Improved IPv6 Packet Handling by Header Compression in UMTS

Andreas Kemper, Ibrahim Karacan

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. A key difference to existing *IP version 4* (IPv4) implementations is the usage of *IP version 6* (IPv6) for data services in UMTS. Apart from various advantages of the new protocol version, IPv6 itself uses a significantly enlarged header as compared to IPv4. In combination with a relatively low amount of data per packet, this turns out to be a crucial issue in bandwidth-limited wireless systems.**

Hence the following investigations are concerned about the specific (dis-) advantages when using IPv6 instead of IPv4 in UMTS. Special respect is drawn to the effect of header compression, as provided by the *Packet Data Convergence Protocol* (PDCP). Apart from this, particular services, like *Voice of IP* (VoIP), are known to produce fairly small packets. Consequently, the second part of the investigations is dedicated towards the effects of using different *Maximum Segment Sizes* (MSS) and packet sizes in the IP layer.

Keywords—**UMTS, IPv4, IPv6, PDCP, Header Compression, Performance Evaluation, Protocol Simulation, System Modelling**

I. Introduction

The wireless delivery of multimedia services is one of the goals of 3rd generation mobile communication systems. UMTS [1, 2] will provide data services with different data rates of up to 2 Mbps, depending on the current scenario. Disregarding the actual bandwidth, the mentioned applications are typically based on the TCP/IP protocol stack [3]. Different to original Internet applications, the *UMTS Terrestrial Radio Access Network* (UTRAN), in conjunction with the 3GPP protocol stack, imposes specific parameterizations for the TCP/IP [4]. Furthermore, for instance video broadcasting, is demanding for new features, such as multi-cast support. Last but not least this criterion, together with some other issues, imply the use of IPv6 [5–8] rather than IPv4.

With respect to the intended investigations, this paper first of all presents a brief overview on the structure of the simulator. Following, the differences of both header versions with special respect to TCP and UDP headers are presented. Furthermore, the implemented compression mechanisms are shortly explained and exemplary depicted for the TCP header. Using the rather simple UDP header, a practical calculation of the achievable gain is shown. Following this summary, delay and throughput characteristics for a file download via the *File Transfer Protocol* (FTP) are presented. Thereby special attention is payed to different MSSs and changing FTP packet sizes.

For this purpose the *UMTS Radio Interface Simulator* (URIS), a flexible UMTS protocol simulator, has been used. Apart from an accurate implementation of the considered TCP/IP protocols, it also offers a rather complete 3GPP protocol stack for UMTS. Considering the interaction of IETF and 3GPP protocols enables for comprehensive investigations of the overall system behaviour. In particular the interaction between the IP layer and the RLC layer has been considered in more detail. The required adaptation is provided by the PDCP, while this layer also implements different header compression algorithms. Since IPv6 initially generates more overhead due to larger headers, appropriate compression is highly recommended in most cases.

A. Kemper is with the Chair of Communication Networks, Aachen University, Germany. E-mail: kem@comnets.rwth-aachen.de

II. Simulator Design

Investigations on protocol level require a deeper understanding of the interactions inside the different layers. Therefore, the URIS 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 structure can be found in fig. 1, indicating also the two different planes in the 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 means of the PDCP, which is also part of the UMTS specification.

Most of the protocols, apart from *Radio Resource Management* (RRM) algorithms, are implemented in the *System Description Language* (SDL). This is done individually for mobile and base stations. In addition, the URIS provides various types of *Load Generators* (LGs) on top of the SDL implementation. Besides different multimedia services, in particular the generation of FTP load has been used for the investigations. It produces downlink-only traffic, since the amount of uplink data can usually be neglected. Below the SDL environment, embedding the protocol stacks, a simple channel model has been used.

III. TCP/IP Headers and Compression Modes

According to [9] different types of header compression mechanisms are available. As briefly discussed in [10], not only the headers added by the network protocols IPv4 or IPv6 can be compressed. Furthermore, also the overhead generated by the transport layer, namely from TCP or UDP, can be largely reduced by compression. An overview on the overall achievable ratios is therefore presented in tab. III. Different to the initial overhead, the remaining amount of data after compression is not always constant. This is essentially related to the fact, that for instance different compression algorithms are available for TCP.

