A SIMPLIFIED, COST-EFFECTIVE MPLS LABELING ARCHITECTURE FOR ACCESS NETWORKS

Harald Widiger[+], Stephan Kubisch[+], Daniel Duchow[+], Thomas Bahls[*], Dirk Timmermann[+]

[+] University of Rostock
Institute of Applied Microelectronics and Computer Engineering
18051, Rostock, Germany
Tel./Fax: ++49 (0)381 498 - 7276/7252
{harald.widiger,stephan.kubisch,daniel.duchow,dirk.timmermann}@uni-rostock.de
[*] Siemens AG Communications, 17489 Greifswald, Germany
thomas.bahls@siemens.com

Keywords: Access Networks, MPLS, Reconfigurable Hardware

Abstract

Today, an increasing number of customers subscribes for a high bandwidth internet access. But not only communication speed is demanded. Quality of Service (QoS) moves more and more into the customers focus. Both Carriers and Internet Service Providers (ISPs) have increasing requirements derived from new services they want to offer to their customers. A new hardware solution is presented, which can satisfy many of these upcoming demands, the MPLS User Network Interface (MPLS-UNI). The solution offers reduced MPLS functionality to very cheap hardware costs. All incoming frames are provided with MPLS label stacks. From frames leaving the network the label stacks are removed. The MPLS-UNI works with wirespeed. Only a negligible delay is inserted into the data path.

1 Introduction

Today, more and more customers subscribe for a high bandwidth internet access. But not only speed is demanded. Reliability, availability, and security move more and more into the customers focus [7].

As illustrated in Figure 1, the architecture of current access networks consists of various aggregation levels. The main aggregation points are the line cards at the customers' side of the access network supporting Gbit-Ethernet, the central nodes and broadband access servers at the core side supporting multiple Gbit-Ethernet streams and fibre optics. Every device in the data path has to analyse and process frame parameters like source and destination addresses, Quality of Service (QoS) information, protocol types, and checksums to direct the data streams through the access network.

Actually, home users favour the use of Ethernet based DSL while business customers prefer connecting their own Local Area Network (LAN) directly to the carriers access network. Thus, in the future, access networks have to deal not only with a single customer connected to one line of the line card, but as illustrated in Figure 2, they must aggregate whole LANs at the line cards. Besides high bandwidth the customers demand QoS, diverse services, and traffic differentiation.

In order to meet these various demands, it is necessary to enrich the different frames with additional information. This kind of information like port information is only available within the access network, but is required at other locations of the network. A technique to code information into a frame is the use of VLAN tags. The great disadvantage of VLANs is the limitation to twelve bit. A fine granularity as desired is impossible. Stacked VLANs (QiQ) would

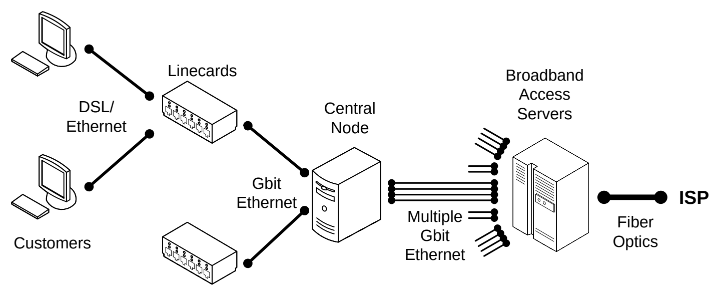

Figure 1: Current Access Network Architecture

produce relief, but common network equipment is not necessarily able to handle QiQ without modifications. Due to the fact that the central nodes (Digital Subscriber Line Access Multiplexer - DSLAMs) are usually located at the edges of an MPLS switched environment, using MPLS labels to carry information through the network comes to mind. An MPLS label offers at least twenty bit for information coding.

The remainder of the paper is organised as follows. Section 2 focuses on the functionality for the MPLS-UNI. Section 3 addresses the architecture of the hardware implementation of the module and the necessary submodules. Implementation results are presented in Section 4. Finally, Section 5 concludes the paper.

2 MPLS-UNI

MPLS is an encapsulation scheme [3]. In MPLS switched networks, frames are forwarded based on MPLS labels within each frame (Figure 3).

