MEDIATEK

Uplink Data Compression

3GPP Solutions for Enhancing the Uplink Performance

White paper

Introduction

Efficient spectrum usage and utilization for different types of user applications are becoming more and more important for the future cellular network deployments, in order to cope with increased data rate and capacity demands. This increases the need for supporting new features and methods such that spectrum, resources and latencies can be improved to reflect to the system performance and user experience. The exponential increase in the demand for data connectivity can put disproportionate pressure on either uplink or downlink performance, depending on the usage of the Smartphones and applications. **Figure 1** shows MediaTek survey on live network traffic and Smartphone profiling. As shown, users enjoy data-intensive streaming and social media services which are the most common usage for mobile internet. Applications can have different types of requirements for downlink/uplink throughput, packet latencies, Quality of Service, or resource assignment from network scheduler. Therefore, network operators are shifting the focus into how to improve the end-user experience and the network capacity with as many different types of data usage.

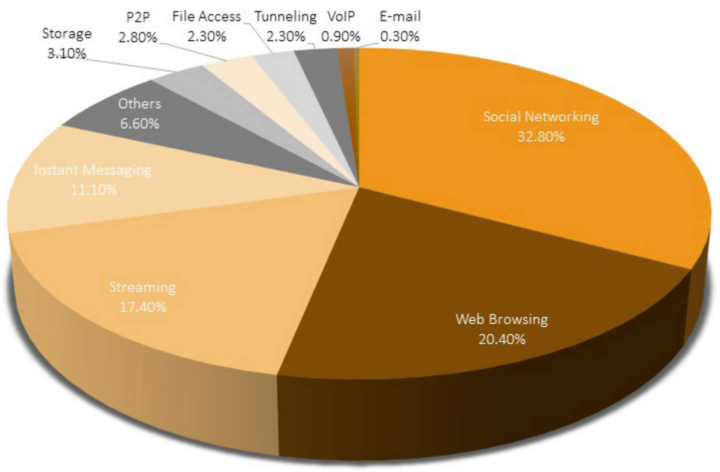

Figure 1: Typical Usage of Data/Applications in Today's Smartphone

There are many different techniques and features available now in 3GPP that can be used to improve the areas of throughput (both uplink and downlink), latencies, in addition to improved capacity, especially for users at cell edge. The categories of these features are summarized in table 1, some have been deployed and others have been specified in 3GPP already. The categories listed in the table provide enhancements to throughput, spectral efficiency, interference, coverage, mobility, and battery consumption.

Table 1: General 3GPP LTE & LTE-A Pro Features

LTE and LTE-A Pro Features	Brief Summary
Carrier Aggregation	Aggregation of spectrum including licensed and unlicensed bands for UL and DL to improve throughput, capacity and latencies
MIMO Techniques	Improve spectral efficiency with many types of MIMO Transmission modes, FD-MIMO, MU-MIMO, SU-MIMO
Higher Order Modulation	Improve spectral efficiency with in both UL and DL
Interference Cancellation and Suppression	Improve cell-edge throughput, capacity and latencies including interference from pilot and data/control channels
Shorter Transmission Time Interval (TTI)	Improve Latencies in ranges below 20 ms
SON & Mobility Enhancements	Features aiming at improved user throughput in handover region, and increase network efficiencies in mobility
Connected Mode Discontinuous Reception (C-DRX)	Improves battery consumption and resource utilization for bursty traffic

However, operators are looking into enhancements coming from techniques deployed at upper layers. One of which is the data compression feature. Compressing the uplink data between a UE and the EPS (Evolved Packet System) can provide gains for user throughput, latencies and capacity by performing such compression for HTTP traffic. This whitepaper discusses the techniques of doing data compression.

Motivation for Uplink Enhancements

The shortage of uplink resource becomes one of the major concerns in the network due to following reasons:

- More mobile internet users are becoming content producers that has become an obvious fact in the recent few years. This can be observed particularly at any kind of mass events—whether it is a football match, a concert requiring real time social media interaction—the uplink direction is now usually dominated for the traffic type rather than downlink in such events.
- The substantial increase of downlink data traffic when using Carrier Aggregation leading to a natural increase of uplink traffic. To satisfy requirements on UE battery consumption and reduce UE complexity, few uplink carriers are used, most typically only one though the number of UL 2xCA deployments now constantly increases.
- Typical UL/DL configuration in TD-LTE network is configuration 2, i.e. 6DL: 2UL subframes, as shown in **Figure 2.** It is quite often that the uplink becomes a bottleneck in case of, e.g. file uploading, video streaming, etc. Thus TD-LTE networks can become the primary beneficiaries of any uplink enhancement feature deployed.

