Pipeline Forwarding of Packets Based on a Low-Accuracy Network-Distributed Common Time Reference

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

***Abstract*—Pipeline forwarding is a technology with the capability of providing both guaranteed quality of service (QoS) and scalability, two fundamental properties for the future Internet. Implementing pipeline forwarding requires network nodes to operate with a common time reference that, in existing literature, is considered to have relatively good accuracy and usually be derived from an external source, such as the GPS or Galileo. This is a major requirement possibly hindering the widespread deployment of this technology, notwithstanding its potential to enable a host of new applications. This paper describes and analyzes a solution for realizing pipeline forwarding based on a low-accuracy common time reference distributed through the network and presents experimental results obtained with a prototypal implementation of the proposed solution.**

***Index Terms*—Distribution of a common time reference, experiments on a network testbed, network synchronization, packet scheduling, pipeline forwarding.**

I. Introduction

TRAFFIC over the Internet continues to grow steadily. In particular, the percentage of traffic requiring quality of service (QoS) in terms of end-to-end delay and jitter has been increasing during the last few years. For example, some applications, such as multimedia ones, need a minimum level of service quality in order to operate properly.

Current approaches to offer controlled quality based on the differentiated services (DiffServ) model [1] combined with overprovisioning of resources cannot withstand a significant increase in the fraction of traffic with QoS requirements due to a combination of the following factors.

- Current approaches rely on the fundamental assumption that differentiated traffic must use only a small fraction of the network capacity. Consequently, the additional network capacity needed when traffic with QoS requirements grows is larger than the increase in (revenue-generating) traffic.
- Given that there are many indicators of technology having reached a point where it does not follow any more Moore's Law of a tenfold increase every 18 months, the additional processing and switching capacity required to follow the steep growth curve of Internet traffic with QoS requirements has a high cost.
- In a possible future scenario in which traffic with QoS requirements might dominate the Internet, the excess network capacity stemming from overprovisioning is likely to remain unused—i.e., not to yield any revenue.

In essence, the upgrade that a network infrastructure relying on the aforementioned approaches should undergo in order to support such traffic increase is most likely to result in costs larger than the economic benefits, i.e., additional revenue brought by such services. Hence, a solution that relies on a more efficient utilization of network resources—i.e., allowing for traffic with QoS requirements to use a large percentage of network capacity—is needed.

On the other hand, approaches based on the integrated services (IntServ) model [2], although somewhat more efficient in the utilization of network resources, have 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) [3], 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 [4], [5]— such as class-based queuing, weighted round-robin, and others—cannot combine optimal delay and resource utilization efficiently (see detailed discussion in [6]).

In summary, existing asynchronous packet-scheduling approaches either require (very) large amounts of network resources or cannot scale to high performance (multiterabit) routers and switches. *Pipeline forwarding (PF)* is a packet-scheduling technique that can satisfy such requirements thanks to its unique combination of simplicity and effectiveness by deploying a *global common time reference* (CTR) for shaping the traffic through the network. PF provides guaranteed quality of service and scalability, as it has been extensively studied both analytically and through simulations (see, for example, [7]–[9]) and experimentation [10], [11]. PF properties basically stem from the predictability it introduces in network operation, hence, on the service offered to packets traversing it. PF is currently deployed in an experimental testbed interconnecting Turin, Milan, and Trento, Italy; the impact of its hypothetical deployment in the network of an Internet service provider such as Telecom Italia has been assessed in the context of a project sponsored by Telecom Italia Labs [12], and its market potential as a commercial application of the Galileo

Manuscript received April 22, 2008; revised October 28, 2008; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor A. Orda. First published July 14, 2009; current version published December 16, 2009.

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).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2009.2015759

positioning system has been evaluated in the context of the Harrison Project funded by the Galileo Supervisory Authority.

Also, S&G Queuing [13] uses a time reference to drive packet forwarding in routers with FIFO-like scheduling complexity—i.e., the solution has the potential to scale to high-performance architectures. However, S&G Queuing relies on a "per-link" time reference derived from the transmitter end independent local clock. The variable drift of clocks used on various input links at a router can lead to the impossibility of maintaining the timing profile characterizing traffic at the network edge, which eventually results in variable delays, jitters, and, in the worst case, buffer overflows and packet loss.

Instead, PF is based on a time reference (CTR) common to all network nodes. Since in much previous work, including prototypal implementations, the CTR is derived from coordinated universal time (UTC), the technology is often referred to as *UTC-based pipeline forwarding*. If UTC is provided through an external channel (e.g., Global Positioning System (GPS) is used in the prototypes described in [10] and [11]) the system is said to be based on an *externally distributed CTR*. If an interswitch synchronization protocol is used to distribute a timing signal through the network (as proposed in [8], for example), the system is said to be based on a *network-distributed CTR*. In both cases, original PF operating principles (as defined in [8]) imply that the CTR error in different nodes be smaller than the PF operation time unit, which is called a time frame (TF).

Relying on such an accurate (either externally or network-distributed) CTR is a major requirement on network nodes and network operations that some see as a hurdle with the potential to hinder PF deployment. This motivates this work that proposes, analyzes, and reports on experiments with a PF implementation supporting a low-accuracy network-distributed CTR. Specifically, this paper makes the following contributions: 1) a solution for CTR distribution with minimum impact on system complexity is defined; 2) a set of operational rules to ensure proper PF operation with CTR error larger than one or more TFs is specified; 3) resulting buffering requirements are devised; and 4) consequences on the quality of the service provided in terms of delay and jitter are analyzed. Notice that although in this work 2)–4) are devised assuming the synchronization model underlying the CTR distribution solution at 1), they can be straightforwardly generalized to various CTR distribution alternatives; i.e., their relevance is not limited to the proposed CTR distribution solution.

In essence, the paper shows how minimal changes to the PF algorithm originally proposed enable proper operation with a low-accuracy network-distributed CTR. Although the proposed changes to PF are minimal—which contributes to the relevance of this work since they do not affect the system complexity—they have a major impact because PF deployability is greatly improved. In particular, given that the proposed CTR distribution solution can be implemented by a low-complexity software module, this work facilitates PF deployment in low-end network nodes, such as at the wired or wireless edge of the network. This is key to take full advantage of PF in terms of guaranteed QoS as its benefits can be fully enjoyed when it is deployed end to end [6].

After a short description of PF and its deployment options (Section II), the paper discusses network synchronization issues in general (Section III-A) and outlines the basic principles of the synchronization solution proposed for the distribution of the CTR through a network (Section III-B). In fact, Section III-B also sets the context for this work: a stable network scenario; i.e., changes in the availability of links and nodes (e.g., due to failures) are not taken into consideration here. Section IV analyzes the impact of a network-distributed CTR on the implementation and deployment of the PF scheduling algorithm. Various options for the distribution of the CTR and a proposed protocol are discussed in Section III, while experimental results on a testbed implementing the proposed solution are presented in Section V. The outcome of this work and future work directions are finally discussed in Section VI.

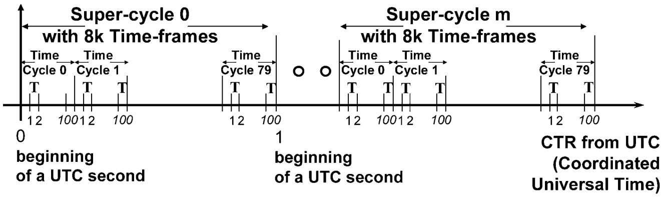

Fig. 1. Common time reference structure.

II. Underlying Principles and Technologies

As the context of this work is a network performing PF of packets, this section briefly introduces this technology and its deployment options. An extensive and detailed description of pipeline forwarding is outside the scope of this paper and is available in the literature [7]–[9].

A. Pipeline Forwarding

In PF, all packet switches utilize a basic time period called time frame (TF). The TF duration T may be derived, for example, as a fraction of the UTC second received from a time-distribution system such as the 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; this timing structure aligned in all nodes constitutes a CTR. Each super cycle might last one UTC second like, for example, in Fig. 1, where the 125-μs time frame duration T 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).

During a resource reservation phase, TFs are partially or totally reserved for each flow on the links of its route. Thus, TFs can be viewed as virtual containers for multiple packets that are switched and forwarded according to the CTR. In the PF deployment in the literature, 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. In another possible deployment, the reservation phase can be done on the fly before transmitting a packet without necessarily maintaining it across multiple TCs.

A signaling protocol must be chosen for performing resource reservation and TF scheduling; i.e., selecting the TF in which packets belonging to a given flow should be forwarded by each router. Existing standard protocols and formats should be used whenever possible. Many solutions have been proposed for distributed scheduling in pipeline forwarding networks [7] and the generalized MPLS (G-MPLS) control plane provides signaling protocols suitable for their implementation. In the traditional traffic management models for QoS support, such as ATM user–network interface and integrated services, applications signal their QoS requirements to the network for each flow (usually called microflow); queuing algorithms used in asynchronous packet switches have to maintain status information for each microflow, which is not scalable. Pipeline forwarding does not require per-microflow status in intermediate nodes, thus having similar provisioning scalability as the DiffServ model, where microflows are aggregated in the network to improve scalability [14].

The basic pipeline forwarding operation as originally proposed in [7] and [8] is regulated by two simple rules: 1) all packets that must be sent in TF k by a node must be in its output ports' buffers at the end of TF $k-1$, and 2) a packet p transmitted in TF k by a node N_n must be transmitted in TF $k+\alpha$ by the following node N_{n+1}, where α is a predefined integer called *forwarding delay*, and TF k and TF $k+\alpha$ are also referred to as the *forwarding TF* of packet p at node N_n and node N_{n+1}, respectively. It follows that packets are timely moved along their path and served at well-defined instants at each node. Nodes, therefore, operate as if they were part of a *pipeline*, from which the technology's name is derived. Consequently, given the TF at which a packet enters the network, the time at which the packet is forwarded by each node and eventually reaches its destination is known in advance with the accuracy of one TF.

