Frame Assembly in Packet Core Networks – Overview and Experimental Results

Wolfram Lautenschläger, Alcatel-Lucent Bell Labs, Stuttgart
Arthur Mutter, Sebastian Gunreben, University of Stuttgart, IKR

Abstract

Literature has proposed Frame Assembly and its variants multiple times to cope with the ever increasing switching density in consequence of increasing link rates. Nevertheless, state-of-the-art networks do not implement and apply it. Skepticism of practitioners and investors regard not only the effective gain of frame switching, but also questions of control, interfacing and performance impact on the existing Ethernet/IP infrastructure.

We present an operational prototype network with frame assembly in its core that seamlessly interfaces to existing Ethernet technology and seamlessly integrates to a standard conform GMPLS control plane. We show the manageable additional effort of assembly at the network edge, the direction how to integrate such network into existing control structures, but also the limited and well controlled impact of assembly on the timing of client application.

1 Introduction

In a packet based end-to-end communication, the application in the end systems initially defines the traffic characteristics, especially the packet size.

This section first highlights the dominance of small packets and then identifies the problem of small packets within core networks. The second section proposes our mitigation of Frame Assembly to overcome this problem in high-speed networks. This section closes with the related work and the overview on the organization of this paper.

1.1 Packet transport networks

The driver for packet based transport networks are the packet based customer networks and the increasing popularity of the Internet. Both of them base on the Internet Protocol (IP).

In the customer networks as well as in the access networks the line rates increased due to the increasing data volume exchanged, e.g. video and other bandwidth hungry applications.

The TCP/IP stack enables an end-to-end communication in networks showing small line rates and high latencies. Consequently, the applications and transport protocols adapted to these requirements.

Protocols for congestion control and reliable transport mechanisms enable a robust e2e communication. These protocols (e.g. TCP) use an acknowledgement mechanism for signaling between the sender and the receiver. In general, these acknowledgements have a small packet size below 100 Byte.

For performance issues, applications reduced their packet size to avoid large transmission delays and packet loss due to congestion. Time critical applications for voice and video and narrow band applications also use packet sizes in the range of 100 to 250 Byte (P and B frames in a video application).

While the network technology and especially the line rate changed during the time, the applications and transport protocols did not fully adapt to the new environment. A recent study of [5] shows that about 50% of the packets have a packet size smaller than 100 Byte.

Besides this, IETF RFC879 recommends end-systems to accept at least 576 Byte packets. This resulted in operating systems using exactly this transfer unit. Although this recommendation is from 1983 packet size distributions from the core network are still able to identify this peak.

In general, due to the dominance of Ethernet in the access the maximum transfer unit is 1500 Byte.

Consequently, the applications and transport protocols do not exploit the maximum transfer unit. With the increasing line rate in the access, the packet rate, especially the packet rate of small packets, also increases. As a result, in the core network, this burden requires unnecessary fast header processing capabilities, which is the most complicated and power consuming task in packet core nodes.

1.2 Packet rate reduction

There are two options to exploit the maximum transfer unit to reduce the packet rate: (1) change the protocol and application behavior in the end systems to exploit the maximum transfer unit; (2) assemble small packets into larger containers in the network.

The first solution requires changes in the end systems. This is in general not possible for a network operator as its influence is quite limited.

The second solution requires changes only in the network and is independent from the end system protocols and applications and thus applicable for network operators.

In the following, we consider only the second option depicted in Figure 1. A network architecture performing packet assembly requires a special node. This node assembles packets in larger containers. Timer or size thresholds limit the number of packets per container. The containers travel the network until the destination node for disassembly. The disassembly node forwards the indi-

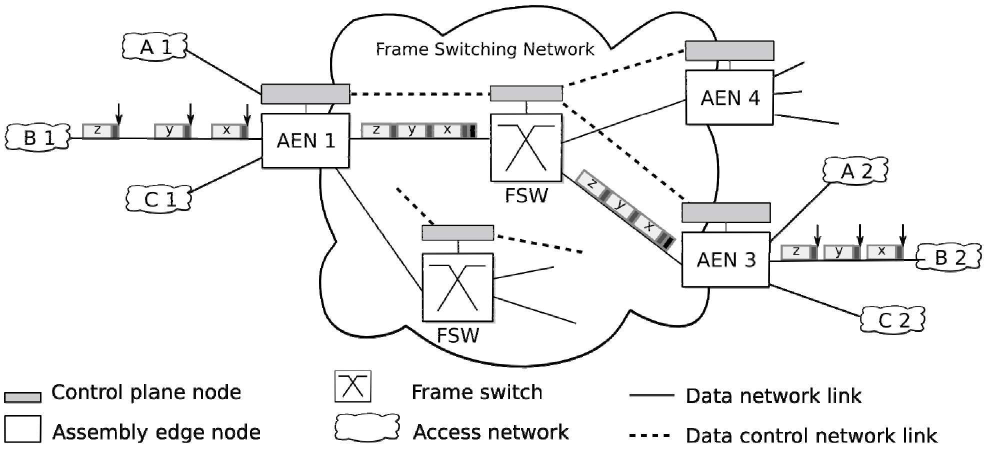

Figure 1: Frame switching network architecture

vidual packets in a burst to the access networks, respectively.

