A Scalable Approach for Supporting Streaming Media: Design, Implementation and Experiments

D. Agrawal[◆], M. Baldi[●], M. Corrà[■], G. Fontana[◆], G. Marchetto[●], V. T. Nguyen[◆], Y. Ofek[◆], D. Severina[◆], T. H. Truong[◆], O. Zadedyurina[◆]

[◆]Department of Information and Communication Technology University of Trento, Via Sommarive 14, I-38050 Povo, Trento, Italy

[■] TRETEC S.r.l., Via Solteri 38, 38100 Trento, Italy

[●]Control and Computer Engineering Department, Politecnico di Torino Corso Duca degli Abruzzi, 24, 10129 Torino, Italy

Abstract

Future Internet traffic will be dominated by on-demand streaming media flows, such as IPTV, 3D/HD video, gaming, virtual reality, and many more. Consequently, future network architectures will need to offer predictable performances to such applications. In order to achieve scalable IP packet switching it is essential to minimize "stopping" of the serial bit streams, in order to minimize: buffer size, jitter and loss. Our recent experimental work demonstrated how an IP network can be implemented without "stopping" the serial bit streams. Consequently, the switch realized is very simple, scalable to 10-100 terabits per second in a single chassis, and suitable for all optical implementation. The implemented testbed uses only off-the-shelf optical and electronic components and was completed in 9-month.

1. Introduction

The Internet has been growing steadily in the past few years. One may envision that demand growth for various services to home users will increase the need for transmission and switching capacity by 100-fold in 10-year. One likely scenario is that the future Internet will be dominated by applications such as (3D/HD) video on-demand, high quality videoconferencing, distributed gaming, (3D/HD) virtual reality, and many more. These applications generate traffic that is either by nature streaming or can be effectively mapped into streaming and handled as such (e.g., large file transfer).

This paper presents a scalable testbed demonstrating a method known as *pipeline forwarding* ([1]-[3]) that is particularly suitable to carry streaming media applications over the Internet since it offers:

1) High scalability of network switches (10-100 Terabit/s in a single chassis);
2) Service guarantees (deterministic delay and jitter, no loss) for (UDP-based) constant bit rate (CBR) and variable bit rate (VBR) streaming applications — as needed, while
3) Preserving the support of elastic, TCP-based traffic, i.e., existing applications based on "best-effort" services are not affected in any way.

In the presented testbed, shown in Fig. 1, two streaming video flows are generated by a video server (to the left), transported, with deterministic delay without loss, through a network of one router and two multi-terabit/s switches (all implementing pipeline forwarding) and delivered to two different video clients. 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. More recently, we demonstrated the same over a testbed with six multi-terabit/s switches and 100 km of fiber. Notice that notwithstanding its limited size, the testbed is fully representative of a real-life network scenario likely to be implemented in the initial deployment phases of pipeline forwarding, as discussed in Section 3 and depicted in Fig. 3

In essence what the testbed demonstrates is a scalable technology and network architecture for supporting and engineering (UDP-based) streaming traffic, while elastic (TCP-based) traffic is kept unchanged and unaffected. Proper and efficient

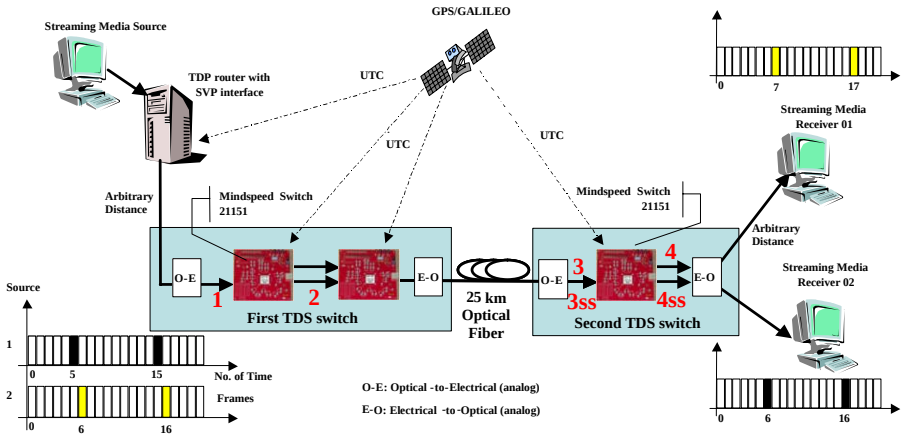

Fig. 1. Testbed functional diagram – two TDS switches no "stopping" of the serial bit stream

