IPv6 Performance Evaluation on Dedicated Channels in UMTS

Andreas Kemper, Matthias Malkowski, Ibrahim Karacan
Communication Networks
Aachen University of Technology
Kopernikusstr. 16
52074 Aachen, Germany
E-Mail: {kem|mal}@comnets.rwth-aachen.de
WWW: http://www.comnets.rwth-aachen.de/~{kem|mal}

Abstract

With the introduction of different multimedia services for the *Universal Mobile Telecommunication System* (UMTS), performance of the TCP/IP protocol becomes a crucial issue. Using IPv4 for this purpose appears to be a good approach due to its fairly well-known behaviour. On the other hand, specific demands of the upcoming applications are not sufficiently covered with this protocol. Typical problems arise from multicast transmissions or when QoS support is required.

With the introduction of the new IPv6, many of these drawbacks have been solved, while performance of IPv6 in conjunction with the 3GPP protocols is mostly unknown. Based on the accurate implementation of IPv4 and IPv6 in our UMTS protocol simulator, comparative studies between those protocols were carried out. While performance is obviously affected by the channel quality, several simulations have been made in this context. Furthermore, the effect of using selected compression algorithms in the *Packet Data Convergence Protocol* (PDCP) has been investigated.

I. Introduction

The delivery of multimedia services to the mobile user is one of the goals of 3rd generation mobile communication systems. UMTS [1, 2] will provide data services with data rates of up to 144 kbps in rural areas, 384 kbps in hotspots and up to 2 Mbps in indoor scenarios. Disregarding the actual bandwidth, the mentioned applications are typically based on the TCP/IP protocol stack [3]. In contrast to the Internet, the *UMTS Terrestrial Radio Access Network* (UTRAN), in conjunction with the 3GPP protocol stack, imposes specific parameterizations for the TCP/IP [4]. Furthermore, new applications like video broadcasting, are demanding for new features, such as multicast support. This criterion together with some other issues, like enhanced QoS support, imply the use of IP version 6 (IPv6) [5–8] rather than IP version 4 (IPv4).

Considering the different protocol versions, this paper first of all presents the enhancements of IPv6 versus IPv4. Following this summary, in particular delay and throughput characteristics are presented. For this purpose the *UMTS Radio Interface Simulator* (URIS), a flexible UMTS protocol simulator, has been used. Besides an accurate implementation of the TCP/IP protocols for both versions , it also offers an almost complete 3GPP protocol stack for UMTS. The interaction of IETF and 3GPP protocols enables for comprehensive investigations of the overall system behaviour. Furthermore, the interaction between the IP layer and the *Radio Link Control* (RLC) layer has been considered in more detail. The required adaptation is provided by the PDCP [9], which also implements different methods for IP header compression. Since IPv6 initially generates more overhead due to larger headers, appropriate compression is mandatory.

II. Simulator Design

Investigation of IPv6 versus IPv4 protocol behaviour requires a deeper understanding of the interactions inside the protocol stack. Hence, all following simulations were carried out with the URIS as a powerful protocol simulator. Its core consists of an almost bit-accurate model for the 3GPP layers one to three, as well as the IETF layers three and four. The implemented protocol stack structure can be found in fig. 1, indicating also the separation of different planes in UMTS. The *Radio Resource Control* (RRC) is located in the control plane with the ability to manage and configure the entire protocol stack. On the other hand, TCP/IP can be found in the user plane to provide the appropriate interface for packet switched connections. Adaptation towards the 3GPP protocol stack is realized by the PDCP, which is part of the UMTS specification. Furthermore, circuit switched services, like typically voice calls, are also connected to the user plane, bypassing the upper layers.

Apart from the *Broadcast Multicast Control* (BMC) and RRC algorithms, all of the indicated protocols are implemented in the *System Description Language* (SDL). This is done individually for every mobile and base station. In addition, the URIS provides various types of *Load Generators* (LGs) on top of the SDL implementation. LGs are able to produce artificial load of various services in a detailed manner. Besides different voice codecs and video streaming services, in particular the generation of FTP and HTTP traffic has been used for the investigations. FTP load generation produces downlink-only traffic, since the amount of *uplink* (UL) data can usually be neglected. HTTP requires true bidirectional traffic modeling, including a much more detailed model to describe web-page loading, rather than individual file transfers. Below the SDL environment, embedding the protocol stacks, a simple channel model has been implemented.