As depicted in fig. 2, basically three different field types can be distinguished in the TCP header. These include the static fields, namely the ports and the packet offset (light gray). Using compression, based on context information

TABLE I
COMMON PARAMETERS OF FTP SESSIONS

FTP Parameter	Distribution	Mean	Variance
Session Arrival Rate $[h^{-1}]$	negative exponential	20	—
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	$\log_{10} 4 \approx 0.6$	$\log_{10} 2.55 \approx 0.4$

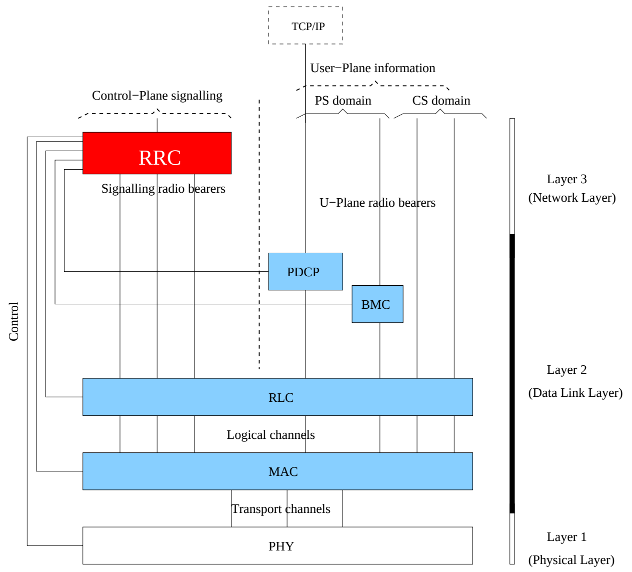


Fig. 1. URIS structure overview

at the decompressor, these fields can be entirely omitted during transmission. Furthermore, most of the other fields (dark gray) contain information usually changing only in small steps or even not at all. These are predestined for delta encoding, namely only transmitting the differences, in case they occur. Third field type to mention are those containing pure random information (transparent), which can not be compressed at all. Considering the figure, only the checksum field belongs to this type, while it will always be transmitted to enable for validation at the receiver.

Looking at the resulting, fully compressed header, its structure looks typically like in fig. 3. It contains now just four mandatory fields (light gray), including the *ContextID* (CID), flags indicating individual delta encoded values and the two byte TCP checksum. Additionally, it might contain random fields (dark gray) and following delta values (transparent), in case the appropriate flags are set.

To provide a simple example for the achievable gain by compression, the combination UDP with IPv6 should now be considered. According to [10] the IPv6 header itself has an initial length of 40 bytes, while UDP (fig. 4) adds another eight bytes. Finally this ends up in a raw overhead of 48 bytes. Using most efficient compression reduces the amount of IP header overhead to zero bytes, since no random or delta encoded fields are included. The only remaining four bytes are resulting from the UDP header. These include just one byte for the CID, one byte for the *Generation value* and two bytes for random fields. The latter are resulting from the header checksum (16 bits), which is required by the receiver to validate the packets. Finally, this results in an absolute reduction of 48 bytes down to four bytes or approx. 9% down to <1% relative to a MSS of 512 bytes.

TABLE II
VARIABLE PARAMETERS OF FTP SESSIONS

FTP Parameter	Value
Packet Size (large) [bytes]	10000
Packet Size (small) [bytes]	2048
MSS (large) [bytes]	1024
MSS (small) [bytes]	512

