Performance Tradeoffs for Header Compression in MPLS Networks

Michael Menth, Oliver Rose
Institute of Computer Science, University of Würzburg, Am Hubland, D-97074 Würzburg, Germany
phone: (+49) 931-8886644, fax: (+49) 931-8886632, email:{menth|rose}@informatik.uni-wuerzburg.de

Abstract

The transmission of low-bitrate real-time traffic is inefficient in IP networks with regard to bandwidth utilization by user data due to the high protocol overhead. Header compression mitigates this shortcoming but it is only applicable on a link by link basis. In this paper we propose the combination of header compression techniques and MPLS technology to enable RTP/UDP/IP header compression over several IP hops. We present numerical results that illustrate the performance gain.

1 Introduction

A large part of today's toll-quality real-time data consists of low-bitrate traffic such as voice, video, and circuit switched data. They originate e.g. in the terrestrial radio access networks of wireless communication systems like GSM or UMTS and are carried over low-bandwidth links. When low-bitrate real-time data are transported over IP networks, the RTP/UDP/IP header suite results in a large overhead that decreases the bandwidth utilization by user data drastically.

Header compression is a means to overcome this inefficiency at the expense of increased processing complexity in the routers. The headers of packets sharing the same context are mapped to a small connection identifier (CID). The payload is carried over the link together with the CID and some additional information instead of the regular header. Based on that information the original header is restored at the receiver. Because the IP information is no longer available in the packet, IP routing does not work anymore. As a consequence, IP header compression is only applicable between two adjacent IP hops.

MPLS is an emerging technology for traffic engineering in IP networks and it is likely to be deployed in future communication systems. A so-called label switched path (LSP) in MPLS establishes a data pipe over several hops between two non-adjacent IP nodes. Packets are forwarded only according to the small MPLS label and the information of the encapsulated IP header is not required. Therefore, header compression can be applied between the first and last label switching router (LSR) of an LSP, and only the end systems need to implement header compression. The intermediate LSRs just forward the data transparently. Thus, with MPLS several hops can profit from a single header compression which makes its use more attractive. In turn, the benefits from header compression encourage the deployment of MPLS.

This work is structured as follows. In Section 2 we consider the transport of low-bitrate real-time data. We point out their quality of service (QoS) requirements and present an admission control (AC) formula for periodic and homogeneous flows. The transport of low-bitrate real-time data over ATM and IP networks is inefficient, so header compression techniques are used to overcome this weakness. In Section 3 we briefly present the basics of MPLS and propose to combine RTP/UDP/IP header compression with MPLS technology. The numerical results in Section 4 show the influence of header compression in MPLS networks for low-bitrate real-time traffic and illustrate some performance tradeoffs. Section 5 summarizes our findings.

2 Transport of Low-Bitrate Real-Time Traffic

In this section we describe a probabilistic QoS criterion for real-time traffic. We propose an AC formula for periodic and homogeneous traffic sources which allows for high link utilization, especially on expensive low-bandwidth links. Low-bitrate real-time traffic leads to transport inefficiencies with current network protocols. This deficiency can be overcome by various header compression and tunneling mechanisms.

2.1 QoS Requirements for Real-Time Traffic

The major traffic volume of today's real-time data is due to telephony. In the future, video and other time critical applications also require circuit emulation for their data transport. In contrast to web traffic, real-time data yield higher revenues but they must be forwarded with low loss and delay. The waiting time is a suitable measure to define a QoS criterion for real-time traffic. A packet has a certain de-

lay budget DB, i.e. it may wait for DB time in the buffer of each node for its transmission due to queuing. A hard bound for the waiting time is a very conservative approach which leads to low bandwidth utilization of the link. Thus, it is better to soften this requirement to a probabilistic approach. The probability for the waiting time to exceed the DB must be smaller than p or in other words: The $(1-p)$ quantile of the waiting time distribution must be smaller than DB. **Figure 1** illustrates this concept. For interactive real-time traffic we use the parameter set $p = 10^{-4}$ and $DB = 5$ ms. Another aspect is the packet loss probability. However, this is not really an issue because the delay constraint requires short queues so that reasonably designed buffers can prevent packet loss.