At the edges of an MPLS network, incoming frames are assigned a label stack [5]. That task is usually performed by Label Edge Routers (LERs). Then, frames are forwarded along a Label Switched Path (LSP) where each Label Switched Router (LSR) makes forwarding decisions based on the content of the labels in the MPLS label stack.

In order to select correct labels for each frame and to update the switching tables of all LERs and LSRs, a label distribution protocol (LDP) [1] is required. At the egress points of the MPLS network, the label stack is removed from the frame and the original frame is restored.

Instead of using MPLS for switching and routing, it is intended to use the MPLS label stack to carry the information mentioned in Section 1. That means, the MPLS labels are not used to make switching decisions. In consequence, no LER implementing the entire LDP is required. However, even in a design without an LDP, the use of MPLS for switching and routing is not excluded. But routing and switching are not within the scope of this paper. The

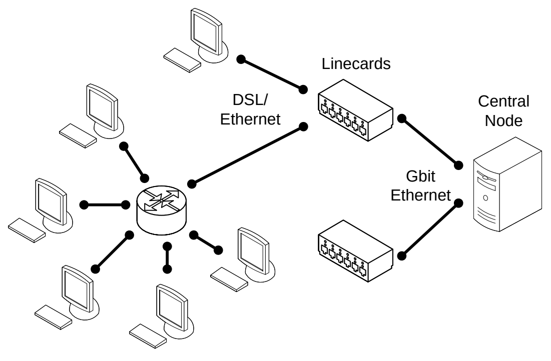

Figure 2: Customer Connections to Access Networks

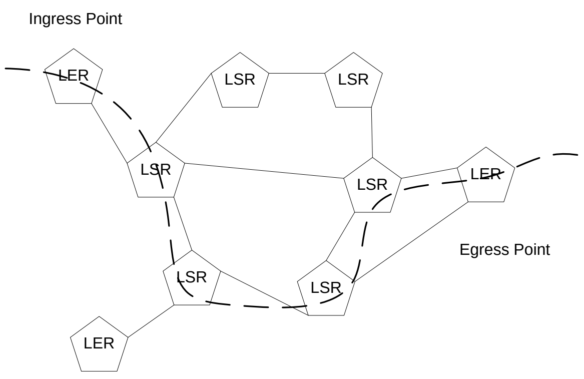

Figure 3: Path of a frame through an MPLS switched network from ingress point to egress point

reduced functionality can be realised with a simplified and cost-effective hardware solution for an MPLS User Network Interface (UNI) as proposed in this paper. Besides these advantages, being a hardware solution, the MPLS-UNI inserts no unnecessary additional delay into the data path.

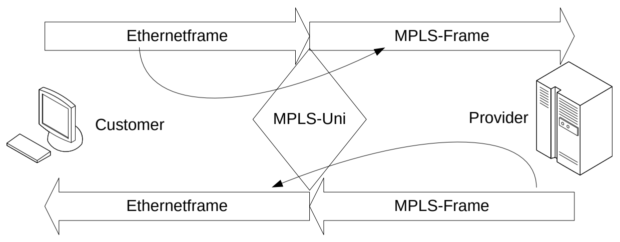

Figure 4: MPLS Labeling

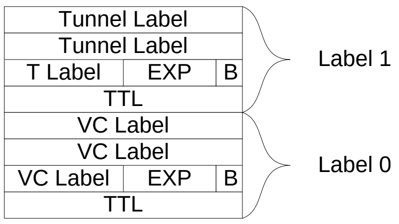

Figure 5: Added Labelstack

The modules purpose is to add an MPLS label stack to all incoming frames in upstream direction and to forward them to the providers' core networks. In downstream direction, the label stacks of all incoming frames are striped off the frame as shown in Figure 4.

The MPLS label stack, inserted by the MPLS-UNI, consists of two different labels as pointed out in Figure 5. The inner label usually describes the virtual channel through the MPLS network. The outer label describes the actual path the frame has to take passing through the network from an ingress point to the corresponding egress point. However, with two labels in the frame at least 40 bits are available for any intended application or information to be carried. Thus, a huge number of different services can be distinguished. VoIP data, for example, might use a much faster route than standard internet traffic. Even if the 40 bits are not sufficient, more labels can be added to the stack.

