Multi-Terabit/s IP Switching with Guaranteed Service for Streaming Traffic

Mario Baldi
Politecnico di Torino

Yoram Ofek
Università di Trento

I. The Problem

The steady Internet growth over the past few years is impressive, but services so far deployed over the Internet are nothing compared to the ones that can still be deployed. One likely scenario is that the future Internet will be dominated by applications such as (3D) video on demand, high quality videoconferencing, distributed gaming, (3D) virtual reality, remote surveillance, and many more. These applications generate traffic that is either by nature streaming or can be effectively handled as such (e.g., large file transfers). Moreover, most of these applications need a minimum guaranteed quality in order to be usable. Consequently, there is a real need to solve scalability *and* traffic engineering simultaneously — specifically, without using over-provisioning in order to provide predictable service.

Concerning scalability, it is interesting noting that Cisco's top-of-the-line router, CRS-1, has a per chassis switching capacity of 640 Gb/s (the announcement of 92 Tb/s is to be divided by 2, to avoid twice packets first entering and exiting the switch, and then by 72 chassis's), which represents an improvement over the Cisco 12000 by a factor of only 2 after 5 years of development — not the 18 months during which the Internet traffic doubles.

This paper shows how the Internet can benefit from UTC-based pipeline forwarding of IP packets that enables (*i*) ultra-scalable IP switches – 10-50 Tb/s in a single chassis, (*ii*) quality of service (QoS) for UDP-based streaming applications (as a bonus since a deterministic service is inherent to the switching solution itself), while (*iii*) preserving elastic TCP-based best-effort traffic as is. Notice that no change can be seen when observing a link: standard (whole) IP packets encapsulated into Ethernet or PPP frames transit.

II. UTC-Based Pipeline Forwarding

Implementing UTC-based pipeline forwarding for real-time packet scheduling requires IP packet switches to be synchronized with a *common time reference* (CTR). *UTC* (coordinated universal time) offers a CTR that is globally available through various time-distribution systems such as the global positioning system (GPS) and, in the future, Galileo.

Synchronized IP packet switches use a basic time period called time frame (TF) whose duration is derived from the UTC second. Time frames are grouped into time cycles (TCs) and TCs are further organized into super cycles, each of which typically lasts one UTC second. The transmission capacity during each TF is partially or completely reserved to one or more flows. The TC and the super cycle provide the basis for periodic repetitions of the reservation.

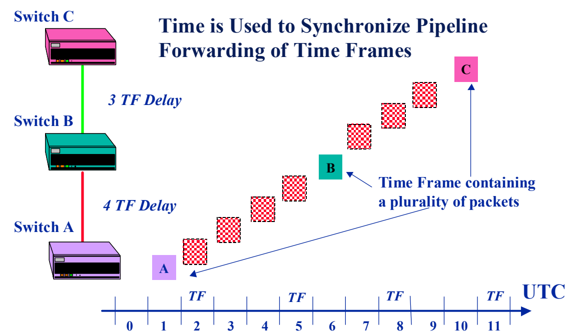

Fig. 1: UTC-based pipeline forwarding

The periodic scheduling within each node results in a *periodic packet forwarding* across the network, which is also referred to as *pipeline forwarding* for the ordered, step-by-step fashion, with which packets travel deterministically (see Fig. 1). The periodic scheduling fits particularly well to periodic (e.g., streaming) traffic, but it can be beneficial in various other contexts, such as large file transfers. UTC-based forwarding guarantees that reserved traffic experiences: (*i*) bounded end-to-end delay, (*ii*) delay jitter lower than one TF, and (*iii*) no congestion and no resulting loss.

III. Optimally Scalabile Design

Pipeline forwarding is a method known to provide optimal performance independent of specific implementation. Introduced by Henry Ford, and still deployed today, in manufacturing processes, pipeline forwarding is part of computers' central processing unit (CPU) operating principles.

Applying pipeline forwarding to IP packets over the Internet enables the construction of a 10-50 Tb/s switch (Fig. 2) in a single chassis with the following optimal properties: (*i*) only input buffers of minimum size with optimal speedup of 1, (*ii*) switching complexity $O[N{*}log_a N]$, (*iii*) switching speedup of 1, (*iv*) minimum switching controller complexity, and (*v*) unaffected support of (TCP-based) elastic traffic through best-effort or differentiated service.

In the switch design shown in Fig. 2, non-pipelined IP packets (i.e., packets that are not part of the reserved traffic, such as IP best-effort packets) can be effectively supported by a hybrid design. In essence, streaming media and large file

This work was supported by funds from the European Commission under the IP-FLOW contract No MC-EXC 002807 and partly under the E-Next Project FP6-506869.

transfer are handled accommodates optimally through pipeline forwarding, while elastic best-effort traffic (which will constitute only a small fraction of the future traffic) through traditional high complexity routing.