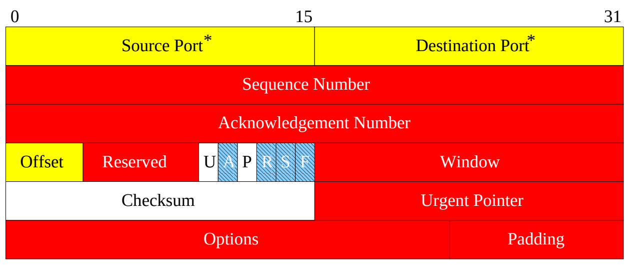


Fig. 2. Uncompressed TCP header

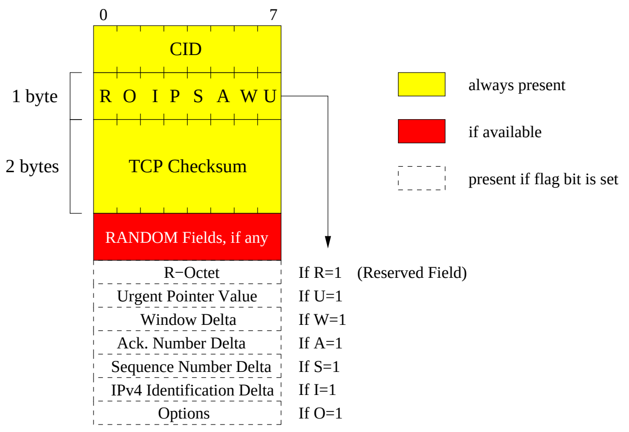


Fig. 3. Full compressed TCP header

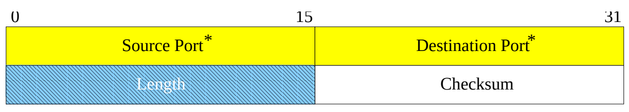
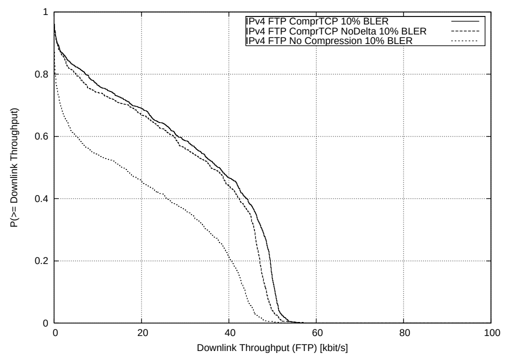
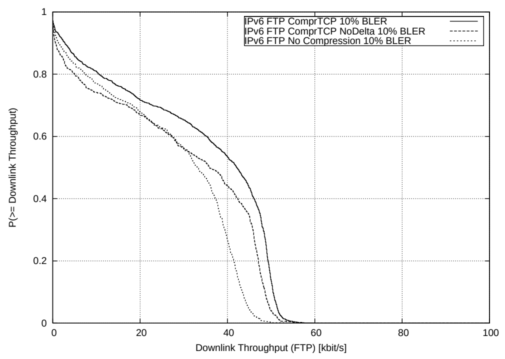


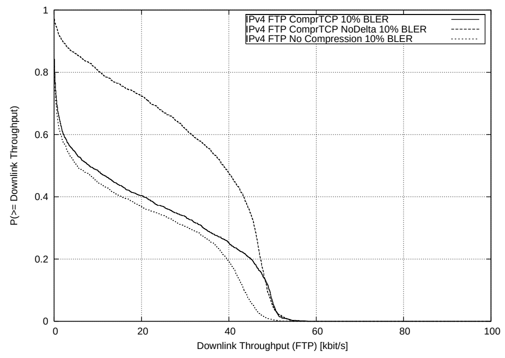
Fig. 4. Uncompressed UDP header



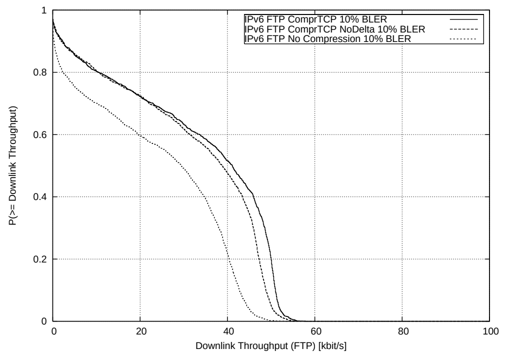
(a) Throughput, IPv4, 0.5 kB MSS, Large Packets



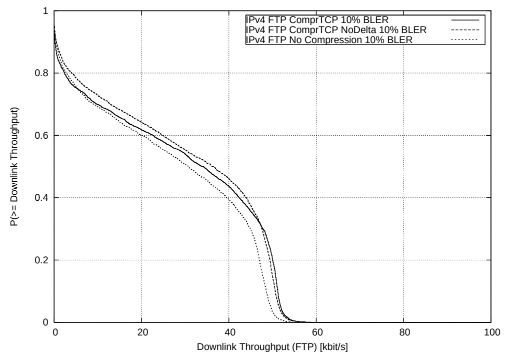
(b) Throughput, IPv6, 0.5 kB MSS, Large Packets



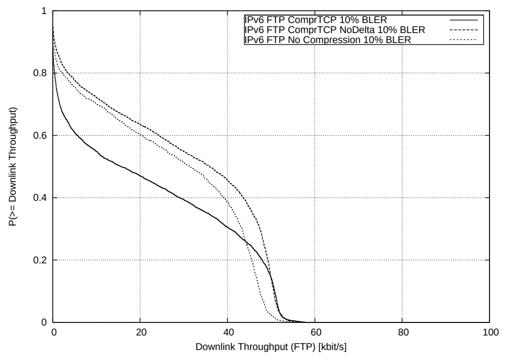
(c) Throughput, IPv4, 0.5 kB MSS, Small Packets