support of streaming UDP-based applications is getting increasingly important due to the fact that more and more streaming media traffic over the Internet is using UDP. Such applications need a minimum level of service quality in order to operate properly and current approaches to offer controlled quality based on the Differentiated Services (DiffServ) model combined with over-provisioning do not scale as they assume that differentiated traffic is much smaller compared to the entire network capacity. On the other hand, approaches based on the Integrated Services (IntServ) model 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) [4], a.k.a. weighted fair queuing (WFQ). Moreover, PGPS and other similar well known scheduling algorithms [5][6], such as, class based queuing, weighted round robin and others, cannot combine optimal delay and resource utilization efficiently (see detailed discussion in [7]). In summary, existing asynchronous packet switching approaches require (very) large buffer (i.e., a lot of "stopping" of the serial bit stream) without the necessary performance guarantees. Conversely, the presented testbed enables the realization of (very) low complexity, highly scalable IP switches for optimal support of streaming real-time traffic.

Section 2 focuses on pipeline forwarding, which the technology is underlying the testbed, by presenting pipeline forwarding operating principles and how it can be used to support streaming media without "stopping" the serial bit stream. Section 3 discusses the multimedia system that can utilize this testbed architecture. Then the testbed is described in Section 4, which provides design, implementation, and some measurements, while concluding discussion is presented in Section 5.

2. Underlying Principles and Technologies

2.1. Pipeline Forwarding

Pipeline forwarding is a known optimal method that is widely used in computing and manufacturing. The necessary requirement for pipeline forwarding is having common time reference (CTR). In this design UTC (coordinated universal time) is used for CTR, consequently, the method used in the testbed is called UTC-based pipeline forwarding ([1]-[3]).

In UTC-based pipeline forwarding all packet switches are synchronized, while utilizing a basic time period called time frame (TF). The TF duration (T_f) may be derived, for example, as a fraction of the UTC second received from a time-distribution system such as the global positioning system (GPS) and, in the near future, Galileo. TFs are grouped into time cycles (TCs) and TCs are further grouped into super cycles, each super cycle lasts for one UTC second. TFs are partially or totally reserved for each flow during a resource reservation phase. The TC provides the periodicity of the reserved flow. This result in a periodic schedule for IP packets to be switched and forwarded, which is repeated every TC.

For example, in Fig. 2, the 125μs time frame T_f is obtained by dividing the UTC second by 8000; sequences of 100 TFs are grouped into one TC, and runs of 80 TCs are comprised in one super cycle (i.e., one UTC second). Note that the TF duration is a link parameter that is proportional to the link capacity, for example, reasonable time frame duration for 10GE is 12.5μs (instead of the above 125-μs, which is suitable

for GE).

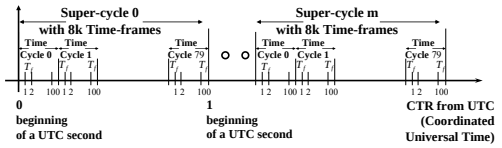

Fig. 2. Common time reference structure

The basic pipeline forwarding operation is regulated by two simple rules: (i) all packets that must be sent in TF t by a node must be in its output ports' buffers at the end of TF $t-1$, and (ii) a packet p transmitted in TF t by a node n must be transmitted in TF $t+dp$ by node $n+1$, where dp is an integer constant called forwarding delay, and TF t and TF $t+dp$ are also referred to as the forwarding TF of packet p at node n and node $n+1$, respectively. The value of the forwarding delay is determined at resource-reservation time and must be large enough to satisfy (i). In pipeline forwarding, a synchronous virtual pipe (SVP) is a predefined schedule for forwarding a pre-allocated amount of bytes during one or more TFs along a path of subsequent UTC-based switches. A hierarchical resource reservation model can be used to set-up SVPs, which enables multiple component SVPs to be aggregated in larger, possibly pre-provisioned, SVPs in the core of the network.