The value of the forwarding delay is determined at resource-reservation time and must be large enough to satisfy 1). Note that the time a packet requires to go from the output buffer of a node to the output buffer of the following one is strictly dependent on the performance of both nodes and the distance between them. Thus, the minimum value acceptable for α could vary depending on the previous hop from which a packet is received. Defining A_n as the set of the neighbors of N_n; a set of different minimum acceptable forwarding delays $\alpha_{nm}, m : N_m \in A_n$ have to be defined for N_n.

PF guarantees that reserved real-time traffic experiences: 1) bounded end-to-end delay; 2) low delay jitter independent of the number of nodes traversed (less than two TFs when the CTR accuracy is smaller than a TF [8]); and 3) neither congestion nor resulting loss.

B. Deployment Options

Time-driven priority (TDP) [8] is a synchronous packet-scheduling technique that enables combining PF with conventional routing mechanisms to achieve high flexibility together with guaranteed service. While scheduling of packet transmission is driven by time, the output port can be selected according to either conventional IP destination-address-based routing, or multiprotocol label switching (MPLS), or any other technology of choice. Within a TF, packets can be switched and forwarded asynchronously, i.e., in an arbitrary order and to different output ports.

In *Time-driven switching* (TDS), originally proposed to realize sub-lambda or fractional lambda switching $(\mathrm{F}\lambda\mathrm{S})$ [9], all packets in the same TF are switched in the same way, i.e., altogether to the same output port. Consequently, header processing is not required, which results in low complexity (hence, high scalability) and enables optical implementation.

Although with a different degree of flexibility, both TDP and TDS can handle nonpipelined (e.g., best-effort) packets that can be transmitted during any unused portion of a TF, whether not reserved or reserved but actually unused.

III. Network Synchronization

A. An Overview

Several applications and technologies require network synchronization for their operation. These requirements are different depending on the specific environment. For example, a distributed software system may require a time-of-day synchronization in order to correctly perform transactions. The network time protocol (NTP) [15] is often used for this purpose; it carries timing information deployed by a software phase-locked loop (PLL) that maintains time-of-day synchronization by recovering the error on the system time introduced by the limited accuracy of the local oscillator. Current implementations of this type of network synchronization are based on an application layer protocol deployed by an application (daemon) process running on clients. SONET/SDH, on the contrary, needs synchronization at the physical layer in order to pace transmission of bits. The timing signal is distributed directly at the physical layer as defined by specific ITU standards.

TDP uses a CTR to determine when to transmit packets, i.e., packets must be sent out in predefined time-slots uniquely identified throughout the whole network. Similarly, in TDS, a CTR is deployed by all switches across the network to determine when to change their input–output interconnections. In particular, in a PC-based implementation of a TDP router [10], a periodic UTC-aligned signal generated (as an interrupt on the PCI bus) by a GPS receiver is used for indicating the beginning of a new TF, i.e., it triggers the transmission of packets scheduled for that TF. Analogously, the switch controller of a TDS switch [11] prototypal implementation uses a signal from a GPS receiver to trigger the reconfiguration of the switching fabric at the beginning of each TF according to a predefined, periodic pattern. Thus, PF requires time-of-day (here represented by the number of a TF within a TC) synchronization, which the GPS distributes with very high accuracy. However, the use of the GPS requires the deployment of GPS receivers (i.e., specific hardware) and the availability of a properly positioned outdoor antenna. Thus, a GPS-based synchronization solution is often impractical for logistics and cost reasons that some see as a drawback with the potential to hinder the deployment of PF. For these reasons, this work investigates PF operation based on a network-distributed CTR.

Several network synchronization techniques have been proposed, including the aforementioned NTP and SONET/SDH synchronization solution. Worth mentioning, IEEE 1588 [16] and Synchronous Ethernet [17] have recently been proposed specifically to provide synchronization in packet-switched networks. All these solutions aim at a very high accuracy, which results in high complexity and, in the case of SONET/SDH, the deployment of dedicated channels for carrying synchronization signals. However, in this paper, we show (see Section IV) that, unlike circuit-switching technologies, like SONET/SDH, PF does not require high accuracy in the realization of the CTR. In fact, since PF nodes handle packets, buffering can be leveraged on to relax accuracy requirements: An appropriate size buffer enables correct PF operation by delaying packets based on the relative accuracy of the time reference on neighboring nodes. For example, if the time reference of an upstream node is early with respect to its downstream neighbor, packets are buffered in the latter until their forwarding time according to local time reference. A late time reference of an upstream node with respect to its downstream neighbor can instead be dealt with by introducing a larger forwarding delay than required by the nominal packet transfer time between the two nodes, which implies additional buffering in the downstream node. Section IV is devoted to devising how a PF node can be dimensioned based on the CTR accuracy and proving that a properly dimensioned system provides the benefits typically offered by PF as originally defined in [7] and [8]. Consequently, the complexity of existing synchronization distribution solutions required to achieve high accuracy is not justified when aiming at PF deployment. For these reasons, a customized, low-complexity network synchronization solution is desirable.

An interswitch synchronization protocol proposed in [18] was specifically adapted to PF in [8]. This solution, aimed at a CTR error among nodes smaller than one TF, requires each node to have a local clock to trigger the beginning of each TF. The CTR distribution solution proposed in this work is based on directly triggering the beginning of a new TF on a node when a *synchronization signal* reaches such node. This protocol is simple and effective as it 1) does not rely on a local clock; hence enabling a 2) software-only implementation; 3) provides the required time-of-day synchronization; and 4) does not require dedicated network resources as the synchronization signal is piggybacked by data packets. Being simple and not requiring specific hardware, the proposed CTR distribution solution is particularly suitable for the deployment in low end nodes, such as at the edge of the network (e.g., home gateways and wireless access points). The following subsections present and analyze this protocol and its implications on the network synchronization.

B. Network Synchronization Model

The proposed method to achieve network synchronization consists in nodes distributing a synchronization signal to their neighbors that can be processed by receiving nodes and used to trigger the beginning of TFs. Since a node could have several neighbors, it could receive more than one timing signal. One of the neighbors is to be selected as the synchronization source for the node. The selection of the synchronization interface has to be done in such a way that each node has a *synchronization path* to a predefined node that acts as *time server* (i.e., a node that distributes a well-defined time reference at which it is synchronized and that becomes the common time reference for the entire network). This results in a logical tree topology, referred to as *synchronization tree*, built over the physical mesh network, as shown in Fig. 2. The root of this tree is named *synchronization signal server* (S3), and the interface from which a node acquires the synchronization signal is called *synchronization signal server port* (S3P). The establishment of this logical tree topology could be automated using several methods. For example, a customization of the spanning tree protocol or the information contained in a routing protocol database (e.g., the OSPF's) could be used. The definition of mechanisms and protocols for these purposes is outside the scope of this paper and left for future work.

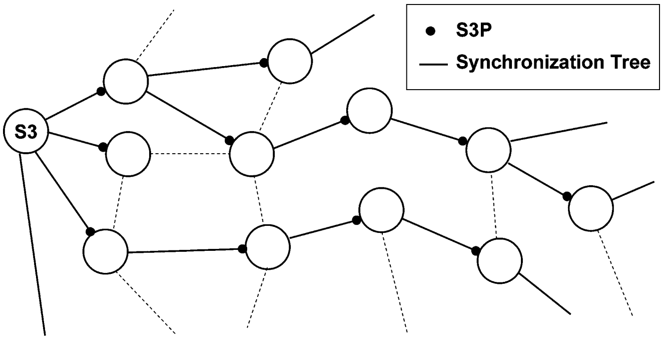

Fig. 2. Synchronization distribution model.

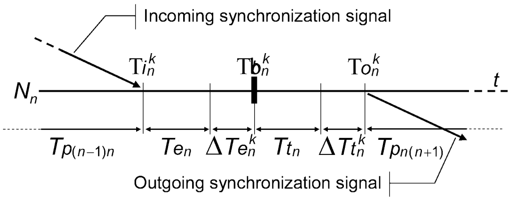

Fig. 3. Notation.

The resulting synchronization model consists of a synchronization signal generated by a server (the S3) spreading like a wave through the network and reaching all nodes. Since the synchronization signal experiences a nonzero propagation delay and, being network nodes nonideal, its transmission/reception/processing are affected by nonzero variable latencies, each TF features a *synchronization error*, i.e., a *variable* time difference between the beginning of the generic TF k at the S3 and the beginning of the same TF at the generic node N_n. The original PF algorithm was studied and developed under the assumption of all nodes sharing a, possibly UTC-aligned, CTR ensuring that TFs begin simultaneously on all nodes or with a difference across all nodes smaller than a TF [8]. Section IV will show that few minor modifications to the PF algorithm are actually sufficient to allow proper operation when network nodes are affected by synchronization errors of any magnitude. However, such modifications ensure proper PF operation if the synchronization error, and specifically its maximum variation, are known. Hence, the remainder of this section is devoted to the synchronization error analysis, under the assumption that the synchronization signal on reaching a node directly triggers the beginning of a new TF.[1]

Let (see Fig. 3):

- $\mathcal{T}i_n^k$ be the instant when the synchronization signal that determines the beginning of TF k reaches the S3P of the generic node N_n, i.e., the instant at which TF k should begin at N_n if there were no latencies.
- $\mathcal{T}b_n^k$ be the instant at which the TF k actually begins at N_n.
- $\mathcal{T}o_n^k$ be the instant at which the synchronization signal is transmitted by node N_n at the beginning of TF k.
- Te_n be the minimum time that N_n needs to react to the synchronization signal, which includes receiving and processing latencies.

[1] The following analysis can be easily extended to other approaches, such as synchronizing through the network a local clock that triggers the beginning of a new TF.