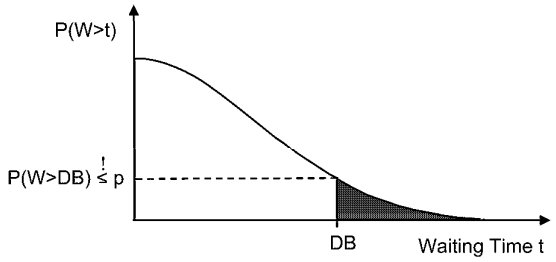

Figure 1: The quantile of the waiting time defines a delay criterion for QoS.

2.2 Admission Control for Periodic Real-Time Traffic

In the terrestrial radio access network of GSM or UMTS, leased low-bandwidth links are used to interconnect the users with the core network. This is a costly solution and, therefore, it is desirable to use the rented capacity efficiently to avoid unnecessary expenses. A high network utilization is desired but the real-time constraints of the data must not be violated. To solve this conflict, the dimensioning of the network capacity is rather tight and AC for new flows prevents congestion on a link.

An efficient AC takes advantage of the flow characteristics. The low end-to-end delay requirement influences also the traffic generation process. The time to assemble a data packet by the application is kept short, therefore, the user payload size of a data packet is small. Hence, real-time traffic is often characterized by the periodic production of small samples.

We consider the transport of homogeneous flows over a low-bandwidth link. The periodic packet inter-arrival time t_{iat} is the same for all flows. In addition, we assume that all the packets have the same size and, as a consequence, the same service time t_{st}. This leads to a deterministic queuing system with period t_{iat}. Therefore, it is obvious that the buffer is empty at least once in a period if the load of the system is smaller than 1. The queuing behavior is fully determined by the arrival pattern of the joint packet arrival process within a single inter-arrival time interval of time t_{iat}. To obtain the waiting time distribution, this system can be modeled by an $N * D/D/1$ queue which denotes the multiplexing of N identical flows with constant packet inter-arrival and service time. An analysis of that system is found in [1]. By essentially randomizing all possible arrival patterns the waiting time distribution for packets in this system is obtained. An adaptation of the approximated formula to our context yields:

$$P(W > DB) = \exp\left(\frac{-2 \cdot DB}{t_{st}} \cdot \left(\frac{DB}{N \cdot t_{st}} + 1 - \frac{N \cdot t_{st}}{t_{iat}}\right)\right) \quad (1)$$

2.3 Header Compression for Low-Bitrate Real-Time Traffic

Now, we consider transport alternatives for low-bitrate real-time traffic in the Internet. First, we present the conventional RTP tunneling approach which is inefficient regarding the network utilization by user data. Header compression can be done on a link-by-link basis to overcome this weakness. Finally, packets with compressed headers may be multiplexed into a single RTP/UDP/IP packet and tunneled over several hops to a common destination.

RTP Tunneling. For real-time transport in the Internet, the Real-Time Transport Protocol (RTP) [2] is used. The RTP header comprises 12 bytes. It carries a synchronization and a contribution source identifier (SSRC, CSRC), a timestamp, a sequence number, and some flags. It provides information to resynchronize different streams within an application. The port numbers of the communicating applications at the sender and the receiver are qualified in the User Datagram Protocol (UDP) [3]. The corresponding header is 8 bytes long. Moreover, it records the length of the UDP packet and protects it with a checksum against errors. The IP header carries the addresses of the source and the destination machine. In IP version 4 (IPv4) [4], the header comprises 20 bytes, thereof 4 octets for each IP address. The address space of IPv4 is likely to run out in the future as soon as each end device in an all IP architecture requires an own IP address. Therefore, the new IP version 6 (IPv6) [5] spends 16 octets per IP address and has a header size of 40 bytes. We use this alternative in our investigation. The RTP/UDP/IP protocol suite amounts then to 60 bytes while the average voice packet size is not even 30 bytes large. This results in a protocol overhead of more than 200%.