The UTC accuracy required to implement pipeline forwarding can be relaxed by using two means in the design of routers and protocols: (i) time frame delimiters and (ii) time stamps. The reason for such a relaxed requirement is that the UTC is not used for detecting the time frame boundaries, as they are detected by means of the delimiters. For example, time frame delimiters enable correct mapping of incoming packets to the input channel time frames, i.e., the UTC time frames in which they were transmitted by the upstream node. When a time frame delimiter exists, the correct mapping of TFs can be maintained with a UTC accuracy of $\frac{1}{2}\cdot T_f$ — i.e., UTC±1/2·(12.5μs to 125μs) — without compromising on performance and complexity. Additional mechanisms and buffering can be used to enable an even sloppier synchronization to UTC. Deployment of time stamps (e.g., including in packets the TF in which they were transmitted) enables higher fault and loss tolerant solutions. However, tradeoffs between UTC accuracy, system complexity, delay, and robustness are outside the scope of this paper and will be the subject of further work.

Today, time cards with 1 PPS (pulse per second) UTC with accuracy of 10-20 ns are available from various vendors. These cards are small and cost around $100 each. By combining such time cards with Rubidium or Cesium clocks it is possible to have a UTC reference within the required accuracy for days (with Rubidium) and months (with Cesium) in the event the GPS signal is lost.

UTC-based forwarding guarantees that reserved real-time traffic experiences: (i) bounded end-to-end delay, (ii) delay jitter lower than two TFs, and (iii) no congestion and resulting losses. These properties are ensured also when multicasting is implemented, as described in [2]. Moreover, it is worth highlighting that TFs are virtual containers; consequently, packets can extend beyond the time boundaries of a TF. In order for pipeline forwarding to operate properly and ensure the above properties, network nodes must deploy means to uniquely identify the TF a packet logically belongs to, such as either TF delimiters or time stamps or measuring the transmission time of the first bit of a packet.

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

2.2. Time-Driven Switching

Time-driven switching (TDS) was proposed to realize sub-lambda or fractional lambda switching (FλS) in highly scalable dynamic optical networking [1], which requires minimum optical buffers. In this context, TDS has the same general objectives as optical burst switching and optical packet switching: realizing all-optical networks with high wavelength utilization. TFs can be viewed as virtual containers for multiple IP packets that are switched at every TDS switch based on and coordinated by the UTC signal.

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

Scheduling through a switching fabric is based on a pre-defined schedule, which enables the implementation of a simple controller. Moreover, low-complexity switching fabric architectures, such as Banyan, can be deployed notwithstanding their blocking features, thus further enhancing scalability. In

fact, blocking can be avoided during schedule computation by avoiding conflicting input/output connections during the same TF. Previous results [1] show that (especially if multiple wavelength division multiplexing channels are deployed on optical links between fractional λ switches) high link utilization can be achieved with negligible blocking using a Banyan network without speedup.

2.3. Time-Driven Priority

TDS is suitable for very high speed (optical) backbones, where traffic can be organized in large capacity SVPs handled by high performance switches. More flexibility could however be required at the edge of the network as offered, for example, by conventional IP destination-address-based routing. Time-driven priority (TDP) [3][7] is a synchronous packet scheduling technique that implements UTC-based pipeline forwarding and can be combined with conventional IP routing to achieve the abovementioned flexibility. Packets entering a switch from the same input port during the same TF can be sent out from different output ports, according to the rules that drive IP packet routing. So, routing and forwarding may be based on either the conventional IP destination-address-based routing (as mentioned above), or multi-protocol label switching (MPLS), or any other technology of choice.