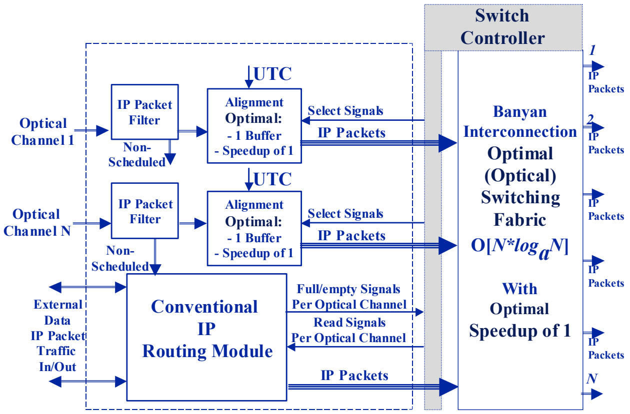

Fig. 2: Hybrid IP routing and UTC-based pipeline forwarding

IV. Experimentation

UTC-based pipeline forwarding was implemented recently in prototypal switch at the University of Trento that is scalable to multi-terabit/s switching capacity. The successful implementation, that required a few (master and PhD) students and researchers and took only 9 months, is a direct outcome of the simplicity (and optimality) of the pipeline forwarding method. The simplicity of this realization did not compromise two most desired performance properties for the future Internet: (1) switching scalability to 10 Tb/s in a single chassis and (2) predictable QoS performance for streaming media and large file transfers.

In particular, two key issues in the scalability o the switch are the switching fabric and its controller. The former is implemented by interconnecting in a Banyan topology (i.e., the lowest complexity, thus most scalable, interconnection network) commercially available Mindspeed M21151 switches, that are 144 x 144 crosspoint switches with a transfer rate or 3 Gb/s (i.e., a 400 Gb/s switching capacity). The switch controller was implemented with limited effort on an FPGA. It receives a pulse per second signal from a GPS receiver to realize the common time reference and controls multiple switches.

The prototypal pipeline forwarding switch was deployed in the testbed shown in Fig. 3 that includes also a prototypal pipeline forwarding router developed at the Politecnico di Torino [4]. Two streaming video flows are generated by a video server (to the left), transported, with *deterministic quality of service*, through a network of one router and two multi-terabit/s switches (all implementing pipeline forwarding) and delivered to two different video clients. The pipeline forwarding router is responsible for time-shaping the packet flows generated asynchronously by the vireo streaming sources, i.e., to forward packets towards the first multi-terabit/s switch during the proper TFs. IP packets carrying video samples are transported unchanged as a whole end-to-end. Namely, no change can be seen by observing packets flowing on any link of the testbed as only conventional IP packets encapsulated into Ethernet frames travel across the network testbed.

Some Recent Related Publications

[1] V. T. Nguyen, R. Lo Cigno, Y. Ofek, "Design and Analysis of Tunable Laser-based Fractional Lambda Switching (FLS)," IEEE INFOCOM 2006.

[2] M. Baldi and Y. Ofek, "Fractional Lambda Switching - Principles of Operation and Performance Issues", *SIMULATION: Transactions of The Society for Modeling and Simulation International*, Vol. 80, No. 10, Oct. 2004, pp. 527-544.

[3] D. Grieco, A. Pattavina and Y. Ofek, "Fractional Lambda Switching for Flexible Bandwidth Provisioning in WDM Networks: Principles and Performance", *Photonic Network Communications*, Issue: Volume 9, Number 3, Date: May 2005, Pages: 281 – 296.

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

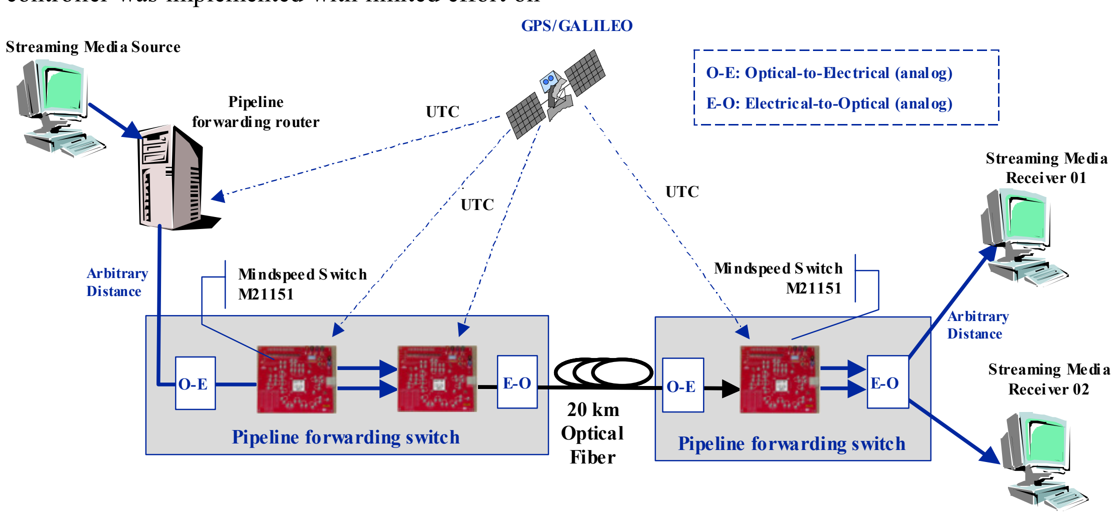

Fig. 3: Multi-Tb/s switch testbed prototype setup