The Token Based Switch: Per-Packet Access Authorisation to Optical Shortcuts

Mihai-Lucian Cristea[1], Leon Gommans[1], Li Xu[1], and Herbert Bos[2]

[1] University of Amsterdam, The Netherlands
{cristea,lgommans,lixu}@science.uva.nl
[2] Vrije Universiteit Amsterdam, The Netherlands
herbertb@cs.vu.nl

Abstract. Our Token Based Switch (TBS) implementation shows that a packet-based admission control system can be used to dynamically select a fast end-to-end connection over a hybrid network at gigabit speeds. TBS helps high-performance computing and grid applications that require high bandwidth links between grid nodes to bypass the regular Internet for authorised packets by establishing shortcuts on network links with policy constraints. TBS is fast and safe and uses the latest network processor generation (Intel IXP2850) and the Fairly Fast Packet Filter software framework.

1 Introduction

Grid and other high-performance applications tend to require high bandwidth end-to-end connections between grid nodes. Often the requirements are for several gigabits per second. When spanning multiple domains, fibre optic networks owners must cooperate in a coordinated manner in order to provide high-speed end-to-end optical connections. Currently, the administrators of such connections use paper-based long-term contracts. There exists a demand for a mechanism that dynamically creates these fast end-to-end connections (termed *lightpaths*) on behalf of grid applications. The use of lightpaths is also envisaged for applications that are connected through hybrid networks [1]. A hybrid network contains routers and switches that accept and forward traffic at layers 1, 2, or 3. In other words, hybrid networks consist of traditional (layer 3) routed networks which allow for optical (layer 1 or 2) shortcuts for certain parts of the end-to-end path. Currently, the peering points between routed networks of the Internet Service Providers (ISPs) by way of the Border Gateway Protocol (BGP) policies determine statically what traffic bypasses the (slow) routed transit network and which links they will use to do so. However, when considering hybrid networks interconnected over long distances, we would like the peering points to play a more active/dynamic role in determining which traffic should travel over which links, especially since multiple links often exist in parallel. Therefore, an important role for the peering points is path selection and admission control for the links.

Figure 1 shows a hybrid network composed of three ISPs interconnected through routed networks (Internet) and also through two optical links managed by different owners. The connections across the optical links is via the peering points (PPs). An example of such a connection may be a high-bandwidth transatlantic link.

I.F. Akyildiz et al. (Eds.): NETWORKING 2007, LNCS 4479, pp. 945–957, 2007.

Users X and Y on the left access servers on the right. We want them to bypass the routed Internet and use optical shortcuts instead. However, while not precluded by the model, we do not require each user to have an individual relation with each ISP and shortcut along the path. Indeed, the link owners should not normally have a notion of which specific IP addresses are allowed to access the link. Instead, we expect ISP_A (say, the organisation, department or research group) to have a contract with the link owners and decide locally (and possibly at short timescales) who should get access to the links in domains that are not under its control. For scalability, the model allows the system to be federated along more levels, where ISP_A contacts the authorisation at the next level, which in turn contacts a server at a higher level still, etc. In practice, the number of levels is expected to be small (often one).

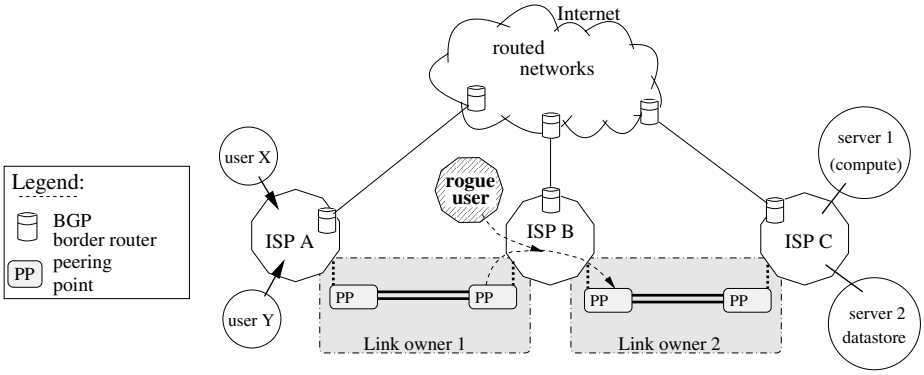


Fig. 1. Peering in hybrid networks: end-to-end lightpaths

A flexible way of bypassing the routed Internet is to have ISP_A tag traffic from X and Y with some sort of token to signal to remote domains that they should be pushed across the optical shortcuts. At the same time, we want to prevent rogue users (e.g., the ISP_B client indicated in the figure) to tag their packets with similar tokens to gain unauthorised access to the shortcuts.