- ΔTe_n^k be the variable component of the time that N_n needs to react to the synchronization signal indicating the beginning of the generic TF k, where

$$0 \leq \Delta Te_n^k \leq \Delta Te_n^{\max} \forall k. \tag{1}$$

- Tt_n be the minimum time that N_n takes to output (i.e., to begin the transmission of) the synchronization signal corresponding to the beginning of a TF. Tt_n is the output side equivalent of the previously introduced Te_n and includes transmitting latencies.
- ΔTt_n^k be the variable component of the time that N_n takes to output the synchronization signal indicating the beginning of the generic TF k, where

$$0 \leq \Delta Tt_n^k \leq \Delta Tt_n^{\max} \forall k. \tag{2}$$

- Tp_{mn} be the propagation delay—considered constant—on the link connecting two adjiacent nodes N_m and N_n.
- Φ_n^k be the synchronization error concerning TF k affecting the generic node N_n. As defined above this equals the delay with which a TF begins at a node N_n with respect to the time at which the same TF begins at the S3, i.e., $\mathcal{T}b_n^k - \mathcal{T}b_{S3}^k$.

Let's consider a synchronization path in the network consisting of a sequence of nodes $\{N_0, N_1, \ldots, N_d\}$, where N_0 is the S3 from which the others receive the synchronization signal through the synchronization path. At N_0, the synchronization signal for a certain TF k starts as soon as the reference clock triggers the beginning of such a TF. Given the above described inaccuracies, the instant at which TF k begins at node N_d is

$$\mathcal{T}b_d^k = \mathcal{T}b_0^k + \sum_{i=0}^{d-1} Tp_{i(i+1)} + \sum_{i=0}^{d-1} Tt_i + \sum_{i=1}^{d} Te_i + \sum_{i=0}^{d-1} \Delta Tt_i^k + \sum_{i=1}^{d} \Delta Te_i^k. \tag{3}$$

That is, the respective synchronization error Φ_d^k consists of two components: a constant (i.e., time-invariant) one ϕ_d, and a time-variant one $\Delta\phi_d^k$:

$$\Phi_d^k = \phi_d + \Delta\phi_d^k \tag{4}$$

where

$$\phi_d = \sum_{i=0}^{d-1} \left(Tp_{i(i+1)} + Tt_i + Te_{i+1}\right) \tag{5}$$

and

$$\Delta\phi_d^k = \sum_{i=0}^{d-1} \Delta Tt_i^k + \sum_{i=1}^{d} \Delta Te_i^k$$
$$0 \leq \Delta\phi_d^k \leq \Delta\phi_d^{\max} = \sum_{i=0}^{d-1} \left(\Delta Tt_i^{\max} + \Delta Te_{i+1}^{\max}\right). \tag{6}$$

In conclusion, a generic node N_n is affected by a synchronization error Φ_n^k, which depends on its position along the synchronization path.

C. Synchronization Signal Transfer Options

Several solutions could be adopted in order to implement the presented CTR distribution method, specifically to transmit the synchronization signal. Some alternatives are presented and compared, leading to the TF delineation protocol described in Section III-D.

The synchronization signal could be transmitted at the *physical layer* using, for example, redundant codes in the line coding, modulation exceptions, or a dedicated wavelength on optical links. Physical layer operation results in a small synchronization error variation, i.e., considering a generic node $N_n, \Delta\phi_n^{\max}$ is limited as the uncertainties on the transmission and reception of the signal are small. However, this solution presents some drawbacks.

- It requires specific hardware, i.e., the logic handling the transmission and reception of the synchronization signal.
- Intermediate layer protocols have to be modified in order to allow the synchronization signal received at the physical layer to reach the PF scheduler at the protocol layer it is operating.
- The resulting PF implementation is not general and portable as it is dependent from both lower layer protocols and the availability of specific hardware.

Transmission of the synchronization signal at an *intermediate protocol layer* has all the drawbacks listed above, while lacking the advantages related to the small variation of the synchronization error. Consequently, although possible, a CTR distribution based on intermediate layer protocols is not a sensible solution unless implementation- or deployment-specific reasons suggest otherwise.

Alternatively, the synchronization signal can be implemented at the *layer at which PF is deployed* (e.g., IP). Two categories of approaches, requiring different types of information into packets, can be envisioned: 1) a time stamp (e.g., TF number and TC number) can be included in each packet, or 2) a TF delimiter can be transmitted at the beginning of each TF.

The former is more robust since a receiving node misses the beginning of a new TF only if all packets transmitted by an upstream node during that TF are lost. Moreover, even if this happens, the node recovers the correct TF from the information carried by packets received during the following TF. However, this solution 1) introduces transmission and processing overhead resulting from the 16-bit (or more) integer that represents the time stamp, and 2) requires modifications to the protocol headers as common protocols (e.g., IPv4, IPv6, Ethernet, MPLS) do not feature any field suitable for carrying such a time stamp.

The solution based on a TF delimiter can be implemented in various ways ranging from defining a control packet to be transmitted as a delimiter to setting a 1-bit field in the first packet transmitted during a TF. The latter introduces a very limited transmission and processing overhead, and it is not unlikely that an unused bit be found in an existing protocol header, thus not requiring major modifications to the standards. For example, a nonreserved codepoint of the DiffServ (DS) field could be used to implement CTR distribution with IP packets. The drawbacks of this solution are:

- sensitivity to packet loss—a node goes permanently out of synchronization when the TF delimiter is lost;
- an additional mechanism is needed at system startup to carry a time stamp allowing each node to initialize the TF and TC identity.

D. TF Delineation Protocol

In previous work [10], the authors defined a protocol for PF routers with an externally-distributed CTR to exchange timing information used for the evaluation of the forwarding TF on each node. Here, this protocol is proposed as a TF delineation protocol in a PF network based on a network-distributed CTR according to the synchronization model presented in Section III-B to implement an efficient and simple CTR distribution solution. The protocol is based on the combination of a robust TF delimiter and compressed time stamp, thus drawing from the strengths of the two solutions while avoiding their drawbacks, both highlighted in Section III-C. As presented in the following, the solution deploys the DS field of the IP header. However, it can be similarly implemented with other protocols (for example, the EXP field of the MPLS shim header can be analogously used).

Three bits of the DS field, i.e., eight unreserved DS codepoints, are used to carry the delimiter/compressed time stamp. Bits 0 × 0c are set in all PF packets to distinguish them from those not receiving PF service (e.g., best-effort or differentiated service packets), bit 0 × 10 is set to 1 (0) in packets transmitted during odd (even) TFs, and bits 0 × 20 and 0 × 40 toggle their value every TC and every super cycle, respectively. This results in an alternating-bit protocol for TF and TC identification.[2] PF routers maintain the number of the TF and TC during which the last received packet was transmitted by the upstream node. This information is updated every time the DS codepoint of a packet is different with respect to the previous packet. TF and TC initialization is performed by setting the TF and TC number to zero the first time the bit corresponding to super cycle (0 × 40) toggles. Consequently, system initialization lasts up to 1 s (i.e., the super cycle duration), but happens only when the router starts up and does not require transmission of additional information. When a node has no packets (including non-PF packets) to transmit during a TF on a given link, it sends sequences of *padding* IP packets with TF and TC marking for keeping the router at the other end synchronized.[3]

While timing information is coming from all interfaces, only the one received through the S3P is used by a router to derive the CTR, i.e., to trigger the beginning of a new TF.

Note that the proposed solution can be implemented with software-only components, thus enabling upgrade of existing equipment and reducing costs with respect to other solutions that are based on integrated circuits used to control local clocks.

IV. Packet-Scheduling Algorithm

Previous works on PF and current implementations are based on a UTC-aligned, accurate CTR, i.e., $\Phi_n^k = 0, \forall k, n$. Considering a network scenario where nodes are characterized by a variable $\Phi_n^k \neq 0$, PF properties, implementation, and deployment rules have to be reconsidered. The modifications required to a PF router initially implemented for and deployed with an externally distributed CTR are presented as an example.

Specifically, a PF router performs four fundamental PF-related steps: It 1) devises the TF during which each received packet was sent out by the previous node; 2) calculates the forwarding TF of the packet based on the predefined forwarding delay; 3) stores the packet in a queue corresponding to its forwarding TF; and 4) whenever a new TF begins, it transmits all the packets stored in the queue corresponding to the TF. The rules and constrains driving these steps are part of the PF scheduling algorithm and are presented in several publications [7], [8] for PF based on an accurate CTR. The following sections analyze the modifications required to such rules and constraints when routers deploy a low accuracy, possibly network-distributed CTR. Note that these results can be applied independently of the network synchronization distribution protocol deployed. The PF rules defined in [7] for a scenario with ideal CTR are generalized here for the case of any synchronization accuracy.

A. TF Duration

One implication of the synchronization error including a variable component is that the actual duration τ_n^k of a generic TF k at the generic node N_n as derived from the synchronization signal is not constant. The actual duration of TF k at node N_n is given by the difference between the beginning instants of TF $k+1$ and k; i.e., $\tau_n^k = \mathcal{T}b_n^{k+1} - \mathcal{T}b_n^k$, which, from the definition of Φ_n^k, can be expressed as

$$\tau_n^k = \mathcal{T}b_{S3}^{k+1} + \Delta\phi_n^{k+1} - \mathcal{T}b_{S3}^k - \Delta\phi_n^k. \tag{7}$$

If, as it is reasonable, the latencies in receiving and processing an external synchronization signal by an S3 are ignored, $\mathcal{T}b_{S3}^{k+1} - \mathcal{T}b_{S3}^k = \mathrm{T}, \forall k$. Thus, given that $\Delta\phi_n^{\min} = 0, \forall n$, we have that

$$\mathrm{T} - \Delta\phi_n^{\max} \leq \tau_n^k \leq \mathrm{T} + \Delta\phi_n^{\max}. \tag{8}$$

Since resource reservation is based on TF nominal duration T, a variable TF duration may result in the impossibility of keeping PF schedules during shorter TFs (i.e., some packets scheduled for a TFs cannot be transmitted because the TF finishes too early), with consequent possible packet backlog at the PF buffers, buffer overflow, and packet drops.