Assembled packets and disassembled packets show different traffic characteristics. This fact is one of the major argues against any frame assembly in the network, although the impact on applications is hard to quantify.

In this paper, we quantify the impact of packet assembly on the traffic characteristic by formal methods and measurement in a testbed. For this purpose, we designed and realized a bidirectional assembly node as well as a complete testbed to show the packet assembly concept is functional.

1.3 Related Work

Literature presents several architectures and implementations of assembly nodes as well as testbeds within the context of Optical Burst Switching (OBS) networks, e.g. [1].

For frame switching networks to the best of our knowledge, only Kornaros et al. describe in [9] an assembly node architecture. The authors present the nodes' ingress direction able to assemble packets into fixed sized frames. They present timer and threshold based assembly at a line rate of 10 Gbps. Nevertheless, the work lacks the egress direction with the disassembly part and neglects fragmentation.

[8] shows a detailed investigation on the traffic characteristics of assembled traffic. They provided the theoretic background but did not consider the practicability in a real network scenario. We applied their methodology and provided a worst case estimation for realistic packet assembly networks.

For the special case of self-similar traffic, Hu provided a detailed analysis on the effect of the assembled traffic in [11]. We restrict our analysis to the affected timescale in the range of the maximum frame assembly time (≈1ms), where the effect of self-similarity is negligible in core networks.

1.4 Organization of the paper

In section 2 we introduce the frame switching architecture and highlight the principle mechanisms. Section 3 is dedicated to the major device in a frame switching network, section 4 show the implemented demonstrator scenario. We quantify the impact of frame assembly in section 5 and close our paper with future work and conclusion in section 6 and 7 respectively.

2 Frame Switching Architecture

Packet assembly of multiple packets avoids the small packets dilemma. Bursts, frames, or containers are the names of the resulting aggregates. The terminology depends on the particular transport technology. Throughout this paper, we call the process frame assembly and the resulting aggregates frames.

This section introduces the Frame Switching Architecture, which performs frame assembly to reduce the packet rate in core networks. It first introduces the basic concepts and classifies the procedure with respect to today's framing procedures. The discussion on the application of packet assembly in the network and the proposed switching principle complete this section.

2.1 Basic Concept

Figure 1 shows a Frame Switching (FS) network. It consists of edge nodes, called Assembly Edge Node (AEN), and core nodes, called Frame Switches (FSW).

At ingress, the AEN assembles packets into container frames while the egress AEN performs the disassembly process. The FSW in between forward the frames from ingress to egress AEN.

The processing delay in an intermediate FSW depends on the frame size. As the line rate increases in parallel to the frame size, the time available to process a frame stays nearly the same. However, the required processing effort per node is constant if packet size and line rate increase in parallel.

Originally, Frame Switching was introduced with modified ITU-T G.709 containers of the Optical Transport Network (OTN). However, other technologies like Ethernet may serve as a container frame format. Mandatory to both is the limitation of the frame size in both technologies (Ethernet Jumbo frames max. 9.6 kByte, G.709 15.2 kByte). Due to the size limitation, assembled