In principle, the signalling of peering requests to the peering points PPs can be done either *out-of-band*, or *in-band*, or both as described in the *Authorisation Authentication Accounting* (AAA) framework (RFC 2904) [2]. In *out-of-band* signalling there is an explicit control channel to PPs, separate from the data channel(s). In an in-band mechanism, the admission control is based on specific data inserted into the communication channel. We opt for *in-band* signalling for reasons of flexibility resulting from the per-packet granularity. Specifically, we insert tokens into each packet as proof of authorisation. Tokens are a simple way to authorise resource usage which may convey different semantics. For instance, we may specify that only packets with the appropriate token are allowed to use a pre-established network connection in a specific time-frame and embed these tokens in the packets of an application distributed over many IP addresses.

However, our tokens differ from common flow identifiers (e.g., ATM VPI/VCI pairs, MPLS labels, and IPv6 flow labels) in that they cannot be tampered with and that they

may be associated with arbitrary IP addresses. In essence, tokens are like IPsec authentication headers (AH [3]) except that we aim for authentication that is more efficient in computation and more flexible than IPsec standard (we can authenticate various fields from Ethernet or IP headers by using a customised mask). In addition, the application domain is different in sense that our TBS system serves applications distributed over many IP addresses.

Tokens bind packet attributes to an issuing attribute authority (e.g., an AAA server in our case). Tokens could be acquired and subsequently used anonymously. Moreover, a token can bind to differrent semantics (e.g., a user, a group of users, or an institute) and decouples time of authorisation from time of use. In the switch described in this paper, tokens are used to select shortcuts and different tokens may cause packets to be switched on different links, but in principle they could also be used to enforce other QoS-like properties, such as loss priorities.

This paper describes both the design of the Token Based Switch (TBS) and its implementation on high-speed network processors. TBS introduces a novel and rather extreme approach to packet switching for handling high-speed link admission control to optical shortcuts. The main goal of this project was to demonstrate that the TBS is feasible at multi-gigabit link rates. In addition it has the following goals: ① path selection with in-band admission control (specific tokens gives access to shortcut links), ② external control for negotiating access conditions (e.g., to determine which tokens give access to which links), and ③ secured access control.

The third goal is also important because part of the end-to-end connection may consist of networks with policy constraints such as those meant for the research application domain in the *LambdaGrid*. Moreover, a critical infrastructure needs protection from malicious use of identifiers (e.g., labels in (G)MPLS, header fields in IPv4, or flowID in IPv6). For these reasons, our token recognition uses cryptographic functions, for example, to implement the Hash function based Message Authentication Code (HMAC) (see RFC2104 [11]). An external control interface is required to negotiate the conditions that give access to a specific link. Once an agreement has been reached, the control interface should accept an authorisation key and its service parameters that will allow packets to access the owner's link. To operate at link speeds we push all complex processing to the hardware. In our case we use a dual Intel IXP2850 with on-board crypto and hashing units.

Several projects address the issue of setting up lightpaths dynamically (e.g., UCLP, DWDM-RAM [4]), and others look at authorisation (e.g., IP Easy-pass [5]). However, to our knowledge, no solutions exist that support both admission control and path selection for high speed network links in an in-band fashion for setting up safe, per-packet-authenticated, optical short-cuts. In addition, our approach makes it possible to set up multiple shortcuts on behalf of applications that span multiple domains. As a result, a multi-domain end-to-end connection can be transparently improved in terms of speed and number of hops by introducing shortcuts.

The remainder of this paper is organised as follows. Section 2 discusses the TBS architecture. Section 3 explains implementation details. The system is evaluated in Section 4. Related work is discussed throughout the text and summarised in Section 5. Finally, conclusions are presented in Section 6.