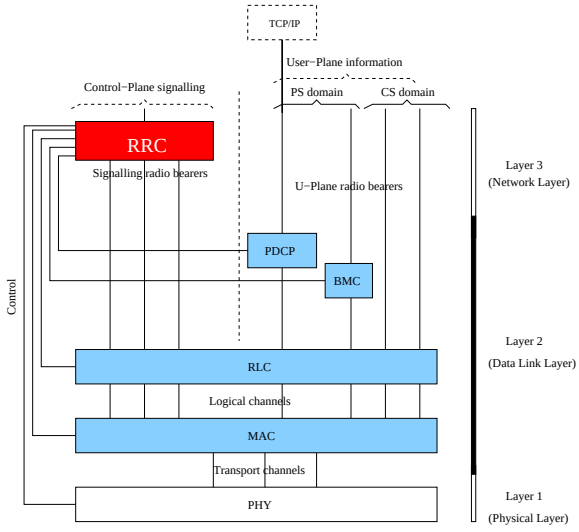


Figure 1: URIS Structure Overview

III. IP Headers and Compression Modes

Considering the focussed simulations, at least a basic understanding of the differences between IPv4 and IPv6 is necessary. The required implementations not only affect the network layer, but also PDCP in layer two and certain adaptations for the transport protocol in layer four. Nevertheless, the most important changes can be found in the IP layer, or to be more precise, in the format of the appropriate IP headers. For convenience, an overview of the IPv4 header is shown in fig. 2, whereas the IPv6 header is depicted in fig. 3. According to their relevance towards header compression, the different fields are marked with different shapes in both cases. In general it can be stated, that the basic IPv6 header is larger compared to the IPv4 header. On the contrary, less different fields and a constant length can be identified for the IPv6 header as compared to IPv4.

For routing issues the latter aspect is a clear advantage of the new header format, since evaluation of fewer headers makes routing decisions faster and simpler. Beyond that, the absence of IP checksums in fig. 3 further increases processing speed, since physical and data link commonly provide CRCs. Further enhancements of the protocol version six include the addressing capabilities [10], a better support for options and a flow labeling field.

The address-range has been extended by four times from 32 bits to 128 bits for sender and receiver together with a new, hierarchical structure. Options are not anymore a variable length part of the basic header, but encapsulated in so-called extension headers in IPv6. This yet simplifies routing, since only additional routing extension headers need to be further evaluated by intermediate hops. Furthermore, routing can be even simplified by using the flow label field. By calculating a specific value for this field, packets of the same connection can be quickly identified without parsing all the other entries. Contrary to many obvious advantages, the relatively large initial header from IPv6 is a clear disadvantage. This is in particular bad on mobile links with fairly limited bandwidth, especially if small packets are used, for instance with voice over IP.

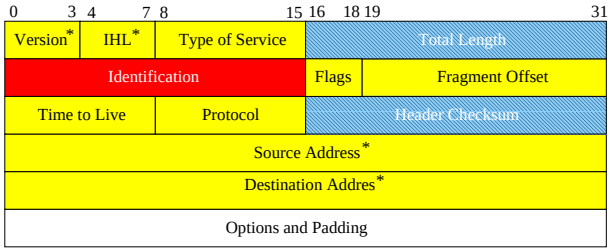


Figure 2: Compression of IPv4 Headers

A solution to this problem is the application of compression within the PDCP sublayer. According to the specification in [11], *Internet Protocol Header Compression* (IPHC) is one of the commonly used algorithms. Apart from IPHC, also *RObust Header Compression* (ROHC) [12] or even *TCP Aware Robust Header Compression* (TAROC) [13] might be used in future systems. Since specification in particular for TAROC is not yet complete, the presented results are all based on the IPHC only. Robust header compression basically compresses IP headers, but also TCP and UDP headers. On the other hand, TAROC and ROHC show better performance in particular on disturbed connections. Nevertheless, one of the main issues have been investigations with IPHC on perturbed channels, with respect to different header formats.