Guaranteeing deterministic quality of service—i.e., no loss and unpredictable delay and jitter due to network congestion—is possible by simply forwarding all packets that match the predefined schedule for TF k—i.e., that have been reserved resources during TF k—even if this requires extending the transmission beyond TF k end, i.e., after $\mathcal{T}b_n^{k+1}$. According to this new operation mode, the transmission of packets scheduled during a TF k ends 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 N_n, but also to a particular output interface.

Definition 1: In a PF node using a network-distributed CTR realized according to the presented synchronization mechanism,

[2] Such a 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. Also, the mechanism can be seen as delimiting the beginning of each TF by changing the DS codepoint.

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

the beginning of a new TF on an output interface is identified by the latest of the following events:

1) the synchronization signal is received at the S3P;
2) the output buffer corresponding to the current TF on the given interface becomes empty.

The above definition is coherent with the original definition of TF with an ideal CTR, in which case 1) is the possibly external timing signal triggering the beginning of a new TF and 2) is guaranteed to happen before such event. Moreover, Definition 1 can be modified to fit other CTR distribution solutions by substituting event 1) with whatever timing event triggers the beginning of a new TF.

Transmission of packets scheduled during a TF lasts at most T (resource reservation is based on this value), while the minimum TF duration, given by (8), is $T - \Delta\phi_n^{\max}$. Thus, the maximum time packet transmissions can continue beyond TF duration, namely after the arrival of a synchronization signal, is $T_{\text{ex}}^{\max} = \Delta\phi_n^{\max}$. This happens when a TF has minimum duration; the condition for such event can be derived from (6) as

$$\begin{cases} \Delta\phi_n^k = \Delta\phi_n^{\max} \\ \Delta\phi_n^{k+1} = \Delta\phi_n^{\min} = 0. \end{cases} \tag{9}$$

Consequently, the maximum error on the beginning of TF $k+1$, i.e., the maximum synchronization error, on a generic output interface of a generic node N_n according to Definition 1 is obtained by adding $T_{\text{ex}}^{\max}$ to the maximum synchronization error of TF $k+1$ as given by (4) and then applying (9), thus obtaining

$$\Phi_n^{\max} = \phi_n + \Delta\phi_n^{\min} + T_{\text{ex}}^{\max} = \phi_n + \Delta\phi_n^{\max}. \tag{10}$$

Thus, it can be concluded that the network synchronization model presented in Section III-B also applies when TFs comply with Definition 1, and the proposed definition of the TF beginning does not affect the maximum synchronization error. However, in this case, the network-distributed CTR features a different synchronization error for each interface. Equation (10) is extended as follows in order to capture this:

$$\Phi_{nm}^{\max} = \phi_{nm} + \Delta\phi_{nm}^{\max}, \quad m : N_m \in A_n \tag{11}$$

where A_n denotes the set of the neighbors of the generic node N_n, as defined in Section II-A, and nm refers to the interface of N_n connected to the link to N_m.

B. Forwarding TF Evaluation

As discussed in Section II-A, PF operation determines a dependency among the forwarding TFs for each packet in all the nodes across the network. The forwarding TF at a generic node N_n can be expressed, in accordance to [8], as

$$F_n = F_{n-1} + \alpha_{(n-1)n} + f \tag{12}$$

where:

- F_n is the forwarding TF of the packet at the generic node N_n;
- F_{n-1} is the forwarding TF of the packet at the previous node N_{n-1} on the path of the packet;[4]
- $\alpha_{(n-1)n}$ is the minimum acceptable forwarding delay (introduced in Section II-A) between node N_{n-1} and node N_n. In order to make sure that packets are already in the output buffer of node N_n when their forwarding TF begins, the forwarding delay must be greater than or equal to the sum of the propagation delay on the link connecting the nodes, the processing time, and additional latencies that characterize both nodes. Given N_{n-1} and N_n, we can express this sum as

$$D_{(n-1)n}^{F_n} = Tt_{n-1} + \Delta Tt_{n-1}^{F_{n-1}} + Tp_{(n-1)n} + Te_n + \Delta Te_n^{F_n} \tag{13}$$

  which, considering the worst case $D_{(n-1)n}^{\max} = Tt_{n-1} + \Delta Tt_{n-1}^{\max} + Tp_{(n-1)n} + Te_n + \Delta Te_n^{\max}$, leads to

$$\alpha_{(n-1)n} = \left\lceil \frac{D_{(n-1)n}^{\max}}{\mathrm{T}} \right\rceil. \tag{14}$$

- f models the adopted forwarding scheme [7]. $f = 0$ represents *immediate forwarding* operation, i.e., applying the minimum acceptable forwarding delay. $f \in \mathbb{N}^+$ models *nonimmediate forwarding* operation, i.e., deploying a larger forwarding delay, which enables reducing blocking probability at the expenses of implementation complexity by not necessarily forwarding a packet as soon as it is available at the output port [7], [9].

The above forwarding TF calculation method refers to the case where all network nodes are perfectly synchronized, i.e., $\Phi_n^k = 0, \ \forall k, n$. If the CTR is distributed through the network using the presented technique, 1) TFs do not begin at the same time on all nodes, 2) the synchronization error is different at each node (i.e., $\Phi_n^k \neq \Phi_m^k$ if $N_n \neq N_m$), and 3) both TF alignment and synchronization error vary in time (i.e., $\Delta\phi_n^k \neq \Delta\phi_n^{k'}$, if $k \neq k'$). This has to be considered in the forwarding TF calculation.

Theorem 1: In a PF network where the CTR is distributed to nodes with a nonzero synchronization error, proper PF operation is ensured when the forwarding TF for a packet is calculated at the generic node N_n as

$$F_n = F_{n-1} + \alpha_{(n-1)n}^* + f \tag{15}$$

where

$$\alpha_{(n-1)n}^* = \left\lceil \frac{\Phi_{n-1}^{\max} - \Phi_n^{\min} + D_{(n-1)n}^{\max}}{\mathrm{T}} \right\rceil. \tag{16}$$

Proof: Given a forwarding TF F_n at a node N_n, a necessary and sufficient condition that guarantees the PF algorithm to work properly is that the time at which the packet transmission is scheduled at node N_n, i.e., the time (corresponding to $\mathcal{T}o_n^{F_n}$) at which transmissions for TF k begins at node N_n, follows the time at which the packet enters the output buffer of the node (denoted as $\mathcal{T}ib_n^{F_n}$), i.e.,

$$\mathcal{T}o_n^{F_n} \geq \mathcal{T}ib_n^{F_n}. \tag{17}$$

Given the definition of $\mathcal{T}b_n^{Fn}$ in Section III-B, we can write

$$\mathcal{T}o_n^{F_n} = \mathcal{T}b_{S3}^{F_n} + \Phi_n^{F_n} \tag{18}$$

and

$$\mathcal{T}ib_n^{F_n} = \mathcal{T}b_{S3}^{F_{n-1}} + \Phi_{n-1}^{F_{n-1}} + D_{(n-1)n}^{F_{n-1}}. \tag{19}$$

[4] Although F_n and F_{n-1} are packet-dependent, the packet is not explicitly indicated to simplify the notation.

Since the inequality (17) has to hold for every value of $\Phi_n^{F_n}$ and $\Phi_{n-1}^{F_{n-1}}$, specifically for the worst case, we can derive

$$\mathcal{T}b_{S3}^{F_n} + \Phi_n^{\min} \geq \mathcal{T}b_{S3}^{F_{n-1}} + \Phi_{n-1}^{\max} + D_{(n-1)n}^{\max}$$

and converting in TFs as a time measurement unit, we obtain

$$F_n \geq F_{n-1} + \left\lceil \frac{\Phi_{n-1}^{\max} - \Phi_n^{\min} + D_{(n-1)n}^{\max}}{\mathrm{T}} \right\rceil .$$

□

Note that (14) can be derived from (16) with $\Phi_{n-1}^{F_{n-1}} = \Phi_n^{F_n} = 0$.

C. Buffer Dimensioning

Buffers have to be properly dimensioned in network nodes to guarantee that no packet is lost when nodes perform PF. The additional delay packets incur due to the deployment of a network-distributed CTR has to be taken into account when dimensioning the buffers.

Theorem 2: Let R be the output link capacity. The size of the buffer on the output interface of a node N_n that guarantees no loss for a pipelined packet is

$$\mathrm{Buff}_n = \left(\alpha_{(n-1)n}^{*} + \beta_{(n-1)n} + f\right) \cdot \mathrm{T} \cdot R, \qquad (20)$$

where, defining $D_{(n-1)n}^{\min} = Tt_{n-1} + \Delta Tt_{n-1}^{\min} + Tp_{(n-1)n} + Te_n + \Delta Te_n^{\min}$

$$\beta_{(n-1)n} = \left\lceil \frac{\Phi_n^{\max} - \Phi_{n-1}^{\min} - D_{(n-1)n}^{\min}}{\mathrm{T}} \right\rceil . \qquad (21)$$

Proof: For each TF, PF output buffers on node N_n have to store packets from the instant (denoted as $\mathcal{T}ib_n^k$ for the generic TF k) they enter the buffer of the node to the moment they begin to be sent out ($\mathcal{T}o_n^k$ for TF k).[5] The maximum difference between such instants over time is

$$\max_k \left(\mathcal{T}o_n^k - \mathcal{T}ib_n^k\right) . \qquad (22)$$

Considering that 1) a queue must be associated to each TF, 2) its size must have sufficient capacity to contain the total amount of bytes that can be transmitted during such a TF (i.e., $\mathrm{T} \cdot R$ bits), and 3) each queue can be reused (i.e., associated to another TF) as soon as the associated TF is over and all its packets have been transmitted, the minimum total number of required queues is given by the minimum number of TFs N_{TF} such that their total duration is longer than (22), i.e.,

$$N_{\mathrm{TF}} = \left\lceil \frac{\max_k \left(\mathcal{T}o_n^k - \mathcal{T}ib_n^k\right)}{\mathrm{T}} \right\rceil .$$

Considering that by definition $\mathcal{T}b_{S3}^k/\mathrm{T} = k$, from (15), (18), and (19) we can derive

$$N_{\mathrm{TF}} = \alpha_{(n-1)n}^{*} + \beta_{(n-1)n} + f. \qquad (23)$$

Given that each queue should be capable of storing $\mathrm{T} \cdot R$ bits, the total buffer requirement is

$$\mathrm{Buff}_n = N_{\mathrm{TF}} \cdot \mathrm{T} \cdot R$$

Note that (20) and (21) are valid also in case $\Phi_n^k = 0,\ \forall n, k$, i.e., when an externally-distributed CTR is deployed all over the network. □

Moreover, adding an extra queue avoids concurrent reading and writing access to memory, thus eliminating the need of a 2–speed up of memory access speed, which can result in a significant cost cut for high-speed interfaces

$$\mathrm{Buff}_n' = \left(\alpha_{(n-1)n}^{*} + \beta_{(n-1)n} + f + 1\right) \cdot \mathrm{T} \cdot R. \qquad (24)$$

Lemma 1: Lossless PF with a network-distributed CTR on the path $\{N_0, N_1, \ldots, N_H\}$ is ensured by deploying on the output interface of each node N_n a buffer of size shown in (25) at the bottom of the page.