2 Architecture

At present, many techniques can be used to build end-to-end network connections with some service guarantees. For instance, Differentiated Service (DiffServ) [6] manages the network bandwidth by creating per hop behaviors inside layer-3 routers, while Multi Protocol Label Switching (MPLS) [7] establishes a path using label switched routers. However, a more radical approach that is typically adopted for layer-1 switches, uses the concept of a *lightpath* [8]. In this paper, a *lightpath* is defined as an optical unidirectional point-to-point connection with effective guaranteed bandwidth. It is suitable for very high data transfer demands such as those found in scientific Grid applications [9]. In order to set up a lightpath, a control plane separate from the data forwarding plane is used on behalf of such applications. For multi-domain control, a mechanism is needed that respects local policies and manages the network domains to set up the end-to-end connection (usually composed of multiple lightpaths) on demand.

2.1 High-Level Overview

Figure 2 shows the architecture used for setting up a *lightpath* using token based networking.

In a nutshell, the process is as follows. On behalf of a subset of its users, ISP_A generates a Link Access Request (LAR) message to use a particular link (from the available links of its peering point) for a certain period of time ①. The details of LAR generation based on user requests are beyond the scope of this article. Interested readers are referred to [10]. The LAR is sent to the AAA server of peering point A ②. The AAA server fetches a policy from the policy repository that validates the request ③. Next, a key (*TokenKey*) is generated and sent to ISP_A and peering point A ④. ISP_A may use the key to create a token for each packet that should use the link. The token will be injected in the packets which are then forwarded to the peering point. The entity responsible for token injection is known as the token builder. On the other side, peering point A uses the same *TokenKey* as ISP_A to check the incoming token-annotated packets. In other words, for each received packet, peering point A creates a local token and checks it against the embedded packet token. The packet is authenticated when both tokens match. An authenticated packet is then forwarded to its corresponding fibre-optic link ⑤. All other packets are transmitted on a 'default' link that is connected to the

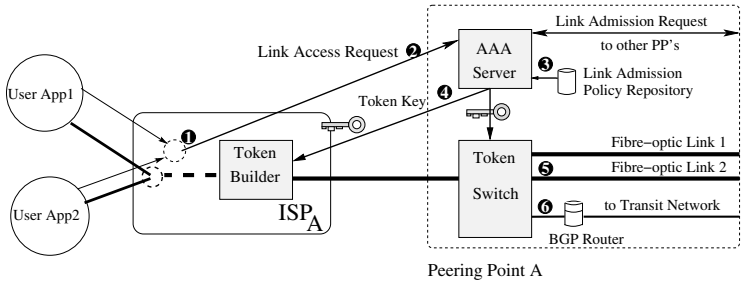


Fig. 2. Token based networking architecture

transit network ⑥. The entity responsible for token checking and packet switching is known as the token based switch.

When a packet arrives at the token switch, we must find the appropriate key to use for generating the token locally. Which key to use is identified by fields in the packet itself. For instance, we may associate a key with a IP source and destination pair, so all traffic between two machines are handled with the same key. However, other forms of aggregation are possible. For instance, we may handle all traffic between two networks with the same key, or all machines participating in a Grid experiment, etc. In general, we allow the key to be selected by means of an *aggregation identifier*, embedded in the packet. The aggregation identifier is inserted in the packet together with the token by the ISP to signal the key to use.

2.2 Token Principles

Compared to other mechanisms (such as certificates), a token is a general type of trusted and cryptographically protected proof of authorisation with flexible usage policies. A token created for an IP packet is essentially the result of applying an HMAC algorithm over a number of packet fields as illustrated in Figure 3 and explained below. An HMAC algorithm is a key-dependent way to create a one-way hash (see RFC2401 [11]). In our implementation we opted for a strong proof of authorisation by means of HMAC-SHA1. It may be possible to use more lightweight algorithms such as RC5 which is also used by IP EasyPass [5]. However, we wanted to evaluate the performance that could be achieved with strong authentication. We believe that using RC5 or similar algorithms would only make it scale better.