2.4. Non-pipelined Traffic

Non-pipelined (i.e., non-scheduled) IP packets, i.e., packets that are not part of a SVP (e.g., IP best-effort packets), can be transmitted during any unused portion of a TF, whether it is not reserved or it is reserved but currently unused. Consequently, links can be fully utilized even if flows with reserved resources generate fewer packets than expected. A large part of Internet traffic today is generated by TCP-based elastic applications (e.g., file transfer, e-mail, WWW) that do not require a guaranteed service in term of end-to-end delay and jitter. Such traffic can be dealt with as non-pipelined and can benefit from statistical multiplexing.

When the network is highly loaded and almost fully booked, unused capacity between almost fully reserved TFs might be smaller than the size of queued non-pipelined packets. In such situation, non-pipelined queues cannot be emptied and a fraction of link capacity is wasted. This can be avoided by applying non-disruptive preemptive priority. The data link layer at the transmitting end of a link splits non-pipelined packets over multiple frames fitting in the unused bandwidth at the end of subsequent TFs. The receiving end of the link reassembles the received fragments. For example, an extension of Multilink PPP in the time domain could be deployed to this purpose.

3. Multimedia System Architecture

Fully benefiting from UTC-based pipeline forwarding requires providing network nodes and end-systems with a CTR and implement network applications in a way that they use it to maximize the quality of the received service. However, the Internet is currently based on asynchronous IP packet switches and IP hosts. Thus, especially in an initial deployment phase, UTC-based pipeline forwarding must coexist and interoperate with current asynchronous packet switches, hosts and applications (e.g., IP video-phones, video-streaming servers and clients, etc.). This is achieved with a network architecture such as the one depicted in Fig. 3, which is also realized in the presented testbed shown in Fig. 1. In such network scenario, senders generate asynchronous multimedia traffic that passes through an asynchronous access network towards a TDP domain (in the presented testbed, depicted in Fig. 1, the TDP network encompasses a single router). Edge TDP routers are connected to traditional asynchronous IP routers through a SVP (synchronous virtual pipe) interface that controls access to the pipeline forwarding network by policing and shaping the incoming traffic flows, i.e., asynchronous packets are stored in a buffer waiting for their previously evaluated forwarding TF.

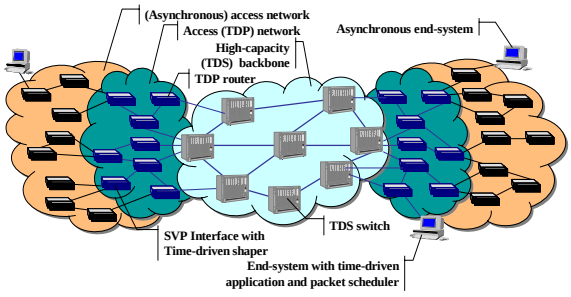

Fig. 3. Architecture for deploying pipeline forwarding

In order to further elaborate on the contribution of the SVP interface to the end-to-end service, the transmission of CBR video is considered as an example. A source generates traffic periodically, resources are reserved with the same period, and packet bursts are transmitted right before the TF in which a reservation was made. If some level of

synchronization exists between the video source and the network, end-to-end delay can be minimized, as analyzed in detail in [7]. If the video source is not synchronized with the network, additional delay is introduced by the interface. However, if the reservation has been properly performed, the video flow is carried through the pipeline forwarding network with deterministic quality, i.e., without congestion and with bounded delay.

As shown in Fig. 3, TDS switches are deployed in the network core as they feature particularly low complexity, hence high performance at low cost. Instead, TDP routers are used at the backbone edge where scalability is not a primary concern, but higher flexibility (stemming from conventional IP routing and full hop-by-hop traffic multiplexing) might particularly beneficial. At TDS switches there is no packet header processing and routing is based on the TF during which packets are sent. Consequently, TDP routers at the backbone edge are instrumental in properly "time-shaping" asynchronous IP packets by transmitting them in pre scheduled TFs. In a way, a TDP router connecting to the TDS core acts as a TDS interface. At resource reservation time, schedules must be defined seamlessly across the TDP backbone edge and the TDS backbone core. Moreover, schedules must be such that packets transmitted to a TDS ingress node during one TF have to reach the same TDS backbone egress node.