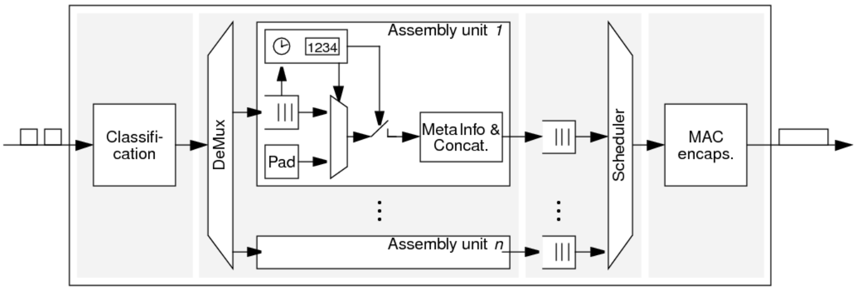

Figure 2: Functional architecture of ingress FAU

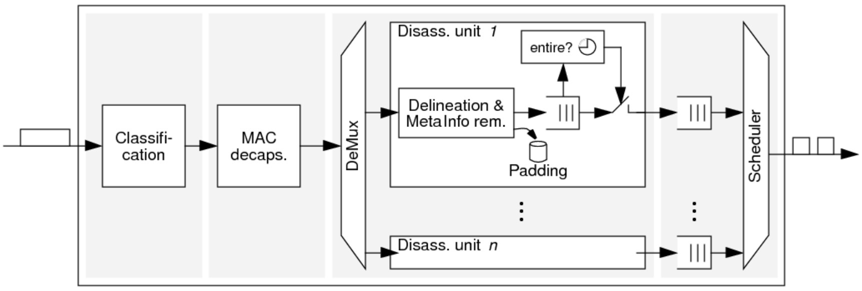

Figure 3: Functional architecture of egress FAU

packets face fragmentation when using maximum frame size for maximum throughput. Further, fixed size container frames use padding in low load situations.

2.2 Assembly procedure

This section classifies the introduced assembly procedure to well-known framing procedures currently applied in carrier networks (e.g., packet over SONET (IETF RFC2615, Generic Framing Procedure (GFP, [10])).

The conventional term framing denotes the representation of logical packets on transport bit streams. These packets arrive randomly with an arbitrary gap in between. The framing procedure of carrier networks usually maps these packets onto a constant bit stream. Besides the logical packets, the framing procedure also maps the gaps between the packets onto the bit stream. Consequently, the carrier bit stream is completely occupied and circuit-switched within the network.

In contrary to this, frame assembly in terms of this paper works differently. First, frame assembly puts packets back-to-back without the intermediate gaps into larger containers. Second, the network forwards these containers individually in a store and forward, packet-switched manner exploiting potential multiplexing gains.

2.3 Application of packet assembly

This section elaborates on the location of the assembly functionality in the network.

Packet assembly is only applicable with aggregated traffic. The packet assembly process always introduces a delay to the assembled packets. As the network performance and application requirements require a limit of this delay, the ingress traffic of a FAU should be large enough to minimize padding. High line rates or multiplexing of smaller line rates implements this requirement. Consequently, this requirement moves the frame assembly functionality away from access towards network core.

The process of frame assembly itself relies on individual packet processing. It suffers from the small packet dilemma the same way as any other network node would do without frame assembly. Hence, frame assembly in core nodes would not save anything. Therefore, the network edge is the potential operational location of a frame assembly node. There, it still requires the highest packet processing capabilities, but the inner core could benefit from a relaxed frame processing rate.

As a result, frame assembly is a core network technology, which is most efficient applicable at network edges with a reasonable bit rate hierarchy from access to core.

2.4 Switching principle

Container frame forwarding within the core requires a routable address, e.g. IP address, or a path identifier to identify a pre-configured label switched path (LSP).

The former has the advantage of stateless and simple core switches but inhibits resource reservation, e.g. bandwidth requirements. The latter requires states within the core switches but enables resource reservation for traffic engineering purposes and quality of service (QoS).

As traffic engineering is a mandatory issue for any new network technology, frame switching follows the LSP principle. LSP maintenance requires a manual or automatic control plane. The most prominent candidate is the Generalized Multi-Protocol Label Switching protocol (GMPLS). In [6] the authors showed, that GMPLS also supports frame switching networks.

3 Frame Assembly Unit

The key element in a frame switched network is the Assembly Edge Node (AEN). It assembles packets to frames

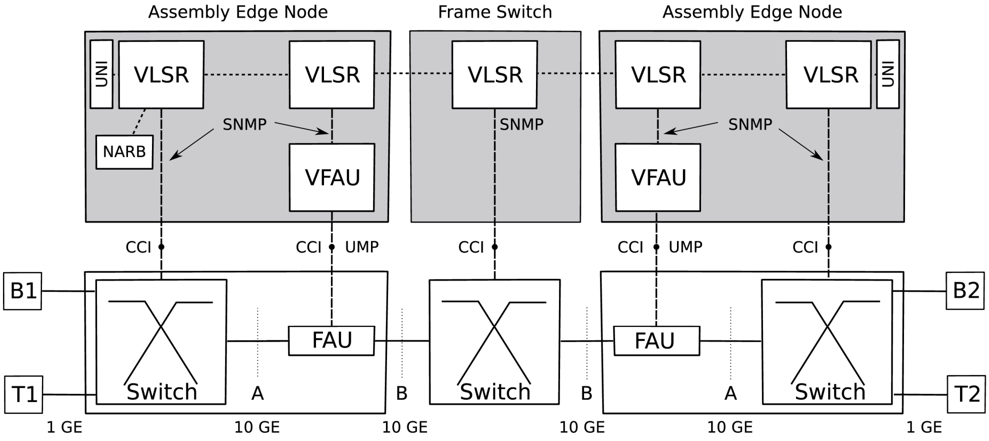

Figure 4: Demonstrator/Testbed

in ingress direction and disassembles frames to packets in egress direction.