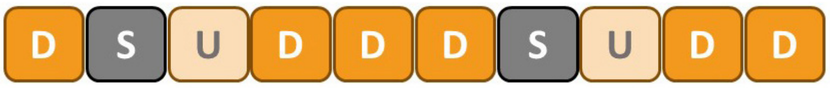

Figure 2: Typical TD-LTE configuration

Another concern for the uplink data transmission is the coverage vulnerability, namely in following aspects:

- As the number of LTE subscribers increases, the uplink interference level reaches 5~10 dB, making uplink transmission in poor radio condition difficult.
- Due to power limitation, RLC segmentation is a common way to extend uplink coverage. However, it is not a preferred solution in some cases because the segmentation size is tied to the RF conditions.
- During VoLTE call setup phase, a SIP message size is about 2KB. When UE is in poor radio condition (e.g. at RSRP of < -120dBm) and/or at high interference locations, it has been observed that a SIP message can be segmented into several RLC PDUs, thus average call setup time and call drop rate are increased. So very large SIP message size becomes a problem.

However, in network deployed with TDD, the complexity of capacity management becomes higher. In TDD, the frame is shared between both uplink and downlink subframes, and therefore, there can be a clear impact to the end-user throughput. In addition to the TDD frame structure, other channels require extra resources in-band with the TDD resources assigned for data channels. The scheduling based on the SIB (System Information Blocks) transmission. For example, SIB-1 requires 9 PRB (Physical Resource Blocks) and repeated every 20ms, and SIB-2/3 requires 15 PRB repeated every 160ms. These resources needed for multiple channels will impact the data throughput even further

Thus the motivation for any new feature introduction to the LTE networks has become obvious to the industry especially for LTE-TDD networks. One of these feature is Uplink Data Compression, UDC. This paper discusses the details of UDC feature, introduced in 3GPP Rel-15, with the following main targets:

- Increase uplink capacity when the amount of data sent in uplink is reduced.
- Increase the uplink and downlink throughput by sending less uplink data same amount of information (on downlink the benefits come from shorter TCP ACKs, needed for the same amount of downlink data).
- Improved uplink interference and coverage by reducing the uplink transmit power (Tx power) for the data channels when the amount of uplink data is reduced.
- Improved latencies by sending the same amount of information with less round trip time.

The other common method for compression has been to compress the header size through RoHC (Robust Header Compression). 3GPP in Rel-14 introduced UL-only RoHC as a complementary mechanism to improve compression performance in certain traffic types. This paper also discusses the details of this compression method.

Header Compression Methods in 3GPP

The radio spectrum resource is precious and has to be utilized economically. However, in the RTP /UDP/IP protocol structure of the services that are widely used, such as streaming services and VoIP; the header fields occupy 40 to 60 bytes for each payload that can be as small as 32 bytes. Transmitting the header fields of the RTP/UDP/IP protocol consumes a large amount of radio link resource. Therefore, 3GPP specified ways to compress the headers of RTP/UDP/IP packets. Typically, the PDCP (Packet Data Convergence Protocol) Header Compression undertakes this function.

Compression techniques are not new to 3GPP specifications. There has been several compression methods introduced in 3G UMTS time, with a focus on the headers given the large size with IPv4 and IPv6 packets. Generally speaking, PDCP header compression has two types of algorithms:

- IPHC (IP Header Compression): can apply to PS data services.
- RoHC (Robust Header Compression): used to apply to VoIP services only but lately considered to be more widely used.

IPHC uses the concept of packet stream context. A context is a set of data which contain field values and value change patterns in the packet header. For each packet stream, the context is formed at the compressor and the decompressor. After the context is established on both sides, the compressor can compress the packets. The compressor and decompressor store most fields of this full header as context. The context consists of the fields of the header whose values are constant and thus need not be sent over the link at all, or whose values change little between consecutive headers so that PDCP header compression uses fewer bits to send the difference from the previous value compared with sending the absolute value.