On the other hand, notice that the output of a TDS switch or a TDP router can be sent directly to an asynchronous IP router or end system, as in the testbed presented in this paper (see video clients on the right-hand side of Fig. 1). UTC-based pipeline forwarding could also be beneficial when confined to subnetworks: edge routers at the ingress of each TDP subnetwork would eliminate the jitter experienced by packets in the asynchronous network then packets would benefit of the controlled delay service provided by the pipeline forwarding subnetwork.

4. Testbed

4.1. Testbed architecture and components

The network architecture for streaming media applications presented in the previous sections has been demonstrated by building the testbed for video distribution shown in Fig. 1. The main components of the testbed are:

1) Asynchronous streaming sources (audio, video and text) implemented by a video server;
2) TDP router that represents a TDP domain at the backbone edge; the TDP router acts both as edge router encompassing a SVP interface and as an interface to the TDS backbone;
3) 25km single-mode optical fiber;
4) Two TDS switches that are constructed with Mindspeed M21151/6 switch boards, with nominal switching capacity of each M21151/6 is 400 Gb/s;
5) FPGA-based controller for scheduling the operation of two M21151/6 (implementing a two stage Banyan interconnection network, which is scalable to 10 Tb/s switching capacity);
6) Streaming media receivers for separately playing the two video streams transmitted from the asynchronous streaming sources.

Time parameters (i.e., TF duration of 100 μs and TC size of 100 TFs) are configured at the TDP router and the two the TDS switches. The switches are configured by the switch controller by means of a simple graphical user interface (GUI). Multi mode fiber is used for the optical links between TDP router and first TDS switch and between second TDS switch and end-systems. The two TDS switches are connected by 25 km single mode fiber. (We comment that recently we managed to successfully "loop" 3-time through the network in Fig. 1, which effectively constitutes 6-node TDS network with total 100 km of single mode fiber; and without "stopping" the serial bit stream! This implies that our approach can be deployed without buffering in almost all metropolitan network scenarios.)

Two asynchronous video flows are generated by the streaming media source, as is shown in Fig. 1, and transmitted to the SVP interface within the TDP router. The streaming video packets are then forwarded via an optical link by the TDP router through Gigabit Ethernet (GE) transceivers to the first TDS switch during different predefined TFs. Specifically, TF 5 and TF 15 belong to one SVP while TF 6 and TF 16 belong to another SVP. The optical signal entering the first TDS switch is converted to an electrical one, switched by the first stage and forwarded to the second switching stage through an electrical connection. Then the video streams are transmitted as an optical signal, through a single mode optical fiber link of 25 km, to the second TDS switch that routes each video stream to a different output. Then the separated video streams are forwarded to two receivers through optical links of an arbitrary length. Video flows are therefore multiplexed on the first and second link they traversed, but TDS ensures that video packets reach their corresponding destination with deterministic delay

(with no jitter), i.e., during TF 6 and TF 16 and during TF 7 and TF 17 respectively, as showed in Fig. 1. Switching of all three switching boards and network interfaces are synchronized with the 1PPS (pulse per second) signal received from three different GPS receivers.

4.2. TDP router: the SVP interface

Generically, in a router data plane packets are moved from input ports to output ports going through three modules that perform *input processing*, *forwarding*, and *output processing*. Thus, the same was realized in FreeBSD-based TDP router. The operations to be performed by a TDP router in each module are discussed in the following.