RTP/UDP/IP Header Compression. When real-time data are exchanged, most of the protocol fields show the following behavior: Either, they do not change at all during the session's lifetime (SSRC, CSRC, ...) or they change steadily by a constant increment (timestamp, sequence number, ...). Header compression relies exactly on this

observation [6, 7, 8]. The constant and steadily changing data compose the session context. To establish the context in the compressor and the decompressor at both sides of a point-to-point link, a full header is exchanged together with a connection identifier (CID) that is unique for a session. Then, only the compressed headers containing the CID and some auxiliary information are transmitted together with the actual payload. The full header is reconstructed from the CID and the context at the decompressor. To make the system more robust against packet loss, full headers are transmitted with regular distance. But under low packet loss conditions this is done only once within 256 frames, which is almost negligible from a performance point of view. Without the information in the IP header, packets can not be routed through an IP network. Therefore, RTP/UDP/IP header compression works only on a link by link basis.

RTP Multiplexing. A means to overcome this disadvantage is to apply header compression between two arbitrarily distant peers and to transport several of the compressed packets in a single IP packet to the destination. We refer to this by RTP multiplexing. **Figure 2** shows the advantage of RTP multiplexing in an IP network over pure header compression. The resulting packet has an ordinary IP header and can be carried transparently through the Internet.

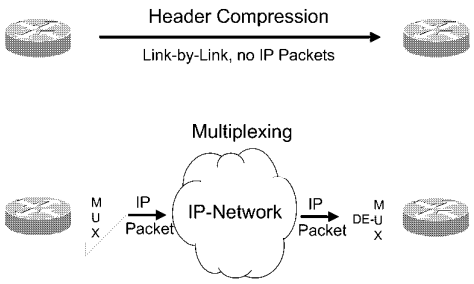

Figure 2: Header compression versus multiplexing.

The present Internet draft for tunneling multiplexed compressed RTP packets [9] is based on an enhanced RTP/UDP/IP header compression algorithm [7, 10] and a layer independent multiplexing protocol [11, 12]. A further compression step reduces the size of the multiplexing layer [13]. This kind of multiplexing requires to collect several samples with a compressed header for the resulting IP packet. The multiplexing interval must be limited to avoid excessive delays for the transported data. This yields an interesting performance behavior that has been studied in [14, 15, 16]. The ATM Adaptation Layer Type 2 (AAL2) [17, 18] is basically the same approach for ATM systems and exhibits similar tradeoffs [19, 20].

3 Header Compression in MPLS Networks

In recent years, the IntServ [21] and DiffServ [22] have been developed for the transport of real-time data in the Internet. However, they lack powerful traffic engineering mechanisms to perform route pinning, load sharing, fast rerouting, and others. This can be achieved by Multiprotocol Label Switching (MPLS) [23]. MPLS establishes virtual tunnels using only a small label for packet forwarding. Therefore, it is a suitable candidate for a tunneling technology with little header overhead. It may be used in conjunction with IP networks and is suited for carrying real-time data with compressed header information.

3.1 Some Basics about MPLS

MPLS is a mechanism to allow packet switching instead of routing over any network layer protocol [23]. The first label switching router (LSR) of a label switched path (LSP) equips the IP packet with a label of 4 bytes and sends it to the next LSR. The LSRs classify a packet according to its incoming interface and to its label. Based on this information, label swapping is performed and the packet is forwarded to the particular outgoing interface. The last LSR only removes the label from the IP packet header. In practice, modern routers are capable to both IP routing and MPLS label switching.
An LSP implements the connection concept. Therefore, MPLS is often viewed as modified version of the Asynchronous Transfer Mode (ATM) with variable cell size. But there is a profound difference: ATM enables with its virtual connection and virtual path concept a two-fold aggregation while MPLS allows for many-fold aggregation using multiple label stacking, i.e. an LSP may be transported over other LSPs. This feature may be exploited for scalable network structures [24].

3.2 Header Compression with MPLS

As we have seen in the previous section, the transmission of low-bitrate real-time data is inefficient if the size of the header leads to a small network utilization by user data. Header compression mitigates the problem at the expense of losing the IP header such that the packet can only be transported over a single link using another special layer 2 transport protocol like PPP.
MPLS can be used for tunneling packets with compressed RTP/UDP/IP headers efficiently because its labels are small and MPLS is recommended for traffic engineering purposes. The compressor and the decompressor are established at the ingress and egress of an LSP. The LSP is a virtual tunnel and carries packets with compressed headers transparently over several IP hops. This idea has been worked out in [25, 26]. Both the RTP/UDP/IP header suite can be compressed and a part of the label stack if multi-

ple label stacking is used. According to the draft, these headers can be reduced to 2 or 4 bytes including the CID.