The IPHC schemes are suitable for radio links with strong link-level checksums but are not robust enough to handle high Bit Error Rate (BER) and long Round Trip Time (RTT). Because high BER and long RTT can commonly impact delay sensitive application, an efficient and robust compression scheme is needed. In this regard, the RoHC technique was developed.

RoHC is a header compression protocol originally designed for real-time audio/video communication in wireless environment. As VoIP media packets are transported as IP traffic, the generated headers of the IP protocols can be massively large. RoHC compresses the RTP, UDP, and IP headers to reduce the size of the entire voice packets.

This decreases the radio resource utilization on the cell-edge and therefore improves the overall cell coverage on both UL and DL. In addition, it reduces the number of voice packet segments, which improves the BLER associated with these smaller-size transmissions. This improves the VoLTE end-to-end delays and jitter. On the LTE side, the most beneficiary of compression is at the moment VoLTE.

The RoHC function in LTE is part of Layer-2 at the user plane of the UE and eNB. Both UE and eNB behave as a compressor and de-compressor for the user-plane packets on UL and DL. The compression efficiency depends on the RoHC operating mode and the variations in the dynamic part of the packet headers at the application layer. A header can be compressed to one byte with RoHC, which efficiently reduces the voice packet size. RoHC in LTE operates in three modes: U-Mode, O-Mode, and R-Mode (Unidirectional, Bidirectional Optimistic and Bidirectional Reliable, respectively). The reliability of these modes and overheads used for transmitting feedback are different. In U-Mode, packets can only be sent from the compressor to the de-compressor, with no mandatory feedback channel. Compared with O-Mode and R-Mode, U-Mode has the lowest reliability but requires the minimum overhead for feedback. In O-Mode, the de-compressor can send feedback to indicate failed decompression or successful context update. Therefore, it provides higher reliability than U-Mode but it generates less feedback compared to R-Mode. In R-Mode, context synchronization between the compressor and de-compressor are ensured only by the feedback. That is, the compressor sends the context updating packets repeatedly until acknowledgment is received from the de-compressor. Therefore, R-Mode provides the highest reliability but generates the maximum overhead due to the mandatory acknowledgment.

RoHC is an extensible framework consisting of the following profiles. Note that the for support profiles 3 and 4 are not very common at present:

- RoHC Profile 0: provides a way to send packets without any compression.
- RoHC Profile 1: compresses packets with IP/UDP/RTP protocol headers.
- RoHC Profile 2: compresses packets with IP/UDP protocol headers.
- RoHC Profile 3: compresses packets with Encapsulating Security Payload (ESP) and IP protocol headers.
- RoHC Profile 4: compresses packets with IP.

RoHC has a significant gain to such traffic where the payload is small but the header size is bigger. From MediaTek in live network trials, the gain of compressing IPv4 headers for VoLTE traffic reached ~92% for UL and ~84% for the DL traffic. Therefore, the industry realized this gain and extended the concept of compression into additional traffic such as TCP/IP.

Therefore, 3GPP introduced industry-standard method to reduce the uplink packet size by standardizing UL-only RoHC in Rel-14. The method is expected to be applicable to several types of application and can be used as a way to achieve higher compression ratio among different file transfers, as shown in later section of the this paper. PDCP entities associated with Data Radio Bearer carrying user plane data (DRBs) can be configured by upper layers (RRC layer) to use header compression either bidirectional (if headerCompression is configured by RRC layer messages) or uplink-only (if uplinkOnlyHeaderCompression is configured by RRC layer messages). If uplinkOnlyHeaderCompression is configured by RRC, the UE shall process the received PDCP Control PDU for interspersed ROHC feedback packet corresponding to the uplink header compression, but shall not perform header decompression for the received PDCP Data PDU. UL-only RoHC has been added to cover RoHC Profile 6 which compresses packets with TCP/IP traffic. The UE indicates its support of the UL-only RoHC through supportedUplinkOnlyROHC-Profiles in the UE capability message sent by uplink RRC layer.