Considering the headers in more detail, different field types with respect to the applicable compression can be identified. Starting with the IPv4 header, most of the fields (slightly shaded) contain static information, which stays the same during one entire connection. These, so-called defining-fields, can be simply suppressed after the initial transmittion of a full header. In case of corrupt packets also an intermediate full header has to be sent to recover the according receiver context. Apart from this rather static fields, both header types contain at least one field with context related information (medium shaded). Typical examples are the total length of the packet or in fig. 2 also the header checksum. Since the *Header Checksum* is not required anymore and *Total Length* can be calculated within RLC, both fields are not transmitted at all. On the contrary the third non-static field-type in the header is only present in the old IPv4. The heavily shaded *Identification* field changes from packet to packet in a sequential and predictable manner. Here the IPHC can easily use delta-compression, meaning that under normal conditions only the relative difference has to be transmitted.

Summarizing the explanations, it can be found that the IPv4 header with initially 160 bits can be scaled down as following. For the defining fields, 112 bits can be suppressed after successful context creation. 32 bits from the context dependent fields plus eventually options have to be transfered with every packet. Finally a few bits are necessary to transport the information from delta-encoding, ending up in approx. one quarter of the initial header length for most of the transfered packets. In comparison to IPv4, the new protocol depicted in fig. 3 makes compression even simpler. The *Identification* field is not present anymore, while nearly all other headers contain defining, e. g. semi-static, entries. Furthermore, the *Payload Length* is not necessary anymore, since packet length is known from RLC. In summary this results in a theoretical reduction from initially 312 bits down to 16 bits, which equals a reduction of almost 95%.

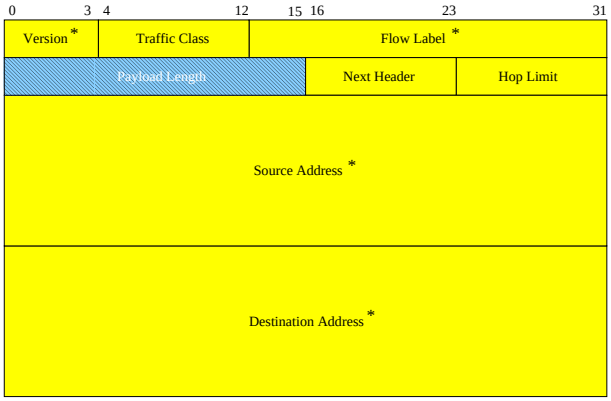
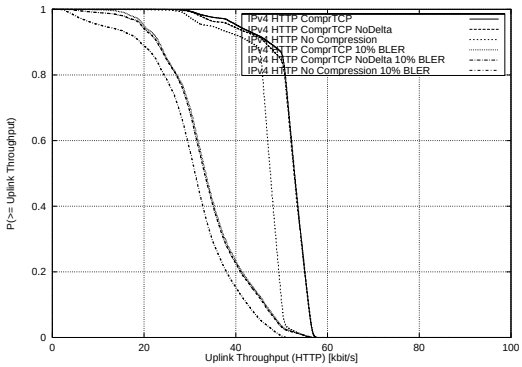


Figure 3: Compression of IPv6 Headers

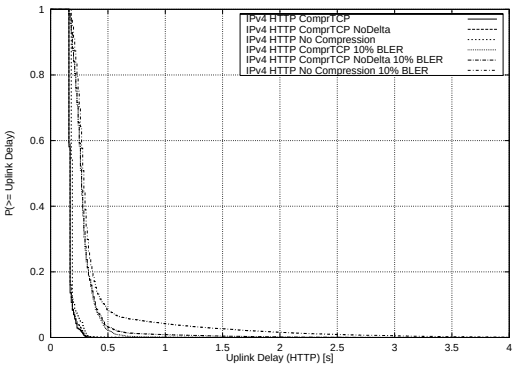
IV. BLER Simulation Results

The BLER scenario investigates the effect of an erroneous channel on the performance of the Internet Protocols by using different header compression algorithms. A 64 kbit/s channel with a drop probability (BLER) of 10% and the maximum segment size of 512 bytes for TCP segments is considered. In contrast to the FTP simulations depicted in fig. 5, HTTP traffic has been considered in the BLER scenario according to fig. 4. Load generator parameterization for both scenarios can be found in tab. 1, whereby either FTP traffic or HTTP traffic is used.