Proof: By substituting $\alpha_{(n-1)n}^{*}$ as derived from (16) in (23) and considering that either $\lceil x \rceil + \lceil y \rceil = \lceil x + y \rceil$ or $\lceil x \rceil + \lceil y \rceil = \lceil x + y \rceil + 1$, in the most conservative case, the number of queues guaranteeing lossless PF operation according to (23) is shown in the last equation at the bottom of the page.

Further considering the definitions given in (1), (2), (4), and (6) we obtain

$$N_{\mathrm{TF}} = \left\lceil \frac{\Delta\phi_{n-1}^{\max} + \Delta\phi_n^{\max} + \Delta Tt_{n-1}^{\max} + \Delta Te_n^{\max}}{\mathrm{T}} \right\rceil + f + 1. \qquad (26)$$

The corresponding amount of buffering can be derived by considering the amount of bits that can be transmitted during one TF, i.e., $\mathrm{T} \cdot R$. □

From (25), the following can be observed.

1) Buffering at a node N_n depends on the maximum variation of timing parameters—specifically the synchronization error and the input latency at the node (i.e., $\Delta\phi_n^{\max}$ and $\Delta Te_n^{\max}$) and the synchronization error and the output

$$\mathrm{Buff}_n = \left(\left\lceil \frac{\Delta\phi_{n-1}^{\max} + \Delta\phi_n^{\max} + \Delta Tt_{n-1}^{\max} + \Delta Te_n^{\max}}{\mathrm{T}} \right\rceil + f + 1 \right) \cdot \mathrm{T} \cdot R \qquad (25)$$

$$N_{\mathrm{TF}} = \left\lceil \frac{\Phi_{n-1}^{\max} - \Phi_{n-1}^{\min} + \Phi_n^{\max} - \Phi_n^{\min} + D_{(n-1)n}^{\max} - D_{(n-1)n}^{\min}}{\mathrm{T}} \right\rceil + f + 1$$

[5] As it is common in router implementations, an additional transmission buffer of size $\mathrm{T} \cdot R$ is provided at the lower protocol layer to store packets as they get transmitted.

latency at the previous node on the forwarding path of the packet (i.e., $\Delta\phi_{n-1}^{\max}$ and $\Delta Tt_{n-1}^{\max}$)—and not on their absolute values.

2) Buffering (as well as the forwarding TF) at a generic node N_n depends on timing parameters at both the node itself and the previous node N_{n-1} on the path of the packet. It can thus be concluded that in order to guarantee lossless service in a PF node N_n, each output interface must be equipped with a buffer of the size shown in (27) at the bottom of the page.

The above is derived by generalizing (25) to take into account the neighbor featuring highest variability of its timing parameters.

D. Delay and Jitter Analysis

A comparison of (14) and (16) shows that the synchronization error impacts directly on the delay introduced by each node; i.e., simple and low-accuracy network-distribution of the CTR is possible at the expenses of increased end-to-end delay and jitter. By analyzing (5) and (6), it can be noted that such increase grows with the number or nodes through which the CTR is distributed—i.e., the sparser S3s, the higher the end-to-end delay and jitter introduced by the network on an average flow. In particular, the time spent by a packet in the output queue of a PF node varies between 0 (when the packet arrives just before the beginning of its forwarding TF) and $\max_k(\mathcal{T}o_n^k - \mathcal{T}ib_n^k)$, as given by (22). The maximum jitter J experienced by the packet through a PF network is given by the maximum time spent in the buffer of the last node on its path $\{N_0, N_1, \ldots, N_H\}$, i.e., from Theorem 2 and (22)

$$J = \left(\alpha^*_{(H-1)H} + \beta_{(H-1)H} + f\right) \cdot \mathrm{T}. \tag{28}$$

Furthermore, the total maximum buffering delay experienced by a packet along the path $\{N_0, N_1, \ldots, N_H\}$ is

$$\sum_{h=0}^{H} \max_k \left(\mathcal{T}o_h^k - \mathcal{T}ib_h^k\right) = \sum_{h=0}^{H} \mathrm{Buff}_h / R. \tag{29}$$

E. Discussion

Previous subsections gave the guidelines to implement PF when the CTR is distributed through the network with arbitrarily low accuracy. Equations (16) and (27) provide the guidelines to dimension a network node so that proper operation is guaranteed by keeping into account maximum and minimum synchronization error at each node. While synchronization over a traditional packet network is significantly affected by queuing delay, which is hard to bound and estimate, over a PF network with the proposed CTR distribution method, the main causes for the synchronization error Φ_n^k at a generic node N_n are 1) nonzero propagation delay, 2) nonzero packet processing time of transmission and reception modules, and—especially—3) their variability that depends on issues ranging from hardware components to system architecture to software implementation.

Providing a reliable estimate of the minimum synchronization error is not critical, as it could in principle be set to 0 or, when the contribution of the propagation delay is significant (i.e., in a long-haul link scenario), to the propagation delay. Also, providing an upper bound on the variation of the propagation delay, which is due to temperature fluctuations caused by changing weather conditions or wear and tear of the medium, is not critical. Propagation delay variations can be accounted for by allowing a safety margin of k TFs in the forwarding delay. k depends on the TF duration, is likely to be 1 in most practical cases, and anyway does not have a significant impact on the end-to-end delay because the propagation delay variation is much smaller than packet-processing delay and the other components of the synchronization error.

The packet processing time introduced by hardware/software modules that perform transmission, reception, and handling of packets (such as the PCI bus or the Ethernet NIC in the PC-based prototype deployed in the experiments reported in the following section) is more critical since it has significant relative and absolute variations. However, proper design and implementation of the router can ensure such time to be bounded, and its value can be devised based on either the system design or experimental characterization through specifically targeted lab tests and measurements.

Since, in the implementation used in this work, the synchronization signal is conveyed in network layer packets, their transmission delay also contributes to the synchronization error. Although varying, this is not critical, as it is obviously bounded by the time required to transmit a maximum transmission unit (MTU) as defined for the specific data link protocol deployed.

If the forwarding delay resulting from an estimate of the maximum synchronization error is not appropriate, the deterministic operation of PF is affected. Deployment of a more sophisticated CTR distribution solution, e.g., deploying a local clock to smooth out the variations of the synchronization error, would ensure that it stays within the estimate, hence ensuring deterministic operation with the proposed modified PF operation. Moreover, the disruption of the deterministic service is temporary, and proper operation is automatically resumed after the first TF not fully utilized to transmit pipelined packets, as ensured by the proposed PF algorithm for low-accuracy CTR.

Improvements to the proposed CTR distribution solution are possible, at the expense of increased complexity, to:

- reduce the synchronization error, and consequently packet delay and jitter resulting from PF (which Section IV-D showed to be dependent on the synchronization error);
- avoid deterministic service being disrupted if the estimate on the maximum processing and propagation delay, i.e., on the maximum synchronization error, is exceeded;

$$\mathrm{Buff}_n = \left(\left\lceil \frac{\max_{m, N_m \in A_n} \Delta\phi_m^{\max} + \max_{m, N_m \in A_n} \Delta Tt_m^{\max} + \Delta\phi_n^{\max} + \Delta Te_n^{\max}}{\mathrm{T}} \right\rceil + f + 1\right) \cdot \mathrm{T} \cdot R \tag{27}$$

- ensure proper CTR distribution and seamless PF operation in case of link and nodes failures.

For instance, the timing information received from all the input links of a node could be used as synchronization signals, along the lines of the solution proposed in [19]. Alternatively, a local clock synchronized with the synchronization signals by means of a PLL could be deployed. However, this is outside the scope of this paper that aims at showing how PF, with minimal changes to the original algorithms, can properly operate with a low-accuracy CTR, even if distributed through the network with a low-complexity protocol. A major outcome of this section is the general validity, i.e., with any synchronization mechanism, of the proposed modifications to the original PF algorithms and system dimensioning as expressed by (16) and (27). However, an analysis of the performance and properties of PF with more sophisticated (and more complex) CTR distribution solutions is left for future work.

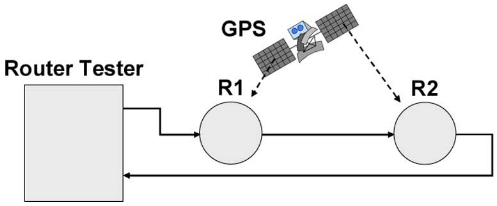

Fig. 4. Synchronization error evaluation testbed.