The reason for applying RoHC on UL rather than DL comes from the type of application used. Live network data shows the following gains in table 2 across different types of applications. The trial is performed with LTE-TDD, 20 MHz (theoretical DL throughput = 110 Mbps, UL throughput = 10 Mbps). The results are given by the KPI related to the overall compression gain in **Table 2.**

Table 2: MediaTek Test Results for UL and DL TCP/IP RoHC

Directions	Applications	Average Packet Size (Bytes)	TCP/IP header ratio (%)	Compression Gain* (%)
Downlink	video streaming	1422.77	3.66	3.18
	web browsing	1156.54	4.49	3.96
	instant messaging	1136.69	4.58	4.02
Uplink	video streaming	54.77	95.09	392.25
	web browsing	121.59	43.07	53.322
	instant messaging	92.90	56.19	86.392

**Compression Gain = [1-(Compressed Size/Original Size)]x100%*

The following points summarize the observation from the testing conducted for the data in table 2:

Observations on packet size distribution:

- DL: More than 70% packets are of size larger than 1400 bytes
- UL: More than 90% packets are smaller than 100 bytes

Observations on RoHC performance evaluation:

- UL RoHC (53%~392%) is more effective than DL RoHC (3%~4%)
- Compression gain increases as TCP/IP header ratio increases

As shown, the gains are observed mainly for the uplink with little or no benefit of compression on the downlink. The reason is that since RoHC cannot compress data payload, it is only useful if header is larger than payload. The table shows that the ratio of header in an uplink packet is higher than downlink, and hence the compression gain on the uplink is higher and more beneficial than the downlink. Therefore, introducing UL-only RoHC gives more simplicity to the overall design (e.g. minimal changes to the UE and Network complexity) and reduces the impact to power consumption of the devices. However, as shown in **Figure 1,** video streaming and instant messaging only account for 28% uplink traffic volume. Therefore, more than 70% traffics cannot be served well by UL-only RoHC. As a result, additional compression method are proposed in order to improve the compression gains for UL traffic.

Several proprietary solutions have been introduced by vendors, e.g. Data Acceleration, DACC feature. These methods focus on performing UL data compression between the UE and the radio access network. However, these algorithms have taken quite sometimes to be standardized in 3GPP in the last few years, which made it difficult to scale into a larger deployment, given the proprietary part of it. In brief, these algorithms target that the compression is performed only on the payload of the TCP, UDP, IPv4/v6 packets. Hence, the compression and decompression layers need to parse the TCP, UDP and IPv4/v6 headers to identify the start and end of the payload. The difficult also manifests itself with the type of application used. These features can compress original IP-level packets into compressed packets. The compressed packets are sent on the air interface in uplink and decompressed to reconstruct the original IP-level packets. IP-level traffic can consist of compressible data. The uplink can have repetitive (TCP ACKs) or textual data, which is a good target for compression. Other kinds of traffic, for example pictures or secure data connections, have little or no benefit of compression. Some data files, such as text files, picture files in the BMP format, and certain text style database files, can be compressed to a large extent. Other types of files are not compressed well. For example, most multimedia files, as they exist in a highly compressed state. These file types use efficient techniques to compress the data they contain. Files that are compressed cannot be compressed to any more significant extent.

As a result, 3GPP took a stand in Rel-15, and started to look into developing solid algorithms in order to unify the type of data compression and what it can achieve for different types of files and applications. This has finally become a standard feature, referred to as Uplink Data Compression, UDC, and applicable to LTE networks. In addition to different compression methods, the major difference between UL-RoHC and UDC is which portion of a data packet is compressed. RoHC compresses TCP/IP headers only and UDC compresses the whole UL packet. In next sections of this paper, we will study UDC algorithm in details and in the last section of the paper, we will show the performance enhancements coming from Rel-14 UL-only RoHC and Rel-15 UDC features.

UDC Standardization Effort in 3GPP

Several years back there already was an attempt to bring the UDC solution to the market. However that was a proprietary one and it did not manage do get wide adoption. The industry's requirement primarily pushed by the LTE-TDD carriers was to have a software feature on top of the modem protocol (without hardware impact) that would contain a compression method ensuring good compression performance, reduced interoperability efforts and wide acceptance between chipset and infrastructure vendors. This work started at RAN plenary #73 in September 2016, and was approved in RAN plenary #74 held in Dec. 2016.

It was proposed in this RAN plenary to study UDC and investigate potential standardized solutions and way forward. The proposal included: identify compression algorithms or compressed data formats for the purpose of performance evaluation, evaluate the performance of the identified compression solutions with compressed data formats based on traffic characteristics in a practical network, and evaluate the probability of decompression failure and the performance impact to the application layer traffic based on agreed traffic types and traffic modeling. The major question from that proposal was which compression method shall be selected.