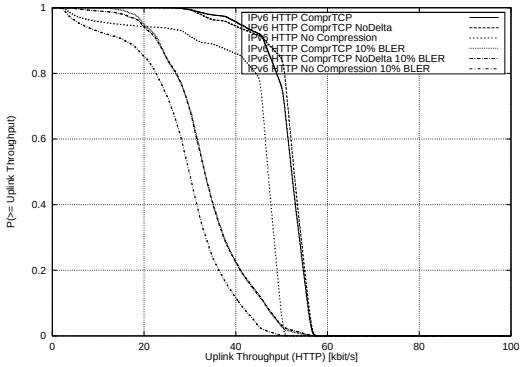
Taking a closer look at the diagrams, fig. 4(a) to fig. 4(d) present the outcomes for the HTTP *downlink* (DL), being predominantly used. Complementary, the graphs from fig. 4(e) to fig. 4(h) show the uplink behaviour. Since the UL has to carry a lot less traffic, mostly in terms of URL requests, it shows a significantly different behaviour. Furthermore, diagrams can be column-wise distinguished by either presenting throughput results on the left hand and delay on the right hand. Starting with the most obvious differences for the uplink throughput, fig. 4(a) and fig. 4(c) are now considered in more detail.



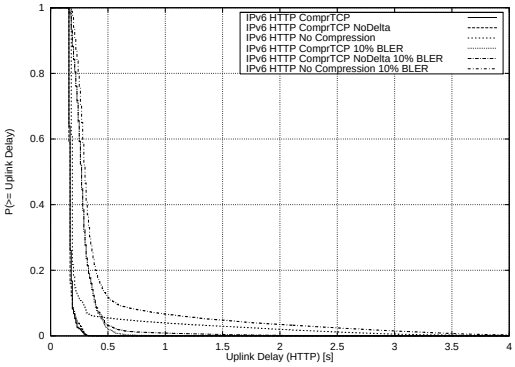
(a) **DL** - Throughput, HTTP, IPv4, 10% BLER



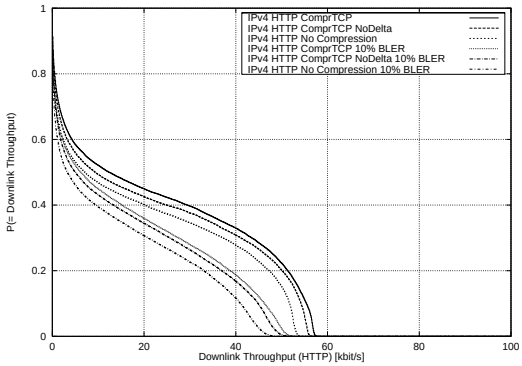
(b) **DL** - Delay, HTTP, IPv4, 10% BLER



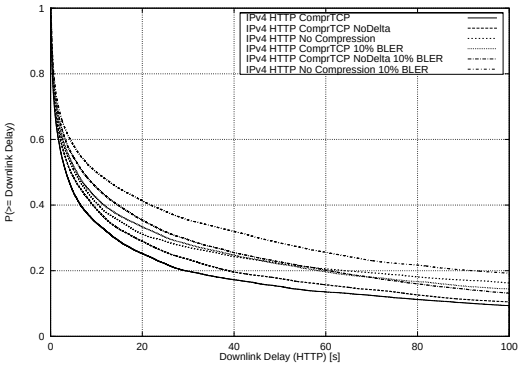
(c) **DL** - Throughput, HTTP, IPv6, 10% BLER



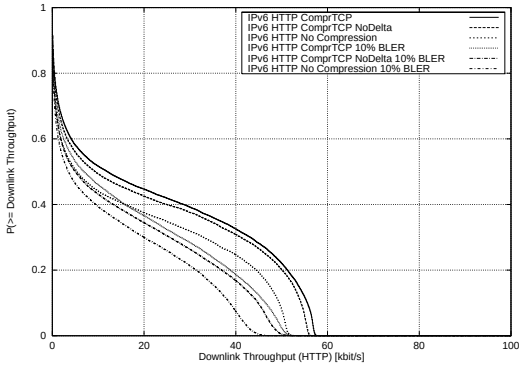
(d) **DL** - Delay, HTTP, IPv6, 10% BLER



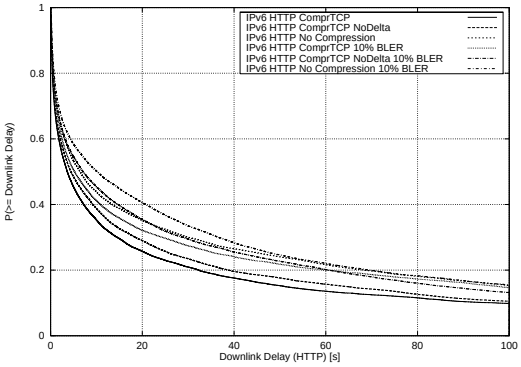
(e) **UL** - Throughput, HTTP, IPv4, 10% BLER