3.1 Assembly Edge Node Architecture

The functionality of an AEN consists of the two independent functions: switching and assembly; switching of traffic flows and assembly of aggregated traffic. Therefore, the realization of an AEN shows two options. (1) A hybrid device incorporates both functions. For resource savings, in such a device the switching and the assembly part may share components, e. g. a common packet buffer. Nevertheless, the shared components have to suffice a larger number of requirements and are therefore more complex. (2) A modular approach shows separate devices for these functionalities.

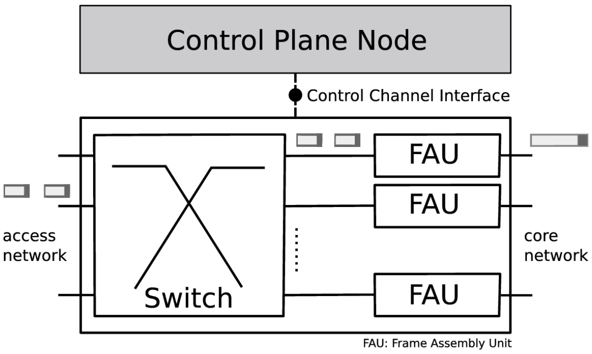

Figure 5: Architecture of assembly edge node

Figure 5 depicts the architecture of such a modular AEN composed of one switch and several Frame Assembly Units (FAU). This approach is highly flexible, as it allows adding ports to an AEN incrementally. We follow the modular approach (2) and focus in the following on the architecture of a FAU. As switches are available for all packet oriented technologies we regard them as solved.

In the following, we show the functional architecture of a FAU.

3.2 Functional Architecture of the FAU

This section gives an overview on the architecture and the functionality of a FAU. We refer to [4] for a detailed description of this device and its implementation.

Figure 2 depicts the functional architecture of the FAU's ingress direction. From left to right, it classifies incoming packets according to destination egress AEN and CoS class and assigns them to a corresponding internal Forwarding Equivalent Class (FEC). The subsequent assembly stage assembles packets of the same FEC to frames (one assembly unit per FEC). Therefore, the FIFO in the corresponding assembly unit collects arriving packet data.

In every assembly unit, the control block monitors the FIFO fill level and triggers a frame generation when exceeding the threshold. A packet arrival into an empty FIFO starts a timer to avoid starvation. Upon timeout, the assembly unit generates a frame irrespective of the amount of available packet data. In case of fixed size frames, the assembly stage fragments packets to fill frames completely. It appends padding if the amount of collected packet data is below the minimum frame size. Before concatenation to one continuous data block, meta information is added to enable packet delineation in egress direction.

The buffering stage stores frames ready for transmission in case of congestion, while the following scheduler handles the frames according to their Class of Service (CoS). The MAC encapsulation stage finalizes the frame for transmission by adding headers and trailers.

Figure 3 shows the functional architecture of the egress direction of a generic FAU. From left to right, it classifies incoming frames according to ingress AEN and CoS and assigns them to an FEC. The MAC decapsulation stage removes the frame header and trailer.

Every unit in the disassembly stage delineates packets with help of the meta information added during assembly. It also drops the meta information as well as padding. In case of fixed size frames, the last data in a frame may be a packet fragment. The FIFO queue stores packets and packet fragments. The control block monitors the FIFO. If it contains an entire packet, it triggers its forwarding. Similar to ingress direction a buffering stage and a scheduling stage take care of packet transmission according to their CoS.

4 Demonstrator

This section introduces briefly our testbed. The reader finds a more detailed description of the whole scenario in [6].

Figure 4 depicts our testbed for the complete FS network to quantify the impact of packet assembly on the traffic characteristics. In the lower part it depicts the data plane while in the upper part it shows the control plane interconnection.

Our testbed consists of three AENs (Figure 4 shows only two AENs because of space limitations) and one core switch representing the FS core network.