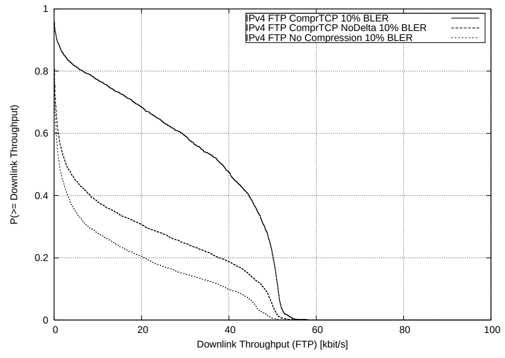
(d) Throughput, IPv6, 0.5 kB MSS, Small Packets



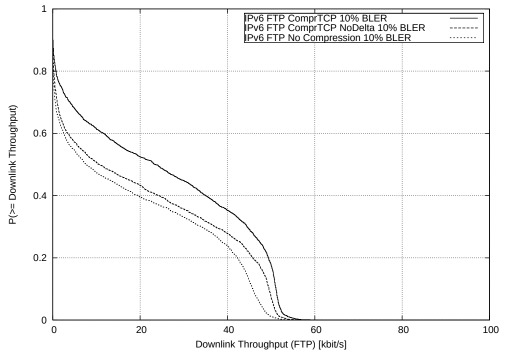
(e) Throughput, IPv4, 1 kB MSS, Large Packets



(f) Throughput, IPv6, 1 kB MSS, Large Packets

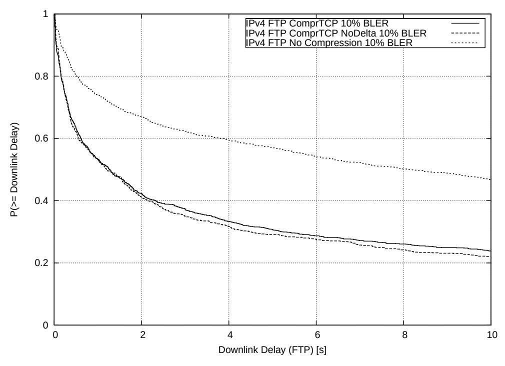


(g) Throughput, IPv4, 1 kB MSS, Small Packets

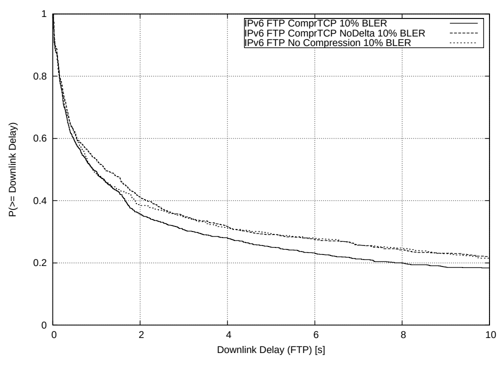


(h) Throughput, IPv6, 1 kB MSS, Small Packets

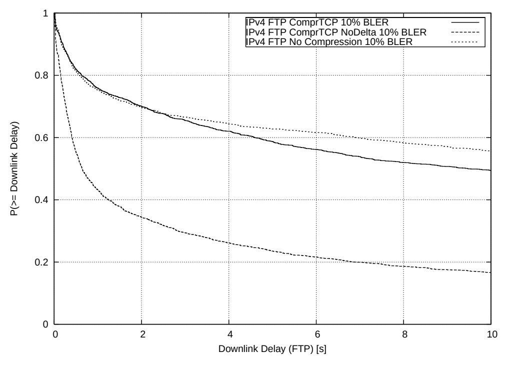
Fig. 5. FTP throughput results



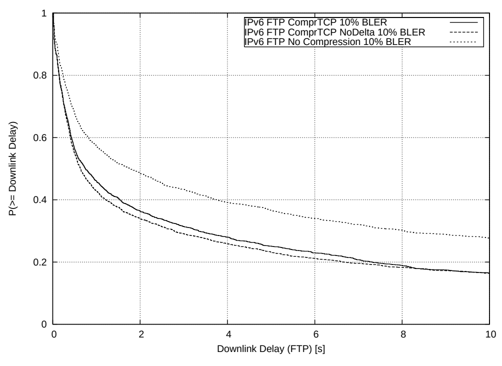
(a) Delay, IPv4, 0.5 kB MSS, Large Packets