On the one hand, this increases the network utilization by user data and decreases the operational costs but on the other hand the introduction of header compression is expensive. To implement compression schemes at high speed, special purpose hardware is designed. This is costly since the hardware must be designed in a scalable manner [27] and compressors and decompressors must work reliably even in case of packet loss and other network failures.

Therefore, it is important to estimate the performance gain attained by header compression in order to balance the expenses against the expected benefits.

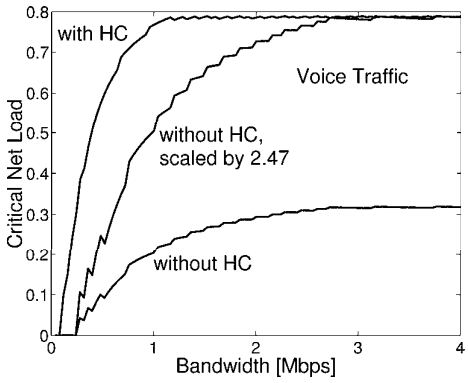

Figure 3: The performance gain by header compression (HC) is more than the reduced mean rate.

4 Numerical Results

A given number of flows N produces a certain offered load for a link. We distinguish between the net load and the gross load. The first takes only the user payload into account, the second one also the protocol headers. We use Equation (1) to determine the maximum number of connections that can be transported over a link without violating the delay constraint. This leads to the critical load. We are specifically interested in the critical net load which is the maximum network utilization by user data.

In this section, we compare the critical net load for voice traffic with and without header compression. We illustrate that packetization of 64 kbit/s circuit switched data (CSD) has a large influence on the critical net load of the link. This leads to the conclusion that the positive impact from header compression on the critical net load is two-fold: the mean rate and the burstiness of the flows are reduced.

4.1 Performance Gain by Header Compression

We consider voice streams coded with 12 kbit/s such that every 20 ms a frame of 30 bytes is transmitted. The conventional RTP/UDP/IP/MPLS protocol suite yields a header size of 64 bytes (= 94 bytes packet size) while header compression allows to work with a compressed RTP/UDP/IP header of 4 bytes. Such a packet is tunneled through an LSP equipped with a label of 4 bytes such that the resulting packet size is 38 bytes large.

Voice transmission without header compression is clearly inferior to the header compression alternative. It is obvious that the gross rate of a stream is reduced from 37.6 kbit/s to 15.2 kbit/s, hence, we expect an increase of network utilization by user data of 147%. However, as we can see in **Figure 3**, the actual growth is even larger, in particular for low-bandwidth links. This fact is due to the reduced burstiness of the traffic. This will be shown in the following.

4.2 Influence of Burstiness and Protocol Overhead

The phenomenon observed above can be illustrated more clearly by a 64 kbit/s CSD emulation service in IP networks. One packet is assembled per transmission time interval (TTI) and sent over the network. For a TTI=20 ms, the resulting user payload size is 160 bytes. We investigate the performance of the system depending on the duration of TTI which is proportional to the resulting payload size. Again, the normal header size is 64 bytes and the size of a compressed header tunneled by an LSP is 8 bytes.

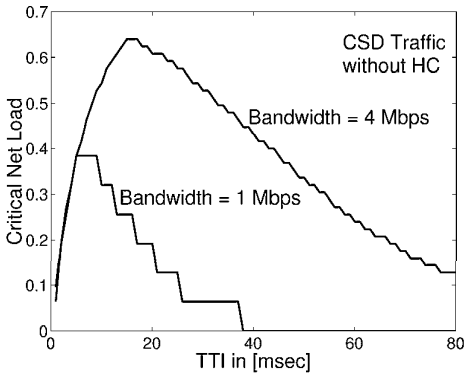

Figure 4: The impact of the link bandwidth on the optimum TTI value.