Two candidates were DEFLATE and APDC. Table 3 summarizes the cons and pros for both methods.

Table 3: DEFLATE-ADPC Comparison

Performace	Complexity	Readiness	Interoperability Testing (IODT)	3GPP Compliance
DEFLATE > APDC	DEFLATE = APDC	DEFLATE > APDC	DEFLATE > APDC	DEFLATE = APDC
DEFLATE better in 6 out of 7 test cases DEFLATE efficiency 5% better with SIP than APDC Gains will be seen with e.g. VoLTE operation	DEFLATE can be used in low-end platforms No advantage with APDC	DEFLATE ready: RFC1951 APDC is not specified	DEFLATE known (RFC1951) and widely used today DEFLATE IODT will be expedited	Network upgrade required (RRC, PDCP) regardless whether DEFLATE or APDC is used

As seen from the table, DEFLATE is superior to or on par with APDC in all categories. This confirmed MediaTek push for this algorithm to become the only standard in 3GPP. Eventually RFC1951 based DEFLATE was approved by 3GPP as the sole compression method for UDC in the 3GPPRAN#77 plenary meeting in September 2017, under MediaTek lead. By the time when this white paper is published (June 2018) 3GPP meeting RAN#80 shall approve the final CRs for 3GPP TS36.300, TS36.306, TS36.323 and TS.36.331 specifications to finalize the standardization process. On the other hand, conformance test work items have been approved in RAN5#79 in May 2018. RAN5 will spend the following nine months to complete this work item. After March 2019, GCF can set its conformance criteria once RAN5 complete the work item. This schedule will open the way for the commercial deployment of the feature in chipsets and networks, and possibly be on time to reach to 5G implementation with minimal enhancements or new work items. **Figure 3** summarizes 3GPP timeline for UDC feature.

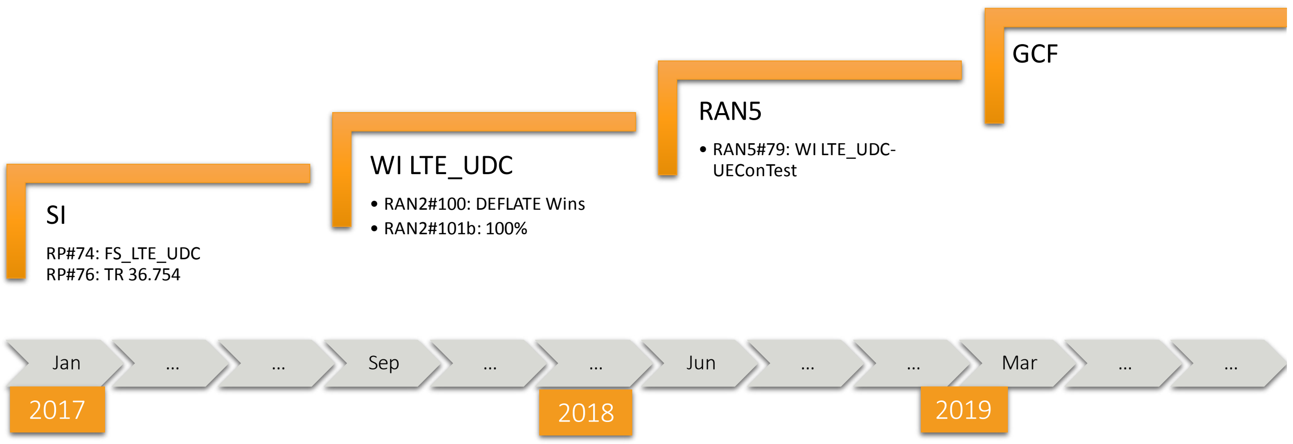

Figure 3: 3GPP Standardization Timeline for UDC

RFC1951 DEFLATE

3GPP UDC is based on RFC1951 DEFLATE data compression algorithm. It is designed to define a lossless compressed data format that:

- Is independent of CPU type, operating system, file system, and character set, and hence can be used for interchange;
- Can be produced or consumed, even for an arbitrarily long sequentially presented input data stream, using only an a priori bounded amount of intermediate storage, and hence can be used in data communications or similar structures;
- Compresses data with efficiency comparable to the best currently available general-purpose compression methods;
- Can be implemented readily in a manner not covered by patents, and hence can be practiced freely;
- Is compatible with the file format produced by the current widely used gzip utility, in that conforming decompressors will be able to read data produced by the existing gzip compressor.