We are using two labels at the moment to code information. But, as the number of labels in an MPLS label stack is not limited by the protocol, a virtually unlimited number of bits are available to code any desired information into a frame.

3 Hardware Architecture

The MPLS-UNI module is a hardware solution and performs its tasks with wirespeed. It was implemented in VHDL and is to be inserted as an FPGA between the central nodes and the broadband access servers in an access network. A block diagram of the architecture is shown in Figure 6.

In upstream direction, the module extracts a key from the headers of each incoming frame. The key identifies a customer or a group of customers. It can consist of different fields.

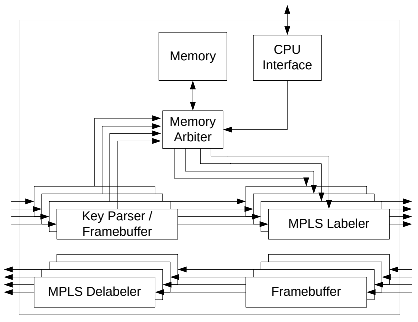

Figure 6: Architecture of the MPLS-UNI

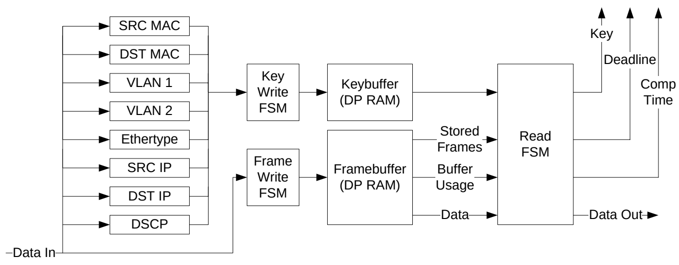

Figure 7: Architecture of Key Parser/Framebuffer

The MPLS label information corresponding to the key is searched in a memory. In order to increase lookup speed, all keys are stored in a sorted memory. This way a binary search can be performed to find the information. This reduces complexity of required number of memory accesses to $O(lg(N))$ with N being the number of keys in the memory. That is unlike the embedded MPLS architecture in [4], which needs a linear search time. The frame is MPLS labelled with the information retrieved from the memory. This means, an MPLS Labelstack is inserted into the frame.When using a CAM memory, the complexity of memory accesses could be reduced to $O(1)$. But that would mean introducing an extra of-chip memory to the system with negative results regarding flexibility and costs.

In order to process a data rate of 1 Gbit per second, the MPLS-UNI has to work with a frequency of 125 MHz. On each clock cycle one byte of data is transferred through the data path. As can be seen in Figure 6, multiple parallel data paths can be used. To reach the desired throughput of four Gbit per second, four parallel data paths are implemented as desired in the access network environment.

The functionality in downstream direction is simpler than the upstream functionality. Here, the MPLS labelled Ethernet frames are unlabelled and forwarded. No memory lookups are necessary. In the following subsections, the detailed architecture of the submodules is presented.

3.1 Key Parser with Framebuffer

As mentioned above, the Key Parser is capable of parsing different fields of incoming Ethernet frames. Possible fields are source and destination MAC address, VLAN tags, Ethertype, source and destination IP address, and the DSCP field in the IP header. Any combination of these fields can be configured at compile time.

As pointed out in Figure 7, there is an independent functionality to parse each of the mentioned fields. The parsing results are combined and stored in a First In First Out memory (FIFO) of the FPGA. The whole frame is stored in another FIFO. The size of the frame buffer is configurable. Depending on the frame buffer's size, the key buffer's size is chosen automatically to ensure that in any case keys for each stored frame can be buffered, even if only minimal frames of 60 byte are stored.

In order to reduce the area complexity of the MPLS-UNI, the developer configures which fields of the Ethernet frame shall be parsed by setting a bit vector in a configuration VHDL file. This way only the required parsing functionality is implemented in the key parser.

Controlled by a read Finite State Machine (FSM), both frame buffer and key buffer are read out. The frame buffer presents two four-bit values which describe the fill level of that buffer. Buffer Usage shows the real fill level ("0000" - buffer is empty; "1111" - buffer is full), and Stored Frames shows the number of frames in the buffer. This information is required by the memory arbiter described below.