In the data plane, we use Ethernet technology (including virtual local area networks extension of IEEE 802.1Q) in the core and in all access networks. In the access we deploy 1 Gigabit Ethernet and in the core 10 Gigabit Ethernet. Standard Ethernet jumbo frames of 9 kByte build our container frame within the core. We interconnect the access networks transparently on layer 2 by transporting all Ethernet packets through the FS network.

Each AEN consists of an aggregation switch and a FAU (cf. Figure 5). The FAU connects to one of the 10 GE uplink ports of the switch. The switch classifies incoming traffic from access side on a per port basis and switches it to the appropriate outgoing port connected to the FAU. The classification process applies the Virtual LAN concept at reference point A in Figure 4. Application of VLAN simplifies classification within the FAU and decouples FEC from the attached clients' MAC addresses. The FAU classifies incoming packets based on their VLAN header, assembles packets belonging to one FEC into Ethernet jumbo frames and forwards them to the core network.

As FS networks show the connection oriented communication principle, while Ethernet is connection less, we emulate a connection in the core. Therefore, one VLAN per bidirectional connection reflects the end-to-end connectivity between AEN (reference point B in Figure 4). This also enables class of service differentiation in the core switch by using the VLAN-priority field in the jumbo frames VLAN header.

We realized the bidirectional FAU on an evaluation board with two Xilinx Virtex-4 FX100 FPGAs (one per direction), two optical 10 Gigabit Ethernet interfaces for data plane and a 1 Gigabit Ethernet interface for CP connection. We designed the FAU prototype in VHDL supporting seven FECs simultaneously per direction. The authors provide in [4] an in depth description of the FAU prototype architecture and implementation.

We realize the control plane for path maintenance with the GMPLS control plane implementation of the DRAGON project [7]. This control plane implements a virtual router (VLSR) for control plane message processing and a user network interface (UNI) for path requests and monitoring. Besides this, it includes a path computation element (NARB) for constraint based path calculation.

We extended the control channel interface (CCI) between the control plane nodes and the data plane nodes by a virtual FAU (VFAU) by a protocol gateway between the simple network management protocol (SNMP) and the protocol for configuring the FAU (UMP). We further modified to UNI to signal the assembly timer value to the ingress and egress node. Due to the extension, the UNI also includes the information on the class of service.

5 Performance evaluation

This section provides a performance evaluation on the packet assembly functionality. It first estimates the performance gain with respect to the reduced header processing rate. Second, it calculates the minimum required load for packet assembly in normal operation. As a last topic, it estimates the impact of packet assembly on a downstream buffer device.

5.1 Performance gain

This section quantifies the performance gain of packet assembly. We estimate the performance gain with the following assumptions:

- Packet size distribution of packets to be assembled range between 64 and 1520 Byte in Ethernet networks,
- Container frame format: $L_c = 9\,kByte$ Ethernet Jumbo frames, constant frame size
- FAU ingress/egress link rate: $r = 10\,Gbps$
- Minimum packet size in Ethernet networks: $\underline{L}_p = 64\,Byte$
- Ethernet interframing gap and preamble: $I = 20\,Byte$

The minimum packet size requires the maximum packet processing rate $\overline{r}_p$ in the node. This rate evaluates to

$\overline{r}_p = \dfrac{r}{\underline{L}_p + I} = 14.9\,Mpps$ (mega packet per sec.).

The same traffic assembled in Jumbo frames requires only

$\overline{r}_c = \dfrac{r}{L_c + I} = 139\,kpps$ (kilo packet per sec.)

The reduction in the required header processing capability is more than factor 100, although the Jumbo frame exceeds the maximum payload packet size only by factor 6. The benefit of frame assembly is even more impressive in the case of 100 Gbps links: Without frame assembly, the required header processing capability would be 149 Mpps per link. This is challenging to implement, since it is in range of the clock rate of the underlying ASIC technology.

5.2 Minimum nominal load

The frame assembly process usually implements a timer, a size based threshold or a combination of both. If the assembly process implements the second option only, it suffers from the risk of packet starvation in partially filled frames that do not complete due to missing follow-up traffic. Therefore, we implement a combination of timer