Figure 4 shows that an optimum TTI exists for the transmission of CSD. This can be explained as follows. The user payload size in a packet is proportional to the TTI while the header size is constant. Hence, large TTIs yield a large proportion of user payload in the transported data volume. Conversely, large TTIs lead to long packets that have an adverse impact on the maximum gross load on the link because of the increased burstiness of the traffic.

Hence, there must be an optimum TTI that maximizes the critical net load. The graph contrasts the critical load for links with a bandwidth of 1 and 4 Mbit/s and it is evident that the optimum TTI depends on the link bandwidth. A similar behavior is observed for a 4 Mbit/s link with and without header compression. Header compression clearly increases the critical net load (cf. **Figure 5**) but it does not displace the optimum value for TTI.

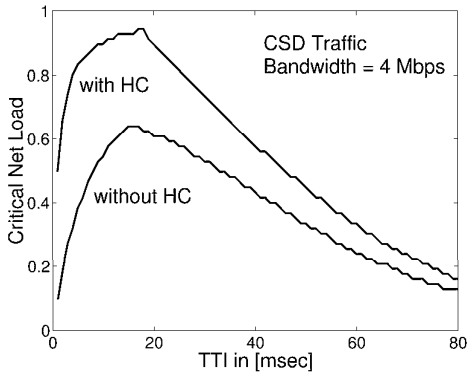

Figure 5: The impact of header compression (HC) on the optimum TTI value.

The experiments with CSD traffic prove that increased burstiness alone (even with slightly decreased mean rate) has already an adverse impact on the critical gross load. Its effect depends on the bandwidth. The benefit of header compression is not only the smaller mean rate but also the reduced burstiness of the streams. The first one contributes to a higher user data proportion in the gross data and the second one allows a larger critical gross load in the system. Thus, the benefit of header compression is larger than expected intuitively.

5 Conclusion

In this work we proposed to combine RTP/UDP/IP header compression with MPLS technology for the efficient transmission of low-bitrate real-time traffic. An adaptation of existing header compression techniques to MPLS allows the application of that concept to LSPs, i.e. more than one IP hop can benefit from a single header compression. Header compression reduces the gross rate of low-bitrate streams such that the transmission capacity of networks increases for voice traffic by almost 150%. For the transport of circuit switched data services or other multimedia applications it is important to choose a suitable packet size to maximize the network performance. This also motivates the positive impact of reduced data burstiness in the presence of header compression. It leads to a considerable and more than intuitively expected performance gain on low-bandwidth access links. In addition to traffic engineering capabilities, this finding is another reason to use MPLS in future networks.

References

[1] James Roberts, Ugo Mocci, and Jorma Virtamo, *Broadband Network Teletraffic - Final Report of Action COST 242*, Springer, Berlin, Heidelberg, 1996.

[2] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RFC1889: RTP - A Transport Protocol for Real-Time Applications," ftp://ftp.isi.edu/in-notes/rfc1889.txt, Jan. 1996.

[3] J. Postel, "RFC768: User Datagram Protocol," http://www.ietf.org/rfc/rfc0768.txt, Sep. 1980.

[4] Jon Postel, "RFC791: Internet Protocol," http://www.ietf.org/rfc/rfc0791.txt, Aug. 1981.

[5] Stephen Deering and Robert Hinden, "RFC2460: Internet Protocol Version 6 (IPv6) Specification," ftp://ftp.isi.edu/in-notes/rfc2460.txt, Dec. 1998.

[6] Michael Degermark, Bjorn Norgren, and Stephen Pink, "RFC2507: IP Header Compression," ftp://ftp.isi.edu/in-notes/rfc2507.txt, Feb. 1999.

[7] Stephen L. Casner and Van Jacobson, "RFC2508: Compressing IP/UDP/RTP Headers for Low-Speed Serial Links," ftp://ftp.isi.edu/in-notes/rfc2508.txt, Feb. 1999.

[8] Mathias Engan, Stephen L. Casner, and Carsten Bormann, "RFC2509: IP Header Compression over PPP," ftp://ftp.isi.edu/in-notes/rfc2509.txt, Feb. 1999.

[9] Bruce Thompson, Tmima Koren, and Dan Wing, "Tunneling Multiplexed Compressed RTP (TCRTP)," http://www.ietf.org/internet-drafts/draft-ietf-avt-tcrtp-06.txt, Feb. 2002.