(f) **UL** - Delay, HTTP, IPv4, 10% BLER



(g) **UL** - Throughput, HTTP, IPv6, 10% BLER

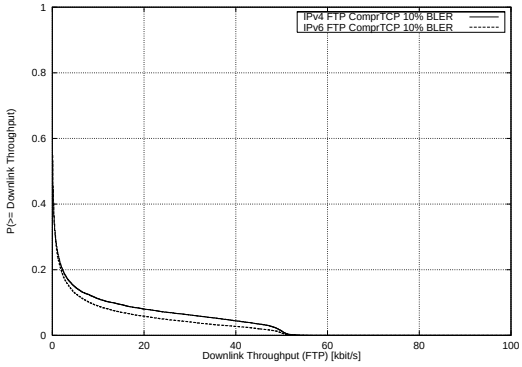


(h) **UL** - Delay, HTTP, IPv6, 10% BLER

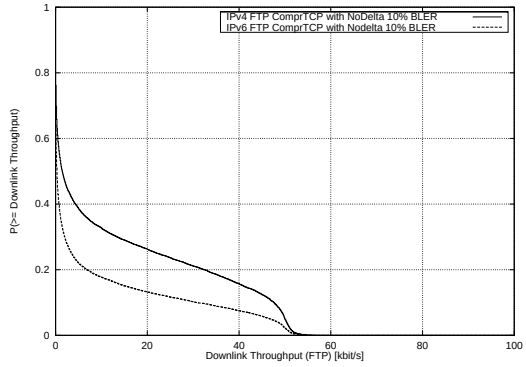
Figure 4: HTTP BLER Simulation Results

HTTP Parameter	Distribution	Mean	Variance
Session Arrival Rate $[h^{-1}]$	negative exponential	30	—
Pages per Session	geometric	5	—
Reading Time between Pages [s]	negative exponential	20	—
Objects per Page	geometric	2.5	—
Inter Arrival Time between Objects [s]	negative exponential	0.5	—
Page Request Size [byte]	normal	1136	80
Object Size [byte]	$\log_2$-Erlang-k	$\log_2 2521 \approx 11.3$	$(\log_2 5)^2 = 5.4$
FTP Parameter	**Distribution**	**Mean**	**Variance**
Session Arrival Rate $[h^{-1}]$	negative exponential	30	—
Session Size [bytes]	$\log_2$-normal	$\log_2 32768 \approx 15$	$(\log_2 16)^2 \approx 16$
Object Size [bytes]	$\log_2$-normal	$\log_2 3000 \approx 11.55$	$(\log_2 16)^2 \approx 16$
Time between Objects [s]	$\log_{10}$-normal	$\log_{10} 4 \approx 0.6$	$\log_{10} 2.55 \approx 0.4$

Table 1: Model Parameters of HTTP Browsing Sessions and FTP Sessions



(a) **DL** - Throughput, FTP, FullHeader



(b) **DL** - Throughput, FTP, NoDelta

Figure 5: FTP DL Compression Results

Since HTTP requests typically produce fairly small packets a vast influence can be noticed from the effectiveness of header compression. For IPv4 basically four different cases can be distinguished. The best performance can be obviously found for the error-free channel using optimal IPHC, including delta compression.

On the contrary the bad channel with a 10% BLER and no compression at all performs worst. Still a fairly good performance can be found for the error-free channel without any compression compared to the perturbed channel with either full or no-delta compression. This difference clearly indicates the problems IPHC has on erroneous channels, where frequent context recoveries largely decrease performance. Similar results can be found for using IPv6 as shown in fig. 4(c). The most obvious difference can be found in the performance for error-free channels without using compression at all. Since IPv6 packets are larger, the amount of connections with a throughput less than approx. 30 kbps increases.