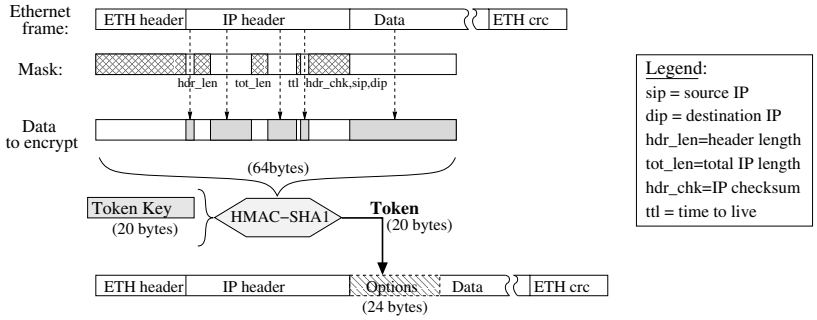


Fig. 3. Token creation

To evaluate the TBS principles, we developed a prototype that stores the token in each packet's IP option field (as defined in RFC 791). An IP option can be of variable length and its field will be ignored by normal routers. Although some routers have a slower processing path for the IP option packets than simple IP packets (because higher level headers will be at different positions in the packet), we noticed that our TBS system works well in high speed, important and pricey sites (e.g., ISPs, grid nodes interconnection points) where all systems and also routers are updated to the state-of-the-art

hardware. We stress, however, that the IP option implementation is for prototyping purposes. More elegant implementations may use a form of MPLS-like shim headers.

Figure 3 shows the process of token creation and insertion in the IP option field. In our prototype, the HMAC-SHA1 algorithm generates the unique token (20 bytes) that will be injected into the packet's IP option field. As an IP option field has a header of two bytes, and network hardware systems commonly work most efficiently on chunks of four or eight bytes, we reserve 24 bytes for the IP option field. In order to ensure the token uniqueness for packets, we need to include fields that are different in every packet. Therefore, a part of the packet data will be included together with the packet header in the HMAC calculation. We mention that some of the first 64 bytes of an Ethernet frame are masked in order to make the token independent of the IP header fields which change when a token is inserted (e.g., header length, total length, IP checksum) or when the packet crosses intermediate routers (e.g., TTL). The mask also provides flexibility for packet authentication, so that one could use the (sub)network instead of end node addresses, or limit the token coverage to the destination IP address only (e.g., by selectively masking the IP source address).

3 Implementation Details

The Token Based Switch (TBS) proposes to perform secured lightpath selection on-the-fly by means of packet based authentication. Therefore, packet checking at high speeds (Gbps) is crucial in our context. The latest generation of Network Processor Units (NPUs) includes crypto units and provides a suitable solution for packet authentication in a TBS system. Although the NPUs are powerful hardware specifically designed for packet processing at high speeds, they consist of a complex architecture (e.g., multi-RISC cores and shared memory controlled by a central GPU). Therefore, building software applications on NPUs is a challenging task. As explained below, we use Intel IXP2850 NPUs as our hardware [12] and extend the Fairly Fast Packet Filter (FFPF) software framework [13] to ease implementation of our software.

3.1 Hardware Platform

Our prototype uses the IXDP2850 development platform (see Figure 4), consisting of dual IXP2850 NPUs ① & ②, ten gigabit fibre interfaces ③, a loopback fabric interface ④ and fast data buses (SPI, CSIX). Each NPU has several external memories (SRAM, DRAM) and its own PCI bus for the control plane (in our setup it connects to a slow 100Mbps NIC). In addition, each 2850 NPU contains on-chip 16 multi-threaded RISC processors (μEngines) running at 1.4GHz, a fast local memory, lots of registers and two hardware crypto units for encryption/decryption of commonly used algorithms (e.g., 3DES, AES, SHA-1, HMAC).

As illustrated in Figure 4, the incoming packets are received by the Ingress NPU (via the SPI bus). These packets can be processed in parallel with the help of μEngines. The packets are subsequently forwarded to the second NPU via the CSIX bus. The second NPU can process these packets and then decide which will be forwarded out of the box (via the SPI bus) and which outgoing link will be used.