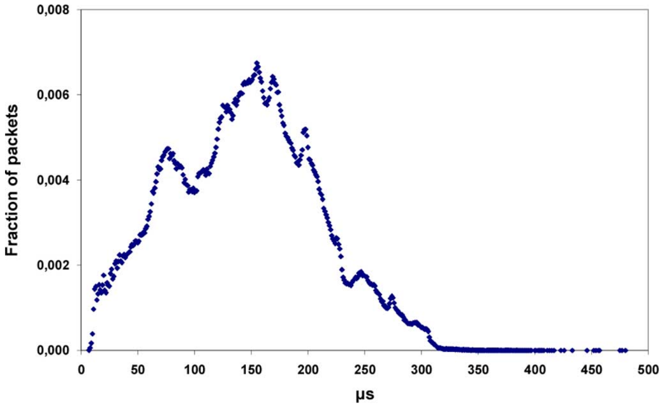

Fig. 5. Potential synchronization error distribution.

V. Experimental Results

Some experiments were run on a testbed of TDP routers implemented by 2.4-GHz Pentium IV PCs running a modified version [10] of the FreeBSD 4.8 routing software. The results demonstrate the effectiveness of the CTR distribution solution proposed in Section III-D and validate the analysis of the modified PF operation presented in Section IV. Results obtained from the experiments are then extended to a large-scale network by applying the equations proven in Section IV, which demonstrates the feasibility and the effectiveness of the proposed method in an arbitrary network.

A. *Synchronization Error Measurement*

If the proposed CTR distribution method is used, the main causes for synchronization error are propagation delay, packet processing time, and their variability that in our PC-based router is due to 1) access to shared resources, such as CPU, memory, communication buses, etc., and 2) the interrupt-driven nature of the FreeBSD kernel. Thus, the resulting synchronization error is, in this case, expected to be particularly large and variable under high traffic load. However, this provides a good reference point since it can be considered as a worst-case scenario in which to experiment with the proposed solution. In fact, special purpose routers usually deployed in real networks are designed to minimize packet handling time and, consequently, its variations.

A first set of experiments was run to measure the various system latencies and devise the synchronization error components introduced by our prototypal network nodes in order to get an idea of its order of magnitude. Fig. 4 shows the measurement setup. An Agilent N2X Router Tester is used to generate a traffic flow that enters TDP router R1, is forwarded to TDP router R2, and then is routed back to the router tester, all across FastEthernet links (100 Mb/s). In this first set of experiments, R1 and R2 execute the TDP scheduling algorithm with an externally distributed CTR that is acquired through a GPS receiver; i.e., TFs on both routers are aligned with UTC. Time from the GPS receivers is also used to measure the interval between the beginning of a packet transmission at the TDP scheduler on R1 until the packet is processed at the IP layer (i.e., where the DS field is processed) on R2. In a network-distributed CTR scenario in which R1 is an S3 and the interface of R2 toward R1 is its S3P, the measured time interval represents the synchronization error at R2, which includes the various contributions identified in Section III-B, specifically

$$\Phi_{\mathrm{R2}} = Tp_{\mathrm{R1R2}} + Tt_{\mathrm{R1}} + Te_{\mathrm{R2}} + \Delta Tt_{\mathrm{R1}} + \Delta Te_{\mathrm{R2}} \qquad (30)$$

where Tp_{R1R2} is negligible for all purposes, having used a short cable between the routers stacked one on top of the other.

Fig. 5 plots the distribution of the synchronization error that the prototypal router potentially introduces. The various components of the synchronization error are measured over 100 test runs with fully loaded links and various packet lengths. In particular, the router tester generates 10 10-Mb/s CBR UDP flows with constant message size on its port connected to R1 fully loading (as a 100-Mb/s aggregate flow) the links. The size of the IP packets is varied in each test run. Sample configurations include: all flows deploying the minimum size of 64 bytes (corresponding to about 20 000 offered packets per second), all flows deploying the maximum size of 1500 bytes (about 800 packets per second), all flows deploying different packet sizes variably chosen between the above minimum and maximum. Each test lasts 15 min; hence, the number of observed packets ranges between about 720 000 and about 18 000 000. The maximum and minimum synchronization errors measured are

$$\begin{aligned} \Phi^{\min} &= \Phi_{\mathrm{R2}}^{\min} = 7\,\mu\mathrm{s} \\ \Phi^{\max} &= \Phi_{\mathrm{R2}}^{\max} = 480\,\mu\mathrm{s}. \end{aligned} \qquad (31)$$

In order to properly configure the forwarding delay and buffering space in all nodes of the testbed network deployed for the experiments reported in Section V-B, the synchronization error at each node should be devised. Since all nodes are based on the same architecture and all links have the same length, measurements done on one of the nodes are, in all likelihood,

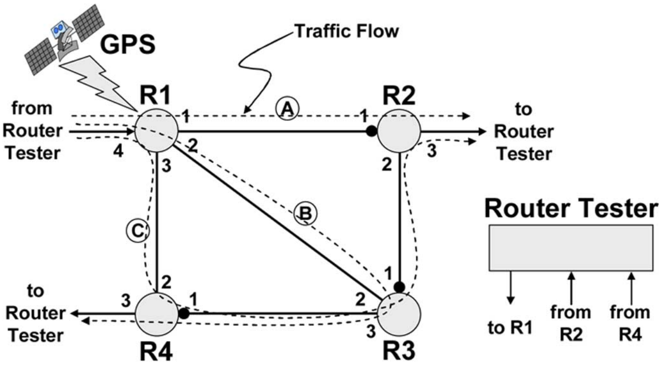

Fig. 6. Testbed.

TABLE I
FORWARDING DELAY AND BUFFER SIZE

Router	Input NIC	Forwarding Delay [TF]	Buffering [byte]
R1	1	4	31250
	2	6	
	3	8	
R2	1	2	31250
	2	6	
R3	1	4	43750
	2	2	
	3	8	
R4	1	6	43750
	2	2	

representative of all the others. Consequently, the synchronization error at a node d hops from the S3 on the synchronization path can be derived from (4), considering that each of the d upstream nodes features a synchronization error characterized according to (31)

$$\Phi_d^{\min} \approx d \cdot \Phi^{\min}$$
$$\Phi_d^{\max} \approx d \cdot \Phi^{\max}. \tag{32}$$

B. Experiments With Network-Distributed CTR

Fig. 6 shows the network testbed deployed in the experiments that involves fourTDP routers connected by 100-Mb/s Ethernet links with 250-μs TFs (i.e., $T = 250\,\mu$s). R1 acquires UTC from the GPS and is an S3; R2, R3, and R4 acquire the CTR through the network using the CTR distribution solution (Sections III-B and III-D) presented in this paper. Each router should select as S3P its interface connected to R1 in order to minimize the network synchronization error. However, since the presented experiments aim at assessing CTR distribution over multiple hops—as it would be in a real work network—S3Ps have been selected differently and are identified by a solid circle in Fig. 6.

For the sake of brevity and without loss of generality, only experiments with TDP immediate forwarding (i.e., $f = 0$) are reported here. The forwarding delay for each input interface and the buffer size required on each output interface is derived from (16) and (27), respectively, by using the network synchronization error figures devised with the first set of experiments—i.e., (31) and (32). Since all nodes are based on the same architecture and all links have the same length, $\alpha_{(n-1)n}$ is the same for every pair of nodes in the testbed and, from (13), $D^{\max}_{(n-1)n} = \Phi^{\max}, D^{\min}_{(n-1)n} = \Phi^{\min}$, and, consequently, $\Delta Tt^{\max}_m + \Delta Te^{\max}_n = \Phi^{\max} - \Phi^{\min}$, $\forall m, N_m \in A_n$, where A_n is the set of nodes directly connected to N_n. Although not necessary in a PC-based monoprocessor router, the additional buffering to avoid concurrent read/write access as specified in (24) was also considered so that experiments are run with the worst-case delay scenario. The resulting system parameters are summarized in Table I. The detailed calculation of the parameters of R2 is reported in the following as an example.

- *Forwarding delay*. R1 acquires UTC from the GPS, hence $\Phi^{\max}_{\mathrm{R1}} = 0$. From (31) and (32), $\Phi^{\min}_{\mathrm{R2}} = 7\ \mu$s and $\Phi^{\max}_{\mathrm{R3}} = 2 \cdot 480 = 960\ \mu$s. Furthermore, as described before, $D^{\max}_{\mathrm{R1R2}} = D^{\max}_{\mathrm{R3R2}} = 480\ \mu$s. This leads to

$$\alpha^*_{\mathrm{R1R2}} = \left\lceil \frac{\Phi^{\max}_{\mathrm{R1}} - \Phi^{\min}_{\mathrm{R2}} + D^{\max}_{\mathrm{R1R2}}}{\mathrm{T}} \right\rceil = 2\ \mathrm{TF},$$
$$\alpha^*_{\mathrm{R3R2}} = \left\lceil \frac{\Phi^{\max}_{\mathrm{R3}} - \Phi^{\min}_{\mathrm{R2}} + D^{\max}_{\mathrm{R3R2}}}{\mathrm{T}} \right\rceil = 6\ \mathrm{TF}.$$

- *Buffering*. Among the nodes directly connected to R2, R3 is affected by the largest synchronization error ($\Delta\phi^{\max}_{\mathrm{R3}} = \Phi^{\max}_{\mathrm{R3}} - \Phi^{\min}_{\mathrm{R3}} = 2 \cdot (480 - 7) = 946\ \mu$s); hence, from (27), it can be derived

$$\mathrm{Buff}_{\mathrm{R2}} = \left(\left\lceil \frac{\Delta\phi^{\max}_{\mathrm{R3}} + \Delta\phi^{\max}_{\mathrm{R2}} + \Phi^{\max} - \Phi^{\min}}{\mathrm{T}} \right\rceil + 1 \right) \cdot \mathrm{T} \cdot R$$
$$= 28125 \text{ bytes}$$

and, from (24),

$$\mathrm{Buff}'_{\mathrm{R2}} = \mathrm{Buff}_{\mathrm{R2}} + \mathrm{T} \cdot R = 31250 \text{ bytes}.$$

The router tester generates three 100-Mb/s UDP flows on the Gigabit Ethernet link to R1. Since the three flows are routed as shown by the dotted lines in Fig. 6, each link between TDP routers is fully loaded. In order to simplify resource reservation (manually performed in our prototypal implementation) and without losing generality, packet size is programmed to periodically vary among the four predefined values 64, 260, 625, and 1041 bytes, which result in 48, 12, 5, and 3 packets contained in each TF, respectively. The paths of the three flows realize every possible scenario a node can be faced with concerning the relation between data traffic and CTR distribution—for example, packets received by a node N_n from a node N_{n-1} where the synchronization error Φ^k_{n-1} is smaller/greater than Φ^k_n, packets received from the S3P, etc. The jitter is measured on each flow (see Table II) during 10 different tests, each one lasting two days. No packet is lost during the experiments, and the jitter does not exceed its analytical upper bound given by (28), which validates the analysis presented in Section IV.[6] As an example, the maximum measured jitter for flow A (730 μs) is smaller than the analytical bound $J = 4 \cdot \mathrm{T} = 1$ ms obtained by substituting $\alpha^*_{(H-1)H} = \alpha^*_{\mathrm{R1R2}} = 2$ TF and

[6] The presented experiments focus on assessing exclusively PF operation with network-distributed CTR and the CTR distribution solution presented in this paper. A general evaluation of PF, its properties, and benefits stemming from its deployment are outside the focus of this paper and were the subject of previous work (see, for example, [6]–[11]).