[10] Stephen Casner, Van Jacobson, Tmima Koren, Bruce Thompson, Dan Wing, Patrick Ruddy, Alex Tweedly, and John Geevarghese, "Compressing IP/UDP/RTP Headers on Links with High Delay, Packet Loss and Reordering," http://www.ietf.org/internet-drafts/draft-ietf-avt-crtp-enhance-03.txt, Feb. 2002.

[11] Gurdeep Singh Pall, Bill Palter, Allan Rubens, W. Mark Townsley, Andrew J. Valencia, and Glen Zorn, "RFC2661: Layer Two Tunneling Protocol L2TP," ftp://ftp.isi.edu/in-notes/rfc2661.txt, Aug. 1999.

[12] Rajesh Pazhyannur, Irfan Ali, and Craig Fox, "RFC3153: PPP Multiplexing," http://www.ietf.org/rfc/rfc3153.txt, Aug. 2001.

[13] Andrew J. Valencia, "L2TP Header Compression (L2TPHC)," http://www.ietf.org/internet-drafts/draft-ietf-l2tpext-l2tphc-04.txt, Oct. 2001.

[14] Michael Menth, "Carrying Wireless Traffic in UMTS over IP Using Realtime Transfer Protocol Multiplexing," in *12th ITC Specialist Seminar*, Lillehammer, Norway, March 2000, pp. 13 – 25.

[15] Michael Menth, "The Performance of Multiplexing Voice and Circuit Switched Data in UMTS over IP Networks," in *Protocols for Multimedia Systems (PROMS2000)*, Cracow, Poland, Oct. 2000, pp. 312 – 321.

[16] Michael Menth, "Analytical Performance Evaluation of Low-Bitrate Real-Time Traffic Multiplexing in UMTS over IP Networks," *Journal of Interconnection Networks*, vol. 2, no. 1, pp. 147–174, 2001.

[17] ITU-T, "I.366.1 Segmentation and Reassembly: Service Specific Convergence Sublayer for the AAL Type 2," June 1998.

[18] ITU-T, "I.366.2 Draft Recommendation: AAL Type 2 Service Specific Convergence Sublayer for Narrow-Band Services," June 2000.

[19] Notker Gerlich and Michael Menth, "The Performance of AAL-2 Carrying CDMA Voice Traffic," in *11th ITC Specialist Seminar*, Yokohama, Japan, Oct. 1998.

[20] Michael Menth and Notker Gerlich, "A Numerical Framework for Solving Discrete Finite Markov Models Applied to the AAL-2 Protocol," in *MMB '99, 10th GI/ITG Special Interest Conference*, Trier, Sep. 1999, pp. 0163–0172.

[21] Bob Braden, David Clark, and Scott Shenker, "RFC1633: Integrated Services in the Internet Architecture: an Overview," http://www.ietf.org/rfc/rfc1633.txt, June 1994.

[22] Steven Blake et al., "RFC2475: An Architecture for Differentiated Services," ftp://ftp.isi.edu/in-notes/rfc2475.txt, Dec. 1998.

[23] Eric C. Rosen, Arun Viswanathan, and Ross Callon, "RFC3031: Multiprotocol Label Switching Architecture," http://www.ietf.org/rfc/rfc3031.txt, Jan. 2001.

[24] Michael Menth and Norbert Hauck, "A Graph Theoretical Concept for LSP Hierarchies," Technical Report, No. 287, University of Würzburg, Institute of Computer Science, Nov. 2001.

[25] Lou Berger and Jason Jeffords, "MPLS/IP Header Compression," http://www.ietf.org/internet-drafts/draft-berger-mpls-hdr-comp-00.txt, Jan. 2000.

[26] Lou Berger and Jason Jeffords, "MPLS/IP Header Compression over PPP," http://www.ietf.org/internet-drafts/draft-berger-mpls-hdr-comp-over-ppp-00.txt, March 2000.

[27] Michael Menth and Stefan Schneeberger, "A Scalable Scheduling Mechanism with Application to AAL2/ATM Multiplexing," in *14th ITC Specialist Seminar*, Barcelona, Spain, April 2001.