3.2 Memory Arbiter

The Memory Arbiter's function is to control access to the memory. As there are four key parsers in upstream direction, four independent keys may arrive at the memory at the same time. To schedule these concurrent memory accesses, a memory arbiter (Figure 8) is required. The memory arbiter decides which of the keys competing for the memory access is to be sent to the memory to find the corresponding entry. This decision is made by applying the Least Laxity First (LLF) algorithm. The LLF, usually used for process scheduling in operating systems, assigns the memory access to the key with the smallest slack. In process scheduling, the smallest slack is computed as difference between two parameters of the process: the deadline to meet and the computation time required. In case of two equal slack values, the process with the smallest deadline is scheduled. In our application, the deadline derives from the space left in the frame buffer. The computation time (Comp Time) derives from the number of frames that are already stored in the frame buffer. Together with the key, both deadline and computation time are presented as four bit values to the memory arbiter. The key with the smallest slack gets access to the memory. We considered using the Enhanced LLF (ELLF) algorithm as presented in [2]. ELLF enhances the LLF in the way that ”thrashing”, meaning unnecessary task switches, is avoided. This feature has no use in our application. In contrary to task switching in operating systems, no costs arise from changing the key parser unnecessarily. Thus, the simple LLF algorithm was chosen for implementation.

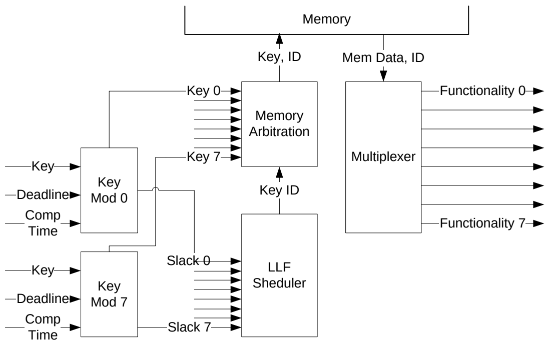

Figure 8: Architecture of the Memory Arbiter

As well as the key parsers, the CPU Interface competes for memory accesses. The CPU interface is responsible administrative access to the MPLS-UNI. In order to maintain validity of all memory entries, CPU accesses have the highest priority compared to the key parsers.

3.3 Memory

The memory, where the information for the MPLS entries is stored, can be implemented either with external Dynamic RAM (DRAM) or with the FPGAs internal BRAMs. That depends on the size and number of necessary entries in the memory. A memory entry consists of the key and the corresponding two 20-bit MPLS labels as can be seen in Figure 9.

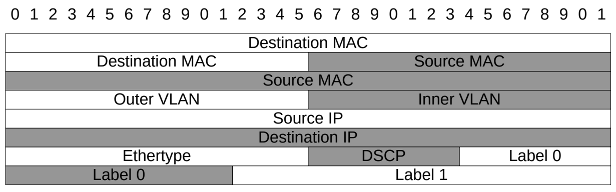

Figure 9: Maximum memory entry of MPLS-UNI

As the key can consist of a maximum of 216 bits (all seven possible header fields), 256 bit entries are the largest configuration to be implemented. As pointed out before, all entries are stored in a sorted manner in the memory. That reduces the complexity of finding a memory entry to $O(N + \log 2(N))$. Considering four 1 Gbit channels with minimum frames with the size of 64 byte and an inter frame gap of 12 byte, 28 clock cycles are available for each frame to find a memory entry without having to drop frames. Theoretically 228 entries can be searched in that time. It has of course to be mentioned, that the implemented search algorithm cannot analyse a memory entry each clock cycle.

In fact two clock cycles are required.