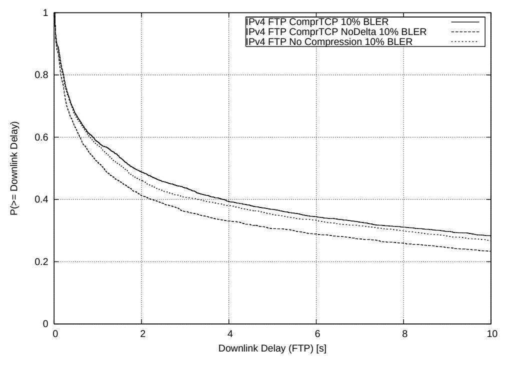
(b) Delay, IPv6, 0.5 kB MSS, Large Packets



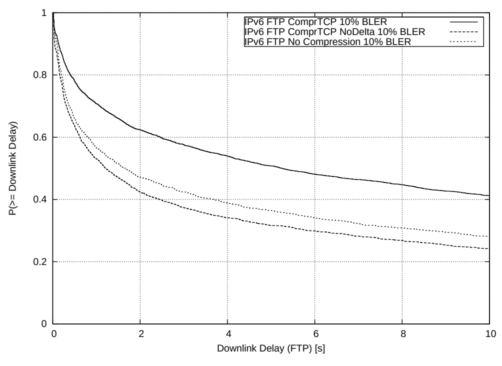
(c) Delay, IPv4, 0.5 kB MSS, Small Packets



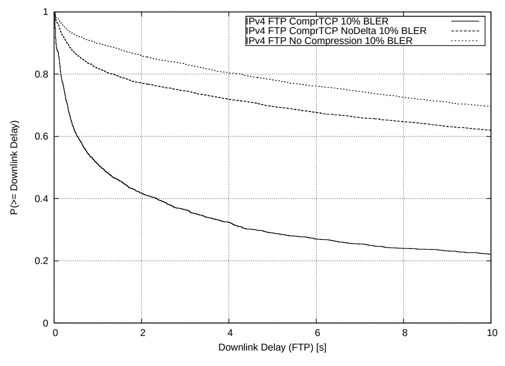
(d) Delay, IPv6, 0.5 kB MSS, Small Packets



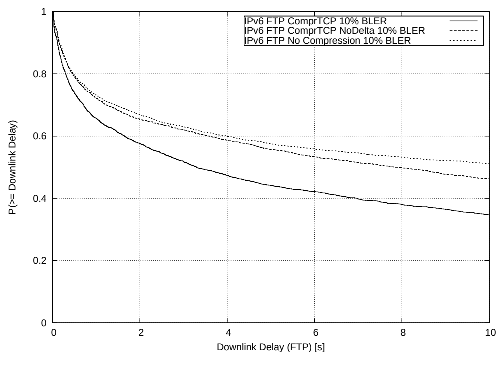
(e) Delay, IPv4, 1 kB MSS, Large Packets



(f) Delay, IPv6, 1 kB MSS, Large Packets



(g) Delay, IPv4, 1 kB MSS, Small Packets



(h) Delay, IPv6, 1 kB MSS, Small Packets

Fig. 6. FTP delay results

TABLE III
HEADER OVERHEAD COMPARISON (IN BYTES)

Protocol	Segment	Header	Overhead	Compr. Hdr.	Overhead
TCP/IPv4	512	40	7.8%	<10	<1%
TCP/IPv6	512	60	11.7%	<10	<1%
UDP/IPv4	512	28	5.5%	4	0.8%
UDP/IPv6	512	48	9.4%	4	0.8%
TCP/IPv4 (mobile IP)	512	60	11.7%	<10	<1%
TCP/IPv6 (mobile IP)	1500	100	7.1%	4-7	0.4%
TCP/IPv6 (mobile IP)	512	100	19.5%	<10	<1%

IV. SIMULATION SETUP

Simulations have been carried out with the already mentioned protocol simulator URIS. To simplify the understanding of effects related to header compression and changing MSS, FTP as a rather simple traffic model has been chosen. Assuming by far most of the traffic on the downlink, the uplink has been entirely neglected. Consequently, the general parameters for the downlink can be found in tab. I, whereas scenario-specific params are in tab. II. The model for FTP traffic generation is based on a certain average amount of sessions per hour. Within every session a variable amount of bytes is generated. This corresponds to the overall amount of data from multiple files. After all every file has its individual size and transmission time, which is modeled by the object size and the time between the objects within every session.