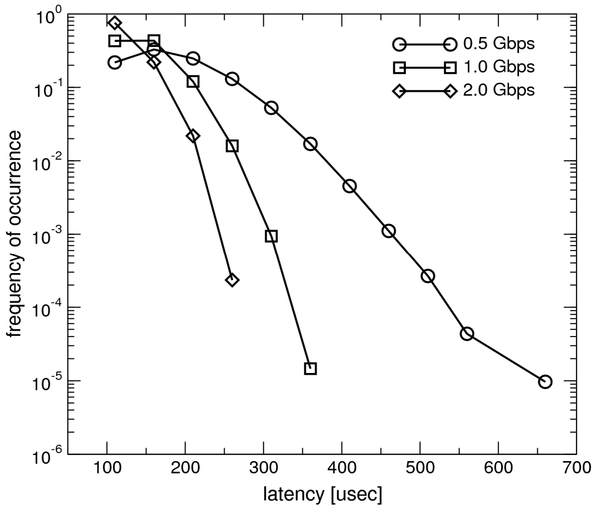

Figure 6: Packet latency due to frame assembly

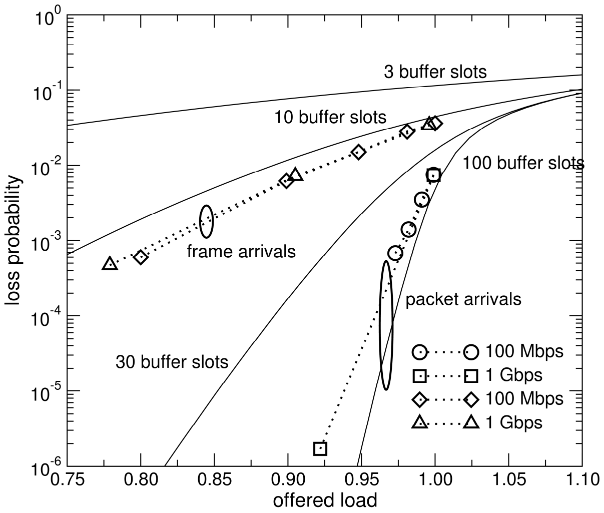

Figure 7: Buffer performance degradation

based and sized based threshold. If the assembly process reaches the frame size limit or the timeout value, it releases a frame. Consequently, we implement size based threshold of L_c with an additional timer T .

The parameterization of the timer depends on the expected load situation of the traffic. If the amount of traffic arriving within the time T is smaller than the frame size L_c , the resulting frames waste capacity as they carry padding. As an alternative, reducing L_c reduces also the amount of padding, but does not reduce the frame rate (and the required processing rate) accordingly.

Both effects occur in low load situations and are as such not exceptional critical. We limit ourselves to the question, which minimum nominal load per forwarding equivalent class is required to limit the timer triggered frame delivery to exceptional cases.

We assume following benchmark parameters:

- timer value $T = 1ms$
- Jumbo frame size $L_c = 9000\,Byte$

A constant traffic flow of $\frac{L_c}{T} = 72\,Mbps$ would fill the frames just in time. A fluctuating traffic flow with the same mean rate would also release some partially filled frames. Anyway, as a rule of thumb, we can state that frame switched forwarding equivalence classes should show not much less than 100 *Mbps* load. Lower load is possible but inefficient.

5.3 Jitter and latency

This section classifies the jitter and packet latency due to packet assembly.

The assembly time of a particular frame depends on the actually incoming traffic. Its maximum equals the assembly timer, while the minimum depends on the load.

Furthermore, the frame assembly process delays the assembled packets. The packets waiting time depends on the assembly timer and the arrival of subsequent packets, which is in general unpredictable. This random waiting time represents an additional jitter. Jitter in the range of milliseconds is a commonly accepted effect in packet forwarding networks. This jitter occurs only once at ingress to the core network and does not accumulated within the core network.

Because of the store and forward mechanism, every core node adds the transmission delay of the larger frames to the individual packets delay. The delivery of a 9 kByte Jumbo frame at 10 Gbps takes less than 8 µs, which is several orders of magnitude below other jitter contributions and can therefore be neglected.

For experimental confirmation of our assumptions, we used the setup of Figure 4. We investigated the impact of frame assembly on a test flow in the presence of random background flows. The background flows represent the aggregated traffic of many independent users, while the test flow represents the particular traffic of one dedicated user or application. In Figure 4, we consider the latency of test flow from node T1 to T2. The background traffic originates at B1 and terminates at B2. Both flows share the same forwarding equivalent class and thus use the same resources in both FAUs.

The background traffic shows an average rate of 0.5, 1, or 2 Gbps, respectively. We compose the background traffic by an overlay of randomly arriving 10 Mbps application streams and use the same traffic model as in [3]. T1 injects the test flow in the frame switching network at rate of 10 Mbps composed by 500 Byte packets showing a constant inter-arrival time. At T2, we record the latency of the packets after traversal of the testbed. The assembly strategy applies a pure size-based threshold of 9 kByte without any timer. The threshold reflects the maximum quasi-standard transfer unit of Ethernet.