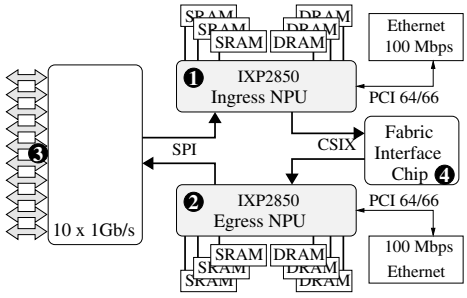


Fig. 4. IXDP2850 development platform

3.2 Software Framework: FFPF on IXPs

The Fairly Fast Packet Filter (FFPF) is a flexible software framework designed for high-speed packet processing. FFPF supports both commodity PCs and IXP network processors natively and has a highly modular design. FFPF was designed to meet the following challenges. First, it exploits the parallelism offered by multiple cores (e.g., the host CPU and the IXP's μEngines). Second, it separates data from control, keeping the fast-path as efficient as possible. Third, FFPF avoids packet copies by way of a 'smart' buffer management system. Fourth, the FFPF framework allows building and linking custom packet processing tasks inside the low-level hardware (e.g., the μEngines of each NPU). For example, a packet processing application may be built by using the FFPF framework as follows. The application is written in a simple packet processing language known as FPL [14], and compiled by the FPL-compiler. Then, the application's object code is linked with the FFPF framework and loaded into the hardware with the help of the FFPF management tools. Most of the complexity of programming low-level hardware (e.g., packet reception and transmission, memory access) is hidden behind a friendly programming language.

3.3 Token Based Switch

The TBS application consists of two distinct software modules: the token builder and the token switch (see also Figure 2). In our prototype, the modules are implemented on a single hardware development system (IXDP2850) although in reality they are likely to be situated in different locations. Therefore, our implementation consists of a demo system as shown in Figure 5.

The token builder application module is implemented on two μEngines in the Ingress NPU, while the token switch module is implemented on two μEngines in the Egress NPU. Although the mapping can be easily scaled up to more μEngines, we use only two μEngines per application module because they provide sufficient performance already. As we will see in Section 4, the bottleneck is the limited number of crypto units.

The **token builder** application implements the token principles as described in Figure 3. FFPF automatically feeds the token builder module with packet handles. The token builder retrieves the first 64 bytes of the current packet from a shared packet

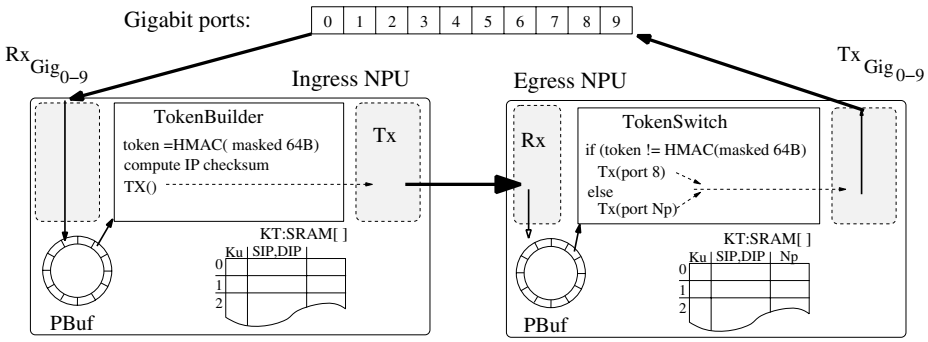


Fig. 5. Token Based Switch using a dual IXP2850

buffer memory (`PBuf`) into local registers and then applies a custom mask over these bytes in order to hide unused fields like IP header length, IP total length, etc. The application also retrieves a proper token key (`K`) from a local KeysTable by looking up an aggregation identifier (e.g., a flow may be identified by the IP source address and/or IP destination address pair, or other aggregates). The aggregation identifier also determines which fields to use in the authentication (the mask). Next, an HMAC-SHA1 algorithm is issued over the first (masked) 64 bytes of packet data using `K` (20 bytes) as encryption key. The encryption result (20 bytes) is then 'inserted' into the current packet's IP option field. This operation involves shifting packet data to make space for the option field. It also involves re-computing its IP checksum because of the IP header modification. Once the packet has been modified it is scheduled for transmission. In this prototype, the ingress NPU packets are transmitted out to the egress NPU via a fast bus.

The **token switch** application implements the token switch machine from the system architecture (see Figure 2). FFPF automatically feeds the token switch module with packet handles. The token switch checks whether the current packet has the appropriate IP option field, and extracts the first 64 bytes of the original packet data and the token key value from the option field into local registers. Next, it applies a custom mask over the 64 bytes of data. The application also retrieves a proper token key from its local KeysTable. If no entry is found in the KeysTable the packet cannot be authorised and it will be sent out to a default port (e.g., port 8) for transmission over a (slow) routed connection. Otherwise, an HMAC-SHA1 algorithm is issued over the 64 bytes of data and using the token key value (20 bytes) as encryption key. The encryption result is compared to the built-in packet token. When they match, the packet has been successfully authorised and it will be forwarded to its authorised port (`Np`).