Edge routers need to shape asynchronous traffic entering the TDP/TDS network. Consequently, their input module corresponding to SVP interfaces (i.e., interfaces not connected to a pipeline forwarding node) comprise mechanisms to classify incoming packets, identify the data flow they belong to, and select a TF in which they will be forwarded according to the current resource reservation setup. The evaluated *forwarding TF* determines the output buffer where packets will be stored by the output module. *Dummynet* [8], a FreeBSD filtering extension module using programmable rules was used in the SVP interface implementation to classify incoming asynchronous data flows and store packets up to one TF before output modules fetch them for transmission.

The forwarding module processes packets according to the specific network technology; specifically, in the presented prototype packet routing and forwarding is based on IP. No modification is required to the forwarding module of a router when adding TDP as it is simply a packet-scheduling technique.

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

The *alternate queuing* (*ALTQ*) [9] framework developed for FreeBSD, which for historical reasons (the TCP/IP stack was first developed on the BSD platform) is considered a reference platform for development and testing of new networking functionalities, includes many packet-scheduling policies such as First-In First-Out (FIFO), Weighted Fair Queuing (WFQ), Class Based Queuing (CBQ), hierarchical fair service curve (HFSC). The TDP output module is simply implemented as just another scheduling discipline within ALTQ.

The UTC is provided to the router using a Symmetricom bc637PCI-U GPS receiver PCI card [10] that can generate interrupts at a programmable rate ranging between less than 1 Hz to 250 kHz (1PPS (pulse per second) and 4 μs). Two global variables added to the FreeBSD kernel to hold the current TF and TC number are updated whenever these interrupts occur. The current version of the prototype does not include signaling functions, i.e., that TFs and TCs are statically allocated by manual configuration. Nevertheless, the prototype does not loose the generality and utmost attention was paid to accurately avoid any design limitations or assumptions that could prevent signaling functions from being implemented in the testbed in the future.

4.3. TDS switch implementation

Fig. 4 and Fig. 5 show a photograph and a block diagram of the implemented TDS switch, respectively, which has three major parts:

- GPS receiver (for UTC time signal): an EPSILON Board OEM II [10] provides accurate and stable time and frequency signals for synchronization.
- Switch controller: an Opal Kelly XEM3001 module [12] is used for implementing the FPGA (field programmable gate array) based switch controller. The XEM3001 consists of an EEPROM, a USB microcontroller, a PLL, and a 400,000-gate Xilinx Spartan-3 FPGA (XC3S400-4PQ208C) sub module.
- Switching fabric: as graphically depicted in Fig. 5, the switching fabric is implemented as a network of interconnected Mindspeed M21151/6 switching boards [13]. Each board is a low-power CMOS, high-speed 144 x 144 crosspoint switch (400 Gb/s switching capacity) with integrated Clock Data Recovery (CDRs), input equalization, and built-in system test and broadcasting features. Each CDR is preceded by a programmable input equalizer.

The communication between the controller and the M21151/6 boards has three important signals: classification address, data path, and strobe signal as shown in Fig. 5. The FPGA-based controller is connected to a PC via a USB 2.0 link. Control data,

configured from GUI or retrieved from disk, are downloaded by the PC and stored (or updated) in a memory implemented in the FPGA prior to the actual operation of the switch. Multiple finite state machines (FSMs) are implemented in the FPGA. Each FSM coordinates one Mindspeed board. All FSMs are coordinated using UTC time received by the controller as 1PPS and 10 MHz signals.