Figure 6 shows the experimental probability distribution (histogram method) of the latency for different background load levels. After removal of the constant propagation delay, the maximum of the distribution shifts reciprocally with the traffic load. This fits to the load dependent waiting time during frame assembly. Furthermore, in low load situations, the waiting time shows a rather long tail, but any timer-based assembly limits the maximum additional delay.

5.4 Downstream buffer performance degradation

A more subtle problem arises from the clustering of packets at output of a frame switched network. The packet delivery process changes the random distribution of packets into clusters of packets. The original interarrival time between packets vanishes (cf. Figure 1). The relative shift of packets on the time scale corresponds to the jitter as explained above. Successive frame arrivals and frame disassembly results in successive bursts of packets.

Some of these cases may increase the packet loss due to degradation of the buffer performance in downstream packet switches. This is especially critical, since downstream packet switches are out of scope of a frame switched network. In opposite to FS network itself, we cannot expect any additional adaptation there.

The following analysis relies on the theory of the time scales in packet traffic as explained in [3] and on the investigation of buffer operation in presence of application streams in [2].

First, we consider the time scale of the packet clustering. The traffic volume between two consecutive frames remains the same before and after the assembly. Additionally, the assembly timer as well as the packet traffic loads in combination with the size limitation limits this traffic volume. The packet position at the egress FAU shifts within the same interval. The assembly process does not affect any dimensioning considerations at larger time scales, e.g. time scale of application buffer holding times or time scale of application stream duration.

Second, we consider the downstream buffer device after the egress FAU. Therefore, we consider the traffic characteristic of the assembled frames arriving at the egress FAU. We assume a size based assembly process, which is equivalent with the minimum required load assumption of section 5.2. Here, we distinguish two cases, depicted in Figure 8 and Figure 9.

The first scenario considers a one-to-one communication between two FAU, while the second scenario considers a many to one FAU. In both scenarios, many independent sources feed the FAU on the network ingress side. Consequently, the arrival process of the packets equals a Poisson process (reference point A).

In the first scenario (Figure 8), the interarrival time of the assembled frames (reference point B) follows an Erlang distribution depending on the packet load and packet size distribution especially for maximum size packets [8].

As long as the original packet traffic does not overload the ingress device, the buffer filling increases at most by the size of one frame. The process of frame assembly at ingress separates the frames at distances that prevent from more than one additional frame content in the buffer on top of the normal packet load. With a frame size of 9 kByte and typical packet buffers of 130 kByte this additional load is comparably low.

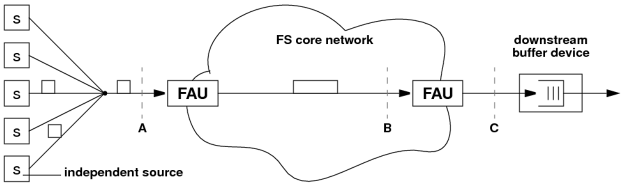

Figure 8: Single source scenario

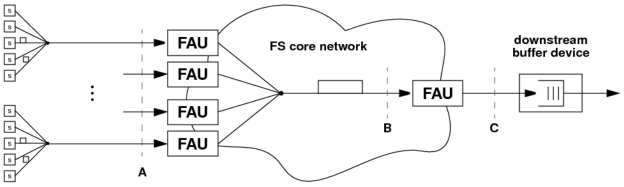

Figure 9: Multiple sources scenario

The situation is different in the second scenario depicted in Figure 9. Here, the packet clusters arrive from many different and independent frame switched paths (FEC). For a large number of sources the traffic hitting the egress FAU converges to a Poisson process (reference point B).

In this scenario, the traffic of each FEC individually summates to the mean traffic load occurring at the downstream buffer device (reference point C). Consequently, the traffic flow of each FEC in the second scenario is smaller than in the first scenario. Although, for a worst case estimation, we assume the frames completely filled.

The theoretical buffer performance depends on the number of *buffer slots*, where a buffer slot is the amount of memory that is able to hold one of the randomly and independently arriving traffic portions. In Ethernet, this portion is at maximum 1520 Byte, one packet.

Assembled traffic uses frames of 9 kByte, which blows off in a packet burst after the egress FAU (reference point C). A real buffer does not care of the difference between bursts of individual smaller packets or equally sized large container frames. It shows a fixed amount of memory, in our example device, a 10 GEth switch, it is in range of 130 kByte. For individual Ethernet packets (1520 B) this corresponds to 85 buffer slots, for 9 kByte bursts of packets it has only 14 buffer slots!