4 Evaluation

Figure 6 shows the system setup used for proving the concepts of token based switching [15].

The two IXP2850 NPUs (Ingress and Egress) boot from a Linux boot server machine. At run-time we use FFPF for the control path (running on the Linux server and

both embedded linux NPUs). Three other linux machines (Rembrandt 6, 7 and 8) serve as clients and are each connected via gigabit fibre to an NPU gigabit port. In order to find out the maximum data rate the proposed system can handle, we evaluate the following scenario:

- An UDP traffic stream was generated (using the `iperf` tool) from Rembrandt6 to port 6 of the IXDP2850;
- A key was set for authorising traffic (Rembrandt6 $\rightarrow$ 7) to end up on port 7 and another key was set for (Rembrandt 7 $\rightarrow$ 6) to end up on port 6;
- Unauthorised traffic goes to the default port (port 8);
- To prove that authorised traffic ends up on port 7 and unauthorised traffic ends up on port 8, Rembrandt7 and 8 were connected to the IXDP2850 ports 7 and 8, respectively. We used the `tcpdump` tool on Rembrandt7 and 8 for listening to their gigabit ports.

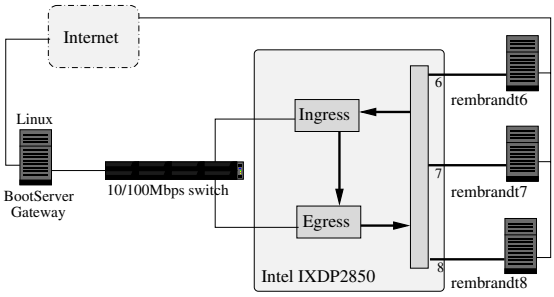
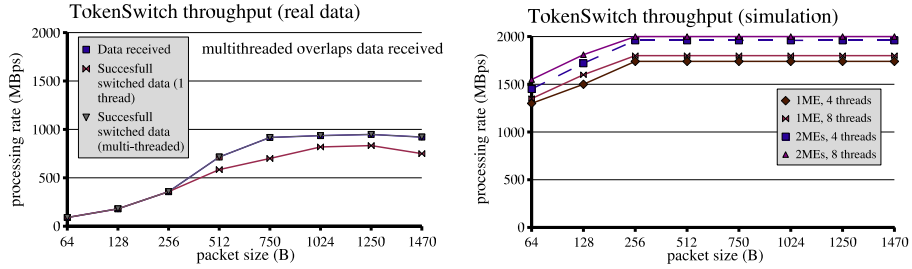


Fig. 6. Token Based Switch demo

The performance of the above test is shown in Figure 7.a. It has two charts: ① 'data received' which represents the received rate in the IXDP2850 box and ② 'successfully switched data' which denotes the rate that the TBS could handle properly using just a single thread for processing. The 'data received' chart is low for small packets because of the Gigabit PCI card limitation used in the Rembrandt6 PC for traffic generation. So, for small packets it reflects the limitations of the traffic generator rather than those of the TBS. The second chart, 'Successfully switched data', is lower than the first one for high speeds because here we are using a single threaded implementation. The multithreaded version coincides exactly with the 'data received' chart and is thefore not visible.

While we cannot predict the real performance for speeds above 1 Gbps without performing measurements with a high-speed traffic generator, we estimated the outcome by using the Intel's cycle accurate IXP simulator running in debug mode. Table 1 shows the cycle estimation for a 150 bytes packet processed by each software component from the Data path (Rx, Tx, Rx_CSIX, Tx_CSIX, TokenBuilder and TokenSwitch). Table 1.a illustrates the cycles spent for one packet in each software module of the FFPF implementation on IXP2850. These modules are optimised for multithreading packet processing (e.g., receiving, storing, transmitting). The first row in Table 1.b contains the cycles spent for one packet in a single threaded version of the token builder and



ⓐ running in hardware ⓑ running in cycle accurate simulator

Fig. 7. Token Based Switch performances

token switch modules. We note that these values are high because all subtasks (e.g., encryption, token insertion, checksum computation) run linearly (no parallelism involved at all) and use only one crypto unit each. This single threaded version gives the performance shown in Figure 7.a. The next rows illustrate various implementations of the multithreading version. Although we should expect better performance when we increase parallelism (e.g., more threads, or more μEngines), having only two crypto units available per NPU limits the performance to the value of roughly 2000 cycles (the token switch module spends its time mostly on authentication, while the token builder module does also token insertion in the packet).

