributed Common Time Reference," *IEEE/ACM Transactions on Networking*, Vol. 17, No. 6, pp. 1936-1949, Dec. 2009.

[15] M. Baldi, Y. Ofek, "Blocking Probability with Time-driven Priority Scheduling," *SCS Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2000)*, Vancouver, Canada, July 2000.

[16] H. Schulzrinne et al., "RTP: A Transport Protocol for Real-Time Applications,"*IETF Std. RFC 3550*, July 2003.

[17] Felser, M.; , "Real-Time Ethernet - Industry Prospective," *Proceedings of the IEEE* , Vol. 93, No. 6, pp.1118-1129, June 2005.

[18] Q. Dong, S. Banerjee, J. Wang, D. Agrawal, "Wire speed packet classification without tcams: a few more registers (and a bit of logic) are enough," *ACM Int. Conf. on Measurement and Modeling of Computer Systems (SIGMETRICS 2007)*, San Diego, CA, June 2007.

[19] A. Kennedy, X. Wang, and B. Liu, "Energy Efficient Packet Classification Hardware Accelerator," *Proc. IEEE Int. Symp. On Parallel and Distributed Processing (IPDPS 2008)*, Miami, FL, Apr. 2008.

[20] P. Gupta and N. McKeown, "Classifying Packets with Hierarchical Intelligent Cuttings," *IEEE Micro*, Vol. 20, No. 1, pp. 34-41, Jan. 2000.

[21] S. Singh, F. Baboescu, G. Varghese, and J. Wang, "Packet classification using multidimensional cutting," *ACM Int. Conf on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM 2003)*, Karlsruhe, Germany, Aug. 2003.

[22] M. Baldi, G. Marchetto, Y. Ofek, "A Scalable Solution for Engineering Streaming Traffic in the Future Internet," *Computer Networks (COMNET)*, Vol. 51, No. 14, Oct 2007, pp 4092-4111.

[23] M. Baldi, G. Marchetto, G. Galante, F. Risso, R. Scopigno, F. Stirano, "Time Driven Priority Router Implementation and First Experiments," *IEEE Int. Conf. on Communications (ICC 2006)*, Istanbul (Turkey), June 2006.

[24] Symmetricom, "bc637PCI-U," [Online]. Available: http://www.symmttm.com/products_blt_bc637PCI-U.asp

[25] D. Agrawal, M. Baldi, M. Corrà, G. Fontana, G. Marchetto, V. T. Nguyen, Y. Ofek, D. Severina, H. T. Truong, and O. Zadedyurina, "Scalable Switching Testbed not Stopping the Serial Bit Stream," *IEE Int. Conf. on Communications (ICC 2007)*, June 2007.

[26] D. Agrawal, M. Baldi, M. Corrà, G. Fontana, G. Marchetto, V. T. Nguyen, Y. Ofek, D. Severina, H. T. Truong, and O. Zadedyurina, "A Scalable Approach for Supporting Streaming Media: Design, Implementation and Experiments,"*IEEE Int. Symp. On Computers and Communications*, July 2007.

[27] M. Baldi, M. Corrà, G. Fontana, G. Marchetto, Y. Ofek, D. Severina, O. Zadedyurina, "Scalable Fractional Lambda Switching: A Testbed," IEEE/OSA Journal of Optical Communications and Networking, Vol.3, No.5, pp.447-457, May 2011.

Mario Baldi holds an M.S. with honors (Summa Cum Laude) in Electrical Engineering (1993) and a Ph.D. in Computer and Systems Engineering (1998) both from Politecnico di Torino, Italy. He was Vice President for Protocol Architecture at Synchrodyne Networks, Inc., New York and Vice Dean of the PoliTong Sino-Italian Campus at Tongji University, Shanghai, China. On leave from his position as Associate Professor at the Department of Control and Computer Engineering of Politecnico di Torino, he is currently Senior Member of Technical Staff at Embrane, Inc., Santa Clara, Ca.

Guido Marchetto received the Ph.D. Degree in Computer and System Engineering in April 2008 and the M.S. Degree Summa Cum Laude in April 2004, both from Politecnico di Torino. He holds a post-doctoral position at the Department of Control and Computer Engineering of Politecnico di Torino.