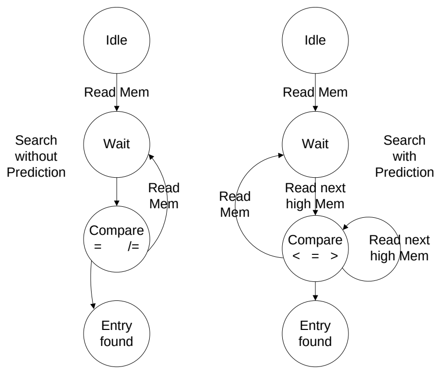

Figure 10: Searching with and without a prediction

After setting the read address of a certain memory entry, it takes one wait cycle until the memory outputs are valid. Then a comparator can decide if an entry is the correct one or the upper respectively lower half of the unsearched memory has to be searched further. To increase look up speed a prediction is executed. After setting a read address, the next probable memory address is assigned in the following clock cycle. According to the search algorithm, the following address in the upper half of the unsearched memory is selected as next address. In half of the cases, this assumption is correct. This way, in average 1.5 instead of 2 clock cycles per memory lookup are necessary. As there is never a Gbit Ethernet link under full load with only minimal frames in a real scenario, even memory accesses should suffice. Assuming an average frame size of 400 Byte, 100 clock cycles are available for the search algorithm. The maximal frequency of the memory module is determined by the comparison of the key with the stored entry. To maintain a minimal frequency of 125 MHz the comparator width must be limited. Therefore the search algorithm is configurable. The comparison can be done in 1 to 3 steps depending on the selected key width and the speed grade of the utilized FPGA. To increase the step number even further, only minor changes of the code are necessary. This of course increases the number of clock cycles necessary for a search, as up to three compare states are to be traversed. However, depending on the correlation degree of the stored entries, mostly only one ore two compare states have to be traversed.

Sorting the memory entries of course has the disadvantage of increased time consumption for insertion and deletion. Both operations have a time complexity of $O(N + \log 2(N))$. In average, $N/2 + 2 \cdot \log 2(N-1)$ memory accesses are required. As changes in the look up table occur quite seldom in access network environment, these additional costs are acceptable when increasing lookup speed from $O(N)$ to $O(\log 2(N))$.

3.4 MPLS-Labeler and -Delabeler

For setting up the frames with an MPLS label stack, the labeling scheme, proposed by Martini et al. in [6], is used to conform with the MPLS hardware within the core network. This frame structure is shown in Figure 11. Here, the frames are encapsulated completely. A new Ethernet header together with the created MPLS label stack is added in front of the frame. The CRC checksum at the end of the frame is replaced by a new CRC value calculated over the entire new frame. In downstream direction, both Ethernet header and MPLS label stack are removed from the frame and the CRC is recalculated before transmitting the frame.

4 Implementation Results

A first implementation was accomplished using a Xilinx Virtex4 XCVFX20 FPGA with the speed grade 11 (Figure 12). The amount of reconfigurable resources needed for the design considerably depends on the configuration of the MPLS-UNI. The MPLS-UNI can be configured in different ways.

Firstly, the structure of the keys of the functional module must be defined. Secondly, the number of Gbit channels used

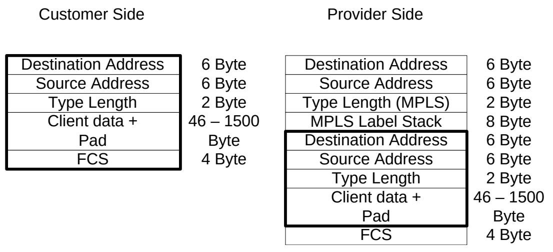

Figure 11: Frame Structures

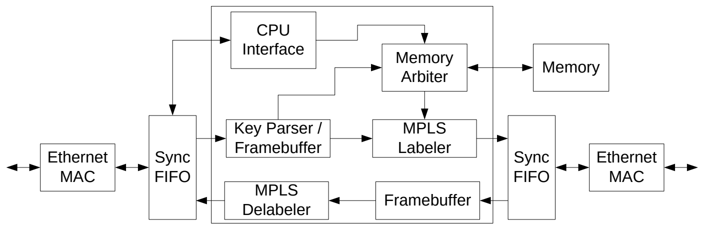

Figure 12: Implementation of the MPLS-UNI

must be specified. These configurations can be made by changing constant values in a configuration VHDL file. The complete functionality of the MPLS-UNI for one Gbit channel was synthesised. Depending on the configuration of the keys, the MPLS-UNI required 1125 slices when configured with a minimal key size. Configuring a maximum key size, the number of slices increased to 2227. Typically 1500 slices are required. Typically means that the key consists of the source MAC address, the destination IP address and one VLAN tag (96 Bit). Additionally, logic was needed to connect the MPLS-UNI to two Ethernet interfaces and to implement an internal memory as described in Section 3.