Note that our prototype implements each TBS module (token builder and token switch) on only one NPU of the IXDP2850 hardware system. The reason this was done is that we have only one of these (expensive) IXDP2850 devices available in our lab. In a real setup, however, each TBS module may use the full dual-NPU IXDP2850 for building or checking tokens and therefore the system performance is expected roughly to double compared to our presented figures, mainly because we would benefit from the availability of four crypto units.

Table 1. Cycle budget

FFPF module	μEngine cycles
Rx	408
Tx_CSIX	276
Rx_CSIX	504
Tx	248

(a) FFPF on IXP2850 overhead

TBS	TokenBuilder	TokenSwitch
single threaded	5777	3522
4 threads, 1 μEngine	3133	2150
8 threads, 1 μEngine	3000	2100
4 threads, 2 μEngines	2600	2100
8 threads, 2 μEngines	2500	2000

(b) TBS overhead

Using the cycle estimation given in Table 1, we express the throughput as a function of the packet size, number of threads and number of μEngines: rate=f(packet size, threads, μEngines) without taking into account additional contention. The estimated throughput for our multithreaded version goes up to roughly 2Gbps (Figure 7.b).

We also measured the latency introduced by our Token Based Switch system. The token builder application (the whole Ingress NPU chain) takes 13.690 cycles meaning

a 9,7μs processing time (introduced latency), and the TokenSwitch application (the whole Egress NPU chain) takes 8.810 cycles meaning a 6,2μs latency. We mention that a μEngine in the IXP2850 NPU runs at 1400MHz.

5 Related Work

In addition to commercial solutions for single domain provider-controlled applications such as Nortel DRAC, Alcatel BonD, some research is also underway to explore the concept of user-controlled optical network paths. One of the leading software packages is the User Controlled Lightpath Provisioning (UCLP) [16]. UCLP currently works in a multi-domain fashion, where all parties and rules are pre-determined. Truong et al [17] worked on policy-based admission control for UCLP and implemented fine-grained access control.

Some interesting work in the optical field is also done in Dense Wavelength Division Multiplexing-RAM [4,18], where a Grid-based optical (dynamic) bandwidth manager is created for a metropolitan area. Our approach is different in the sense that we provide a mechanism to *dynamically* set up *multiple shortcuts* across a multi-domain end-to-end connection. Therefore, an end-to-end connection can be easily improved in terms of speed and hop count by introducing 'shortcuts' based on new user's agreements.

IP Easy-pass [5] proposed a network-edge resource access control mechanism to prevent unauthorised access to reserved network resources at edge devices (e.g., ISP edge-routers). IP packets that are special demanding, such as real-time video streams, get an RC5 encrypted pass appended. Then, at edge-routers, a linux kernel validates the legitimacy of the incoming IP packets by simply checking their annotated pass. Unlike our work, the solution aims at fairly low link rates. While our solution shares the idea of authentication per packet (token), we use a safer encryption algorithm (HMAC-SHA1) and a separate control path for key management (provided by AAA servers). In addition, we demonstrate that by using network processors we are able to cope with multi-gigabit rates.

Most related to our TBS is Dynamic Resource Allocation in GMPLS Optical Networks (DRAGON) framework [19]. This ongoing work defines a research and experimental framework for high-performance networks required by Grid computing and e-science applications. The DRAGON framework allows dynamic provisioning of multi-domain network resources in order to establish deterministic paths in direct response to end-user requests. DRAGON's control-plane architecture uses GMPLS as basic building block and AAA servers for authentication, authorisation and accounting mechanism. Thereby, we found a role for our TBS within the larger DRAGON framework and we currently work together to bring the latest TBS achievements into DRAGON.