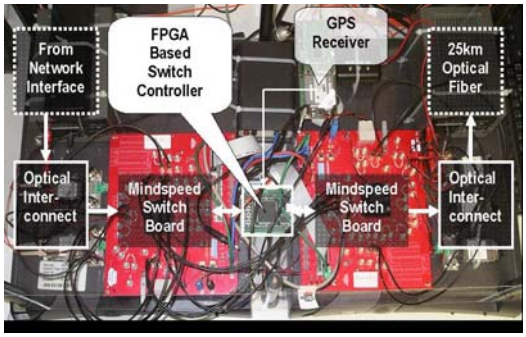

Fig. 4. TDS switch photograph

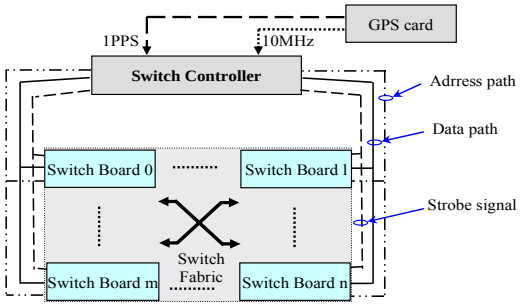

Fig. 5. TDS switch block diagram

Implementing the switch prototype and debugging the entire testbed required specific component level and system level performance measurements and validation. Data integrity has been validated at the output of each switch (i.e., location 1 and location 4 highlighted in Fig. 1) by matching the eye pattern with a standard Gigabit Ethernet 1000 base SX/LX test mask [14]. The result is shown in Fig. 6 and Fig. 7 for location 1 and location 4, respectively, from a snapshot of the screen of an oscilloscope. The signal received at the measurement point is sampled by the oscilloscope and its value plotted on the screen, one yellow dot for each sample. The rectangles and hexagon drawn on the screen reproduce the transmitted eye mask defined in Section 38.6.5 of [14]; the signal is conformant, hence can be correctly received by a compliant receiver, as long as its plotted measures do not touch these geometrical shapes. Note worthily, the measurement at point 4 shown in Fig. 7 was performed after 25 km of single mode fiber. In addition each data link, including switch boards and optical transceivers (GBIC), has been tested for bit error rate (BER) from 0.1 to 3.0 Gb/s, in order to allow for high safety margins.

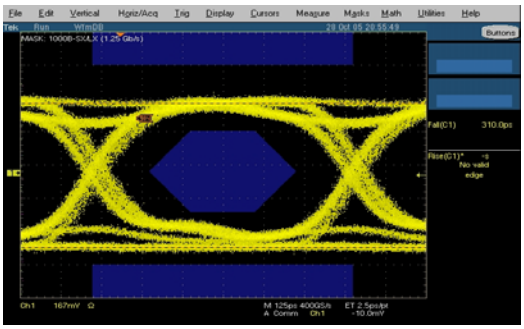

Fig. 6. Eye Pattern at location 1 in Fig. 1

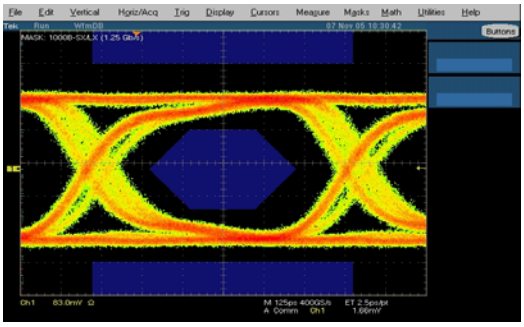

Fig. 7. Eye Pattern at location 4 in Fig.1

Since the current switch implementation deploys neither TFs delimiters nor time stamps (see discussion on UTC accuracy in Section 2.1), safety time windows are inserted to ensure that no data is transmitted if the associated TF is not "well defined". Safety time windows are sized according to the known and measurable error affecting time information from GPS/Galileo.

4.4. System integration

Our testbed consists of a software TDP router and a high speed TDS switch that are interconnected. Once the software and hardware components were interconnected, serious investigation of synchronization, specifically bit synchronization, between various components have been performed. Because a small drift in alignment between the switches and the TDP router can cause packets misrouting loss, inducing bad performance. The harder task would be required in case of placing the switch and TDP router far apart from each other, then it is necessary to test the precise inter-synchronization via GPS and link delay alignment.

Since a PC running FreeBSD is not a real-time system, this is probably not an optimal solution for the presented testbed where the TDP router forwarding to the TDS switch is required to react precisely within a few μs. As a matter of fact, some inherent uncertainties in a non-real-time operating system running on a non-real time architecture lead to variable unpredicted delay (*in handling the GPS interrupts, GE interrupts, latency going through the PCI bus*) that occasionally can be much larger than expected. Therefore, it is potentially much challenging for combining the TDP router operation with the TDS switches that are switching rapidly on a very short time scale.