We verified the effect in an experiment. Figure 7 recalls the theoretical buffer performance curves from [3], which are set into relation with the experimental results. In the experiment, we counted packet losses in a 10 GEth switch in front of a 100 Mbps and of a 1 Gbps downlink.

The reference arrival process was Poisson at packet basis. In the other case, we used bursts of 6 Ethernet packets showing also Poisson arrival characteristics. The observed degradation of buffer performance fits well to the theoretically predicted reduction from 85 to 14 buffer slots.

The practical relevance of the results is ambiguous. The second scenario is possible, but it is very unlikely.

First, the traffic from each FEC is subject to lower load limits (cf. section 5.2). Large numbers of such flows would create huge amounts of traffic at the minimum. Thus, the affected downstream device is still close to the network core, but not to the end user application.

Second, the buffer performance at packet level according to Figure 7 is only the prerequisite of the application stream multiplex (cf. [3]) with anyway much worse overall performance figures.

6 Future Work

The introduction of the traffic dependent jitter but also the buffer degradation due to packet clustering is well investigated and its impact is estimated to be comparably low in practically relevant operating conditions. Anyway, for better confidence of the technology, it is worth to investigate appropriate measures to avoid the undesired side effects.

The packet release process at the egress node may avoid burstification by two simple mechanisms. If the assembly process records the assembly time within the container frame, the packet forwarding may spread the packets uniformly across this time interval. If additionally the original inter-arrival time of the packets is recorded, packets may be released according with their original interarrival time.

Ultimately, as further refinement of the above, the total waiting time for frame assembly and de-assembly could be fixed at a value corresponding to the delivery timer. Any packet (including the first in frame) is time stamped with its waiting time in the ingress FAU. After frame arrival at egress, the packets are released according to the respective waiting time reminder. It is expected that in this case even the presence of frame assembly in a network domain remains almost undetectable to the outside world.

7 Conclusion

In this paper, we provided a detailed description on packet assembly at the network edge to reduce the overall header processing load in a packet based core network. Therefore, we designed and implemented an assembly edge node releasing Ethernet Jumbo frames of 9 kByte carrying multiple packets.

In a demonstrator scenario, we showed a working setup of a prototypical core network working on 10 Gbps. The demonstrator consists of assembly nodes as well as a high performance switch for network core emulation.

We qualified the doubts on packet assembly regarding the change of the traffic characteristic. We provided a detailed analysis and showed that our results fit the outcome from our measurements in the network. We concluded that packet assembly at the network edge has got an impact on the traffic characteristics, but this impact in negligible compared to other sources of delay in a network. In normal network operation, it is expected that frame assembly will not even be recognized by any application as its impact is so low.

Acknowledgement

This work has been funded in parts by the German Federal Ministry of Education and Research (BMBF Grant FLINTSTONE 01BP556)

Literature

[1] F. Masetti, et al., Design and implementation of a multi-terabit optical burst/packet router prototype, in Optical Fiber Communication Conference and Exhibit, Mar 2002

[2] W. Lautenschläger, Equivalence Conditions of Buffered and Bufferless Network Architecture, 9. ITG Fachtagung Photonische Netze, Leipzig, 2008

[3] W. Lautenschläger. Bandwidth dimensionierung in stochastic packet networks. In Proceedings of the 8. ITG Symposium on Photonic Networks, Leipzig, May 2007.

[4] A. Mutter et al. A generic 10 Gbps assembly edge node and testbed for frame switching networks. In TridentCom, accepted for publication, 2009.

[5] R. Sinha et al. Internet packet size distributions: Some observations. Technical Report ISI-TR-2007-643, USC/Information Sciences Institute, May 2007.

[6] A. Mutter et al., Design and Performance Evaluation of a Frame Switching Network, submitted to HPSR conference 2009

[7] T. Lehman et al. Dragon: a framework for service provisioning in heterogeneous grid networks. Communications Magazine, IEEE, 44(3):84–90, March 2006.

[8] M. de Vega Rodrigo and J. Goetz. An analytical study of optical burst switching aggregation strategies. In Proceedings of WOBS 2004, San Jose, October 2004.

[9] G. Kornaros et al., Architecture and implementation of a frame aggregation unit for optical frame-based switching, International Conference on FPL 2008.

[10] E. Hernandez-Valencia et al. The Generic Framing Procedure (GFP): an overview. IEEE Communications Magazine, 40(5):63–71, May 2002.

[11] G. Hu, K. Dolzer, and C. M. Gauger. Does burst assembly really reduce self-similarity? In Proceedings of the Optical Fiber Communication Conference (OFC), Atlanta, March 2003.