6 Conclusions and Future Work

This paper presents our implementation of the Token Based Switch application on Intel IXP network processors, which allows one to select an optical path in hybrid networks. The admission control process is based on token principles. A token represents the

right to use a pre-established network connection in a specific time frame. Tokens allow separation of the (slow) authorisation process and the real-time usage of high-speed optical network links. The experimental results show that a TokenSwitch implementation using the latest Network Processor generation can perform packets authorisation at multi-gigabit speeds.

Acknowledgements

This work was supported by the GigaPort NG and the EU IST NextGrid projects. The authors wish to thank to Harry Wijshoff and Dejan Kostic for their feedback and Lennert Buytenhek for his advice and support.

References

1. Winkler, L.: The hybrid optical and packet infrastructure (HOPI) testbed. Internet2 whitepaper (April 2004)
2. Vollbrecht, J., Calhoun, P., Farrel, S., Gommans, L., Gross, G., Bruin, B., de Laat, C., Holdrege, M., Spence, D.: RFC2904, AAA Authorization Framework. IETF (2000)
3. Kent, S., Seo, K.: Security Architecture for the Internet Protocol. RFC 4301 (Proposed IPsec Standard) (December 2005)
4. Figueira, S., Naiksatam, S., Cohen, H.: DWDM-RAM: Enabling Grid Services with Dynamic Optical Networks. In: Proc. of the SuperComputing, Phoenix, Arizona (Nov 2003)
5. Wang, H., Bose, A., El-Gendy, M., Shin, K.G.: IP Easy-pass: a light-weight network-edge resource access control. IEEE/ACM Transactions on Networking **13**(6) (2005)
6. Blake, S., D.Black, Carlson, M., Davies, E., Wang, Z., Weis, W.: RFC2475, An Architecture for Differentiated Services. IETF (1998)
7. Rosen, E., Viswanathan, A., Callon, R.: RFC3031, Multiprotocol Label Switching Architecture. IETF (2001)
8. DeFanti, T., de Laat, C., Mambretti, J., Neggers, K., Arnaud, B.S.: TransLight: a global-scale LambdaGrid for e-science. Commun. ACM **46**(11) (2003) 34–41
9. de Laat, C., Radius, E., Wallace, S.: The rationale of current optical networking initiatives. Future Generation Computer Systems **19**(6) (Aug 2003) 999–1008
10. Gommans, L., Dijkstra, F., de Laat, C., Taal, A., Wan, A., T., L., Monga, I., Travostino, F.: Applications drive secure lightpath creation across heterogeneous domains. IEEE Communications Magazine **44**(3) (March 2006) 100–106
11. Bellare, M., Canetti, R., Krawczyk, H.: RFC2104, HMAC: Keyed-Hashing for Message Authentication. IETF (1997)
12. Intel Corporation: Intel IXP2xxx Network Processor. IXP NPs product brief (2005)
13. Bos, H., de Bruijn, W., Cristea, M., Nguyen, T., Portokalidis, G.: FFPF: Fairly Fast Packet Filters. In: Proceedings of OSDI'04, San Francisco, CA (December 2004)
14. Cristea, M.L., de Bruijn, W., Bos, H.: Fpl-3: towards language support for distributed packet processing. In: Proceedings of IFIP Networking, Waterloo, Canada (May 2005)
15. Gommans, L., Travostino, F., Vollbrecht, J., de Laat, C., Meijer, R.: Token-based authorization of connection oriented network resources. In: Proc. GRIDNETS, San Jose, CA, USA (Oct 2004)

16. Wu, J., Campbell, S., Savoie, J., Zhang, H., Bochmann, G., Arnaud, B.: User-managed end-to-end lightpath provisioning over ca*net 4. In: Proceedings of the National Fiber Optic Engineers Conference, Orlando, FL, USA (Sep 2003)
17. Truong, D., Cherkaoui, O., ElBiaze, H., Aboulhamid, M.: A Policy-based approach for User Controlled Lightpath Provisioning. In: Proc. of NOMS, Seoul, Korea (Apr 2004)
18. Figueira, S., Naiksatam, S., Cohen, H.: OMNInet: a Metropolitan 10Gb/s DWDM Photonic Switched Network Trial. In: Proceedings of Optical Fiber Communication, Los Angeles, USA (Feb 2004)
19. Lehman, T., Sobieski, J., Jabbari, B.: DRAGON: A Framework for Service Provisioning in Heterogeneous Grid Networks. IEEE Communications Magazine **44**(3) (March 2006)