One of the solutions for the future testbed can be hard-coding TDP scheduling in a specialized hardware box with built-in GE/10GE controller to enable a high-loaded Gb/s-speed pipeline forwarding network testbed.

Nevertheless, in summary, within our current scope — i.e., with proper TF durations, at the expense of industrial robustness, and with limited traffic — the FreeBSD-based router is a low-cost, sufficient device that is able to properly operate with the TDS switches in some scenarios that are meaningful for a testbed of this novel communication architecture.

5. Discussion

Implementing UTC-based pipeline forwarding in a switch that is scalable to multi-terabit/s switching capacity and experimenting with it on a testbed fully reproducing a real-life network scenario has been a rewarding experience. The implementation success is a direct outcome of the simplicity of the pipeline forwarding method. The beauty is that 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 performance for streaming media and large (content) file transfers.

Furthermore, although the presented prototype is based on optically interconnected electronic switches, the serial bit streams are never "stopped"; i.e., the transmitted bits are neither digitized nor digitally stored. In recent experiments it was possible to transmit the serial bit streams through six nodes and 100 km of fiber without “stopping” them. This is significant as it demonstrates that the deployed technology is suitable for providing ultra-scalable switching capacity in metropolitan area networks with minimum cost.

Note that Cisco’s top-of-the-line router, CRS-1, has 640 Gb/s per chassis [the announcement of 92 Tb/s should be divided by 2 (for counting input and output separately) and then by 72 chassis’s], which represent a factor of 2 improvement after 5 years of development. So if the Internet traffic is doubling every, say, 18 months there is a real switching bottleneck on the horizon.

6. References

[1] D. Grieco, A. Pattavina, and Y. Ofek, “Fractional lambda switching for flexible bandwidth provisioning in WDM networks: principles and performance,” Photonic Network Communications, vol. 9, no. 3, pp. 281–296, 2005.

[2] M. Baldi, Y. Ofek, and B. Yener "Adaptive Group Multicast with Time-Driven Priority," *IEEE/ACM Transactions on Networking*, vol. 8, no. 1, pp. 31–43, 2000.

[3] Baldi, G. Marchetto, G. Galante, F. Risso, R. Scopigno, and F. Stirano, “Time Driven Priority Router Implementation and First Experiments,” *Proc. of the IEEE ICC*, 2006.

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

[5] H. Zhang, “Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks,” *Proc. of the IEEE*, vol. 83, no. 10, 1995.

[6] S. Floyd and V. Jacobson, “Link-sharing and Resource Management Models for Packet Networks,” *IEEE/ACM Transactions on Networking*, vol. 3, no. 4, 1995.

[7] M. Baldi and Y. Ofek, “End-to-end Delay of Videoconferencing over Packet Switched Networks,” *IEEE/ACM Transactions on Networking*, vol. 8, no. 4, pp. 479–492, 2000.

[8] “Dummynet,” [Online]. Available: http://info.iet.unipi.it/_luigi/ip dummynet.

[9] K. Cho, “A framework for alternate queuing: Towards traffic management by PC-UNIX based routers,” *Proc. USENIX (Annual Technical Conference’98)*, 1998.

[10] Symmetricom, “bc637PCI-U,” [Online]. Available: http://www.symmttm.com/products_blt_bc637PCI-U.asp

[11] Temex Telecom, [Online] Available: http://www.temex-telecom.com

[12] Opal Kelly,Inc., “XEM3001,” [Online] Available: http://www.opalkelly.com/products/xem3001/

[13] Mindspeed, “M21151,” [Online] Available: http://www.mindspeed.com/web/products/index.jsp?catalog_id=590&cookietrail=0

[14] IEEE 802.3 Working Group, “Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications”, IEEE Std 802.3 2000 Edition, The Institute of Electrical and Electronics Engineers, 2000, ISBN 0-7381-2673-X.