Apart from the mentioned static parameters, the protocol stack has been reconfigured multiple times to investigate different scenarios. These can be distinguished by either using IPv4 (throughput in left column of figs. 5, delay in left column of figs. 6) or IPv6 (throughput in right column of figs. 5, delay in figs. 6). Furthermore, two different MSSs of 512 bytes (upper block of the plots) or 1024 bytes (lower block of the plots) have been parameterized. Finally, also the maximum packet size, as generated by the FTP traffic generators, has been set to either 10.000 bytes or 2048 bytes. Packet size and MSS together are of great importance for the grade of fragmentation in the IP layer, thus careful adaptation appears to be mandatory. On the other hand, using either IPv4 or IPv6 also has a significant influence on the performance.

Summarizing the expected effects, the following remarks can be made in advance. *Increasing the MSS* reduces the relative header overhead and thus should in general allow for a higher throughput. Furthermore, also the delay might increase, since the transmission of every single packet needs more time. On the contrary, *increasing the packet size* results in less context re-creations for the decompressor, since the average amount of packets per session is reduced in this way. Finally, using *IPv6 instead of IPv4* requires much more data to be transfered for the raw header information. On the other hand, the IPv6 header in general allows for entire compression down to zero bytes, which is not feasible for IPv4 headers. The latter at least contains a certain amount of random information, which has to be transfered within every packet.

Due to these differences in the header structures three different types of investigations have been made for every scenario. In particular with respect to the amount of delta-compressible fields performance differs significantly. They can be distinguished by either using no compression (dotted curve), compression without delta information (dashed curve) and full compression (solid curve), including differential encoding. Common to all investigations is a relatively bad channel with a 10% *Block Error Rate* (BLER). Channel quality in particular allows for investigations on the effects of frequently lossed compressed packets belonging to the same context.

V. SIMULATION RESULTS

Considering the results, the primary focus is on the throughput graphs, since the delay curves usually show the corresponding behaviour. Within the simulator, the throughput is calculated as the individual packet size divided by its delay. Thus good throughput performance is typically related to low delay values. Evaluating the first two IPv4 graphs (fig. 5(a) and fig. 5(c)), delta-compression strongly benefits from long packet sequences. These appear in particular if the generated FTP packets are relatively large and thus segmentation occurs frequently. On the contrary, with IPv6 the performance suffers especially in the uncompressed scenario (fig. 5(b) and fig.5(d)) when lowering the packet size. While here the maximum packet size is twice the MSS, small IP packets will occur frequently, containing a significant amount of header overhead.

Turning now over to the enlarged MSS of 1 kB, some other effects can be found. While larger MSS in case of IPv4 show better performance in particular for the uncompressed scenario (fig. 5(e)), this effect is hardly visible in the IPv6 case. Instead, fig. 5(f) shows a noticeably decreased performance for the fully compressed headers. Obviously, the context recovery for the enlarged IP packets needs more time, resulting in increased transmission delays. In general the combination of relatively large MSS and small packet sizes appears to be critical, as the two plots at the bottom (fig. 5(g) and fig. 5(g)) indicate. Reasons for this have been mentioned already, namely limited IP segmentation and/or frequently small packets.

VI. Conclusion

The performance with the default MSS of 512 bytes is often better for IPv6 (fig. 5(b) and fig.5(d)), compared to IPv4 (fig. 5(b) and fig.5(d)). Apart from protocol-inherent mechanisms it benefits in general from a more efficient header compression, as the values in tab. III already suggested. On the other hand, using IPv6 is not always beneficial, but strongly dependent on the correct parameter setup. The rather miserable throughput for the scenario using full compression in fig. 5(f) is such an example.

In general, simulation results have shown that compression of TCP/IP headers increases the throughput significantly in many cases. With regard to the overall results it appears to be mandatory for IPv6 due to the higher overhead. Furthermore, TCP/IP compression with differential encoding, which provides the best compression rate, does not always perform well on channels with high error-rates. In contrast to that, TCP/IP compression without differential encoding shows a robust behaviour on such channels.

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

[10] A. Kemper, M. Malkowski, and I. Karacan, "IPv6 Performance Evaluation on Dedicated Channels in UMTS," Conference paper, Chair of Communication Networks, Aachen University, February 2004.