TABLE II
MAXIMUM JITTER WITH A NETWORK-DISTRIBUTED CTR

Traffic Flow	Max measured [µs]	Analytical bound [µs]
A	730	1000
B	916	3000
C	753	2000

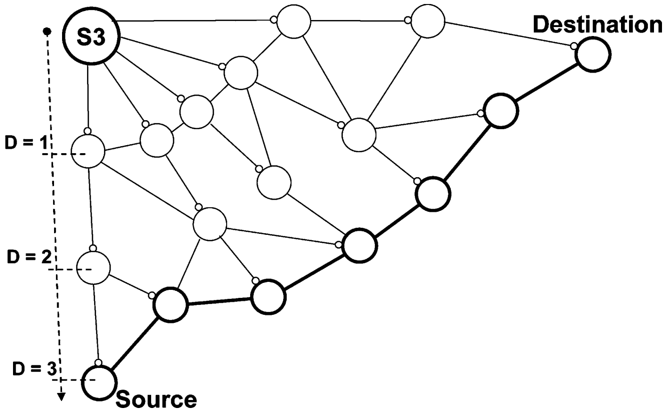

Fig. 7. Slowest path.

$\beta_{(H-1)H} = \beta_{\mathrm{R1R2}} = 2$ TF ($f = 0$ in all our experiments) in (28).

C. Delay and Buffering

Due to the toy network on which they had been devised, the results presented so far do not demonstrate a large-scale deployment of PF with a network-distributed CTR. However, they enable us to validate the presented analysis and devise synchronization error bounds for the prototype router that are given in (31). By applying (16), (27), and (29), the measurements on the testbed in Fig. 6 can be used to evaluate the buffering requirements and the maximum delay experienced by packets over an arbitrary network composed of PC-based prototypal routers. Assuming a network in which the maximum number of hops between any node and an S3 is $\mathrm{D} = 3$, the largest buffer size and buffering delay are the ones on a path traversing only routers at maximum distance D from the S3. Fig. 7 shows an example of such path, which we call the *slowest path*, for $\mathrm{D} = 3$ hops that traverses 6 PF hops, i.e., six links on which PF of packets is performed.

Fig. 8 plots the maximum end-to-end buffering delay [devised using (29)] on a slowest path versus the number of hops H on the path for several values of D. The buffer size (reported in Fig. 10) is calculated according to the procedure detailed in Section V-B (the additional buffer to avoid concurrent read/write access, as discussed in Section IV-C, is not considered here) noticing that, according to (31) and (32), the maximum and the minimum synchronization errors of nodes at a distance D from the S3 are $\mathrm{D} \cdot \Phi^{\max}$ and $\mathrm{D} \cdot \Phi^{\min}$, respectively. The maximum end-to-end buffering delay is devised by applying $\mathrm{H} \cdot (\mathrm{Buff}_H / R)$, derived from (29) by considering that Buff_h is the same for every node N_h in the path as all nodes are at distance D from the S3. For example, $\mathrm{D} \cdot \Phi^{\max} = 1.44$ ms and $\mathrm{D} \cdot \Phi^{\min} = 21\ \mu$s if $\mathrm{D} = 3$ and $\Phi^{\max}$ and $\Phi^{\min}$ assume the values devised for our prototypal router and provided by (31). Hence, the buffering delay experienced after $\mathrm{H} = 5$ hops by a packet traveling on a path whose nodes are at distance $\mathrm{D} = 3$ from the S3 is equal to $5 \cdot (\lceil (1419 + 1419 + 473)/\mathrm{T} \rceil + 1) \cdot \mathrm{T} = 18.75$ ms.

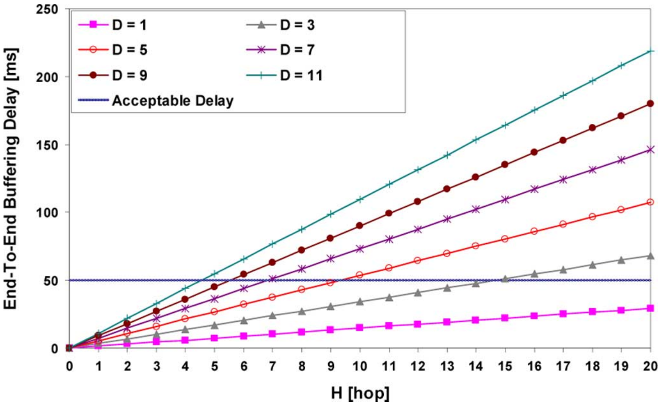

Fig. 8. Maximum end-to-end buffering delay on the slowest path ($\Phi_i^{\min} = 7\ \mu$s, $\Phi_i^{\max} = 480\ \mu$s, $\mathrm{T} = 250\ \mu$s).

Considering that the total end-to-end delay additionally includes propagation delays, we set the maximum acceptable end-to-end buffering delay to be 50 ms (dashed line). Fig. 8 shows that only small distances from the S3 (i.e., low values of D) can guarantee this bound on slowest paths composed by a reasonable number of hops. For example, if $\mathrm{D} = 5$, packets can traverse 9 slowest path hops before exceeding the 50-ms bound, which is a reasonable path length. However if $\mathrm{D} = 11$, the maximum number of hops on the slowest path is only 4 before exceeding the delay budget, which is unreasonable in practical networks. Nevertheless, in an access network where low-end routers such as the deployed PC-based prototypes might be used, $\mathrm{D} = 5$ is a realistic value. Reasonably assuming that each service provider deploys an S3 in its network, the S3 is going to be within a limited number of hops (reasonably less than 5) from end-users. Hence, the presented results show how the proposed solution enables the PF technology to reach also the extreme edge of the network with low cost, not specifically designed nodes—such as a PC-based router.

However, these results somehow represent a worst-case scenario since they refer to a low-performance (from both the software and hardware viewpoint) prototypal router. In fact, high-performance commercial routers have maximum packet-handling latency of few μs. For example, the maximum packet-handling latency measured on a Juniper Networks T640 core router [20] during a zero-loss test with small FIFO buffers and no route-lookup delays (no routes present in the router), is about 50 μs. Although this test has been designed to avoid long buffering and lookup delays, the measured latency includes delays related to some functions—such as processing, switching, and buffering—that do not affect the synchronization accuracy in a PF network, but give the largest contribution to packet latency. Consequently, it is reasonable to assume that the latency of functions affecting the synchronization accuracy, i.e., packet transmission and reception, does not exceed 10 μs. This results in a lower synchronization error in each node and consequently in lower delay, jitter, and buffering requirement when compared to our prototype routers. Fig. 9 presents an estimate of the maximum end-to-end buffering delay if high-performance low-latency routers are used. TF duration is set to 50 μs, which is a suitable value for high-performance routers connected with

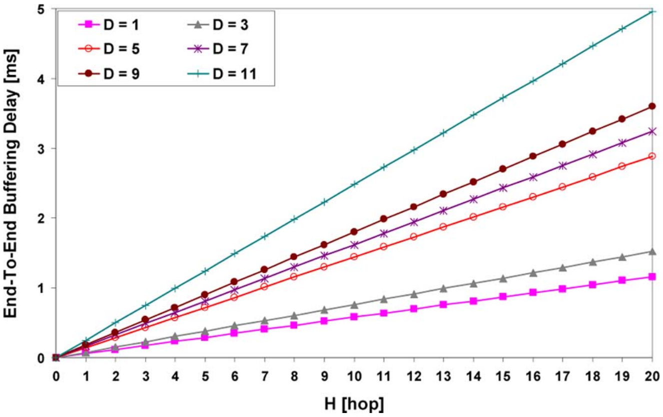

Fig. 9. Estimated maximum end-to-end buffering delay with high-performance commercial routers ($\Phi_i^{\min} = 1\ \mu$s, $\Phi_i^{\max} = 10\ \mu$s, $\mathrm{T} = 50\ \mu$s).

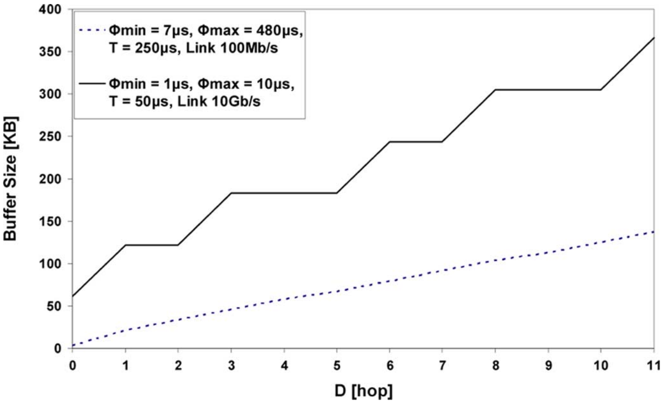

Fig. 10. Buffering requirement.

high-speed links (i.e., 10 Gb/s). Fig. 9 shows that, with $\mathrm{D} = 11$ and 20 hop paths, the maximum end-to-end buffering delay is about 5 ms, thus a small fraction of the 50-ms bound.