The CPU interface has been implemented into the data path for Simulation purposes. Triggering on the destination MAC address, frames destined to the CPU interface are filtered in one of the synchronisation FIFOs. For media access control, hardwired Ethernet MACs provided by the FPGA were used. The module operated at a maximum speed of 130 MHz, although at least 125 MHz are sufficient to handle a data rate of 1Gbit/s. When running at 125 MHz, the module is capable of computing 4 Gbps with frame sizes between 100 and 150 Byte depending on the number of keys in the memory(Figure 13).

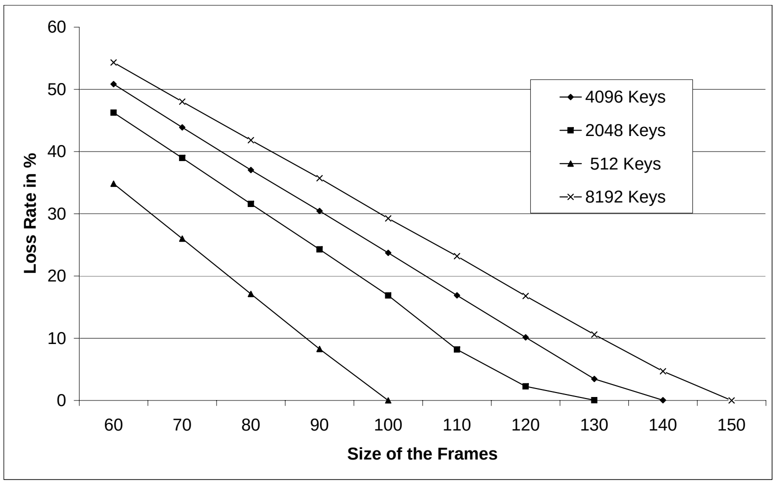

Figure 13: Performace of the MPLS-UNI when driven with data rates of 4 Gbps and different frame sizes

Next steps comprise in-field tests within a testbed provided by Siemens AG, Greifswald, and implementation into access network architecture.

5 Conclusion

The proposed hardware solution provides a powerful and cost-efficient possibility to expand MPLS networks into the access area. The module computes the data streams with wirespeed. Thus, nearly no additional delay in the data path is generated. At a speed of 125 MHz, it can handle data rates of 4 Gbit per second in up- and downstream direction. Due to the fact that our solution is designed for reconfigurable hardware, the functional spectrum can be

Module	**Slices min/typical/max**	**f(MHz)**
MPLS Labeler	120	187
MPLS Delabeler	101	322
Memory Arbiter	152 / 203 / 343	386
CPU Interface	640	210
Key Parser & Framebuffer	336 / 535 / 720	168
Framebuffer	205	185
∑ MPLS-UNI	1125 / 1496 / 2227	170
Memory (internal for 1K entries)	633 / 1049 / 1594	126
Sync FIFOs + MACs	850	169
∑ System	2600 / 3400 / 4700	130

Table 1: Implementation Results

broadened and adapted to future tasks. By inserting the MPLS-UNI module, the workload of the central nodes and switches is not increased considerably. The CPU controlling the access network has only two additional tasks. It has to configure the MPLS tables in the hardware module and to create the labels that are to be configured. The configuration can be done by an administrator. As outlined above, it is not needed to implement any LDP in the MPLS-UNI.

This work is done in cooperation with Siemens AG Communications, Greifswald.

References

[1] L. Anderson. LDP Specification. RFC 3036, January 2001.

[2] J. Hildebrandt, F. Golatowski, and D. Timmermann. Scheduling Coprocessor for Enhanced Least-Laxity-First Scheduling in Hard Real-Time Systems. In *Proc. of the 11th Euromicro Conference on Real-Time Systems*, York, GB, 1999.

[3] L. Martini. Multiprotocol Label Switching Architecture. RFC 3031, January 2001.

[4] R. Peterkin and D. Ionescu. Embedded mpls architecture. In *Proceedings of the 19th International Parallel and Distributed Processing Symposium (IPDPS 2005)*, Denver, CA, USA, 2005.

[5] E. Rosen. MPLS Label Stack Encoding. RFC 3032, January 2001.

[6] E. Rosen, A. Vishwanathan, and R. Callon. Encapsulation Methods for Transport of Layer 2 Frames Over IP and MPLS Networks. work in progress, February 2005.

[7] R. Santitoro. Metro Ethernet Services - A Technical Overview. White paper, 2003.