Delay shows almost exactly complementary behaviour compared to the throughput and is depicted in fig. 4(b) and fig. 4(d). In general, differences are not that significant as for the throughput, whereas again larger headers on ideal channels without compression are still more critical. Comparing the uplink results to those from the downlink in fig. 4(e) to fig. 4(h), the differences are not that obvious anymore. This confirms the asymmetric HTTP traffic behaviour, as produced by the load generators. While a typical URL usually contains only a small amount of bytes, WWW page objects, like images or animations, produce larger IP frames. Thus, the downlink still benefits from header compression in both cases and faces the same problems on bad channels. Nevertheless, the extra amount of data produced by the header is by far not that critical for performance.

V. PDCP Compression Results

The simulation results for the error-rate scenario have also shown that the IPHC does not perform well on channels with high BLERs. The reasons for this behaviour are full headers that have to be send each time a packet gets lost to recover the context. On the other hand, the no-delta compression has performed relatively robust on such channels. In the following scenario these header compression methods are combined. Packets are by default compressed here including delta values. In case of packet losses packets just without delta values, instead of full headers, are sent. Following figures show exemplary results for FTP throughput on a DL channel with 10% BLER. The performance reached by the combination of two compression methods in fig. 5(b) is higher than that obtained by each one separately in fig. 5(a)).

VI. Conclusion

Simulation results have shown that header compression increases the throughput significantly in most cases. As expected, compression for IPv6 headers is in general more profitable compared to IPv4 headers, due to the higher overhead. Furthermore, TCP compression with differential encoding, which provides the best compression rate, does not perform well on channels with high error-rates. In contrast to that, TCP compression without differential encoding shows a robust behaviour on such channels, even if its compression rate is worse than that the delta compression provides.

Since no-delta compression does not invalidate the context, the packets which come after a lost packet can still be decompressed correctly. The delta compression, on the other hand, causes to send a full header each time a packet-loss occurs. While the delta compression is most

effective and the no-delta compression is robust against packet losses, these algorithms can be combined to reach a better overall performance. This can be achieved by switching the compression algorithms with respect to the current context and channel state.

References

[1] B. Walke, *Mobile Radio Networks; Networking, Protocols and Traffic Performance*, Wiley, 2nd edition, 2002.

[2] W. Lu Willie, *Broadband Wireless Mobile; 3G and Beyond*, Wiley, PO19 8SQ , England, 2002.

[3] Information Sciences Institute, "Internet Protocol, Darpa Internet Program Protocol Specification," Standards Track RFC791, University of South California, September 1981, `http://www.ietf.org/rfc.html`.

[4] Ed. Wasserman, "Recommendations for IPv6 in Third Generation Partnership Project (3GPP) Standards," Informational RFC3314, Internet Engineering Task Force, September 2002, `http://www.ietf.org/rfc.html`.

[5] H. P. Dittler, *IPv6-Das neue Internetprotokoll Technik-Anwendung-Migration*, dpunkt.verlag, Ringstrasse 19 b, 69115 Heidelberg, 2nd edition, 2002.

[6] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," Standards Track RFC1883, Internet Engineering Task Force, December 1995, `http://www.ietf.org/rfc.html`.

[7] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification, =====," Standards Track RFC2460, Internet Engineering Task Force, December 1998, `http://www.ietf.org/rfc.html`.

[8] H. Wiese, *Das neue Internetprotokoll IPv6; Mobilitaet, Sicherheit, unbeschraenkter Adressraum und einfaches Management*, Hanser Verlag, Muenchen Wien, 2002.

[9] 3GPP, "Universal Mobile Telecommunications System (UMTS); Packet Data Convergence Protocol (PDCP)," Technical Specification 3GPP TS 125 323 v5.2.0 (2002.09), European Telecommunications Standards Institute, September 2002, `http://webapp.etsi.org/key/queryform.asp`.

[10] R. Hinden and S. Deering, "IP Version 6 Addressing Architecture," Standards Track RFC2373, Internet Engineering Task Force, July 1998, `http://www.ietf.org/rfc.html`.

[11] M. Degermark, "Internet Protocol Header Compression," Standards Track RFC2507, Internet Engineering Task Force, February 1999, `http://www.ietf.org/rfc.html`.

[12] C. Bormann, M. Degermark, and H. Fukushima, "RObust Header Compression (ROHC)," Standards Track RFC3095, Internet Engineering Task Force, July 2001, `http://www.ietf.org/rfc.html`.

[13] H. Liao, Q. Zhang, W. Zhu, and Y. Zhang, "TCP-Aware RObust Header Compression (TAROC)," Internet draft, Internet Engineering Task Force, July 2001.