Fig. 10 plots the amount of buffering nodes required versus the distance D from the S3, We have considered both our TDP router prototype with 100-Mb/s ports and a high-performance router with 10-Gb/s ports. Results show that the buffer size ensuring loss avoidance is limited for both types of routers, even if nodes are $\mathrm{D} = 11$ hops far from the S3. The buffer size required by the high-performance router (solid line in Fig. 10) is roughly three orders of magnitude smaller than the one adopted in current asynchronous routers operating (with loss) according to the DiffServ model. The limited memory requirement is extremely important for the realization of so-called terabit routers as buffers of very high-capacity ports must have high memory access bandwidth, which, at the current state of the art, implies very high per-byte cost. Buffer size as given by Theorem 2 also implicitly defines the upper bound on the per-node jitter (that is also the end-to-end jitter because with PF, jitter is not additive along the path to the destination), as given by (28). The solid curve in Fig. 10 shows how high— performance commercial routers can ensure a low jitter even with a network-distributed CTR on a large backbone. For example, with $\mathrm{D} = 11$ (i.e., nodes as far as 11 hops from an S3), the jitter is about 300 μs (i.e., 6 TFs). As shown by the dotted line in Fig. 10, also low-cost PC-based PF routers can guarantee acceptable jitter, such as 24 TFs or 6 m,s when $\mathrm{D} = 5$, i.e., when the synchronization signal travels a limited number of hops to reach every node. This further demonstrates the feasibility of implementing and deploying PF with a low-accuracy CTR, possibly distributed through the network with a very simple protocol such as the one proposed in this work. Specifically, the feasibility is demonstrated on both a low-cost, low-performance router platform, such as a PC, on an access network and on a high-end commercial router on a large backbone.

VI. Concluding Remarks

This paper analyzes how pipeline forwarding (PF) of packets can be based on a low-accuracy common time reference (CTR) distributed through the network. A CTR distribution solution aiming at simplicity and ease of implementation, possibly by adding a software-only module to existing devices, is also presented. The analytical work and experiments validate both the proposed synchronization solution and the PF modifications to enable its operation with a low-accuracy network-distributed CTR, thus demonstrating its feasibility and applicability to both large-scale, high-speed networks where minimum buffering requirements are of utmost importance and access networks where low-end routers might be deployed.

The experimental results presented in the paper are obtained with a TDP router prototype. Although they could seem poor at first sight, they are in fact very significant: The simplicity of the deployed algorithms, for both packet scheduling and synchronization distribution, enables their software-only implementation in low-cost architectures, such as PCs, thus making them capable of providing guaranteed quality of service. Such low-cost routers provide acceptable delays only in scenarios where synchronization is distributed through a small number of hops, and paths through the network are not too long, such as access networks. In such a scenario, a low-complexity, software-only implementation of both PF and CTR distribution is key to enable PF deployment in home gateways and wireless access points. In fact, devices available in that market segment are currently not equipped with local clocks that might be dedicated to the implementation of the CTR. Moreover, even if PF became more widely adopted, it might not be cost-effective to include *ad hoc* hardware in low-end equipment. On the other hand, the paper argues that implementation of PF with network-distributed CTR in high-end commercial routers is suitable for global-scale operation.

The service guarantees, and especially the scalability featured by PF, cannot be achieved by other existing technologies. Asynchronous packet scheduling fails in guaranteeing quality of service without underutilizing network resources (i.e., without performing resource overallocation), while synchronous techniques like Sonet/SDH require complex architectures and very accurate synchronization (e.g., a PC-based implementation is unthinkable).

Given that forwarding delay, buffering requirement, and jitter are dependent on the synchronization error—the latter two on its variation, as shown by (27)—minimizing it, and especially its variations, is essential. Work can be done in this respect in at least two complementary directions: 1) reducing latencies in network nodes and 2) limiting the maximum number of nodes through which the CTR is distributed.

Regarding the first issue, work is ongoing to improve the performance of the TDP router prototype. Fig. 5 shows that the variable component of the synchronization error in the presented experiments is less than one TF for most packets. The higher latencies randomly observed stem from both hardware architecture and operating systems designed for general purposes and are not therefore optimized for PF operation. Although, as mentioned earlier, the TDP router prototype is not fully representative of optimized commercial routers, work on its improvement as a PF node can provide experience and insight into general implementation issues. Preliminary results after modifying the FreeBSD kernel in this direction are very promising.

In order to limit the number of nodes through which the CTR is distributed, a network can be divided in *synchronization areas*, each one equipped with an S3 from which network nodes derive the CTR. Such areas can be identified dynamically through the protocol for the construction of the synchronization tree according to various criteria. For example, in a scenario in which OSPF is used for distribution of routing information, it could also be used for the construction of the synchronization tree, and synchronization areas can coincide with OSPF areas. The design of a protocol for the construction of the synchronization tree, also supporting the identification of synchronization areas and reconfiguration in case of failure, is the object of additional study that will also address the impact on existing schedules of changing S3 and S3P.

Acknowledgment

The authors would like to thank R. Giacomelli, whose graduation project helped start up this work.

References

[1] S. Blake *et al.*, "An architecture for differentiated services," IETF Std. RFC 2475, Dec. 1998.

[2] R. Braden, D. Clark, and S. Shenker, "Integrated services in the internet architecture: An overview," IETF Std. RFC 1633, Jul. 1994.

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

[4] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proc. IEEE*, vol. 83, no. 10, pp. 1374–1396, Oct. 1995.

[5] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Trans. Netw.*, vol. 3, no. 4, pp. 365–386, Aug. 1995.

[6] M. Baldi and Y. Ofek, "End-to-end delay analysis of videoconferencing over packet-switched networks," *IEEE/ACM Trans. Netw.*, vol. 8, no. 4, pp. 479–492, Aug. 2000.

[7] C.-S. Li, Y. Ofek, A. Segall, and K. Sohraby, "Pseudo-isochronous cell forwarding," *Comput. Netw. ISDN Syst.*, vol. 30, pp. 2359–2372, 1998.

[8] C.-S. Li, Y. Ofek, and M. Yung, "Time-driven priority flow control for real-time heterogeneous internetworking," in *Proc. IEEE INFOCOM*, San Francisco, CA, Mar. 1996, pp. 189–197.

[9] D. Grieco, A. Pattavina, and Y. Ofek, "Fractional lambda switching for flexible bandwidth provisioning in WDM networks: Principles and performance," *Photonic Netw. Commun.*, vol. 9, no. 3, pp. 281–296, May 2005.

[10] M. Baldi, G. Marchetto, G. Galante, F. Risso, R. Scopigno, and F. Stirano, "Time driven priority router implementation and first experiments," in *Proc. IEEE ICC*, Istanbul, Turkey, Jun. 2006, pp. 897–902.

[11] D. Agrawal, M. Baldi, M. Corrà, G. Fontana, G. Marchetto, V. T. Nguyen, Y. Ofek, D. Severina, H. T. Truong, and O. Zadedyurina, "Ultra scalable UTC-based pipeline forwarding switch for streaming IP traffic," presented at the IEEE INFOCOM—Posters & Demos, Barcelona, Spain, Apr. 2006.

[12] M. Baldi, R. Giacomelli, G. Marchetto, and A. Vesco, "On the deployment of pipeline forwarding in a nation-wide internet service provider network," in *Proc. IC-BNMT*, Beijing, China, Sep. 2007, pp. 110–116.

[13] S. J. Golestani, "A stop-and-go queuing framework for congestion management," *ACM SIGCOMM*, vol. 20, no. 4, pp. 8–18, Sep. 1990.

[14] M. Baldi, G. Marchetto, and Y. Ofek, "A scalable solution for engineering streaming traffic in the future internet," *Comput. Netw.*, vol. 51, no. 14, pp. 4092–4111, Oct. 2007.

[15] D. Mills, "Network time protocol (version 3) specification, implementation and analysis," IETF Std. RFC 1305, Mar. 1992.

[16] *Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Std. 1588, 2004.

[17] J. Gildred *et al.*, "Synchronous ethernet," Pioneer Research Center Specification Draft v0.39, Nov. 2003.

[18] Y. Ofek, "Generating a fault tolerant global clock using high-speed control signals for the MetaNet architecture," *IEEE Trans. Commun.*, vol. 42, no. 5, pp. 2179–2188, May 1994.

[19] O. Gurewitz, I. Cidon, and M. Sidi, "Network classless time protocol based on clock offset optimization," *IEEE/ACM Trans. Netw.*, vol. 14, no. 4, pp. 876–888, Aug. 2006.

[20] "Juniper networks T640 performance test report," BTexact Technologies, Technical Report, 2003.

Mario Baldi (M'99) received the M.S. degree in electrical engineering and the Ph.D. degree in computer and systems engineering from Politecnico di Torino, Torino, Italy, in 1993 and 1998, respectively.

He is Associate Professor at the Department of Control and Computer Engineering of Politecnico di Torino, Torino, Italy, and Vice Dean of the PoliTong Sino-Italian Campus in Shanghai, China. His research interests include high performance switching, optical networking, and computer networks in general.

Guido Marchetto (M'06) received the Ph.D. degree in computer and system engineering from Politecnico di Torino, Torino, Italy, in 2008.

He is a Post-Doctoral Fellow at the Department of Control and Computer Engineering of Politecnico di Torino. His research topics are packet scheduling and quality of service in packet-switched networks. His interests include network protocols and network architectures.