DEFLATE is in fact a combination of LZ77 algorithm and Huffman coding. It is important to understand both to get the whole DEFLATE concept.

Huffman Coding

Huffman coding was first introduced by David Huffman in 1952 and since then is widely used in computer systems including such file formats as JPEG, MP3, etc. It is a form of prefix coding prepared by a special algorithm. Each code represents an element in a specific "alphabet" (such as the set of ASCII characters, which is the primary but of course not the only use of Huffman coding in DEFLATE).

A Huffman algorithm starts by assembling the elements of the "alphabet", each one being assigned a "weight" - a number that represents its relative frequency within the data to be compressed. These weights may be guessed at beforehand, or they may be measured exactly from the data passed through, or some combination of the two. In any case, the elements are selected two at a time, the elements with the lowest weights being chosen. The two elements are made to be leaf nodes of a node with two branches. For example there is a following set of elements and weights as depicted in Table 4.

D and E will be picked to make up the branches of a single node – D will be the “0” branch while E will make “1” **(Figure 4).** At this point the complete code is not defined yet but it is known that the codes for D and E will be exactly the same except for the last digit (“0” or “1” respectively). The combined node D-and-E is placed back with the other (as yet) uncombined elements, and given a weight equal to the sum of its leaf nodes: in this case, 8 + 8 = 16. Again, we take the two nodes with lowest weight, which are A, and D-and-E, and join them into a larger node **(Figure 5).**

Element	Weight
A	16
B	32
C	32
D	8
E	8

Table 4: Set of Elements and Weights

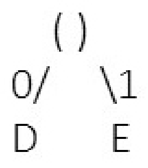

Figure 4: Node D-and-E

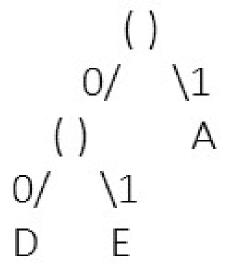

Figure 5: Node A-D-E

This time, when the node A-D-E is put back into the list of elements, all remaining elements have the same weight, 32. Which two of the three are selected to be combined first is not important, at least not in the classical Huffman algorithm. When all nodes have been recombined into a single "Huffman tree," then by starting at the root and selecting 0 or 1 at each step, you can reach any element in the tree. Each element now has a Huffman code, which is the sequence of 0's and 1's that represents that path through the tree. Now, it should be fairly easy to see how such a tree, and such a set of codes, could be used for compression. If compressing ordinary text, for example, probably more than half of the ASCII character set could be left out of the tree altogether. Frequently used characters, like “E” and “T” and “A”, will probably get much shorter codes, and even if some codes are actually made longer, they will be the ones that are used less often.

However, there is also the question: how do you pass the tree along with the encoded data? It turns out that there is a fairly simple way, if you modify slightly the algorithm used to generate the tree. In the classic Huffman algorithm, a single set of elements and weights could generate multiple trees. In the variation used by the DEFLATE standard, there are two additional rules: elements that have shorter codes are placed to the left of those with longer codes. (In our previous example, D and E wind up with the longest codes, and so they would be all the way to the right.) Among elements with codes of the same length, those that come first in the element set are placed to the left. (If D and E end up being the only elements with codes of that length, then D will get the 0 branch and E the 1 branch, as D comes before E). It turns out that when these two restrictions are placed upon the trees, there is at most one possible tree for every set of elements and their respective code lengths. The code lengths are all that we need to reconstruct the tree, and therefore all that we need to transmit.

LZ77 Compression Algorithm

LZ77 compression works by finding sequences of data that are repeated. The term ''sliding window'' is used; all it really means is that at any given point in the data, there is a record of what characters went before. A 32K sliding window means that the compressor (and decompressor) have a record of what the last 32768 (32 * 1024) characters were. When the next sequence of characters to be compressed is identical to one that can be found within the sliding window, the sequence of characters is replaced by two numbers: a distance, representing how far back into the window the sequence starts, and a length, representing the number of characters for which the sequence is identical.

Here is a simple example of highly compressible data:

Blah blah blah blah blah!

The data stream starts by receiving the following characters: “B”, “l”, “a”, “h”, “ “, and “b”. However, look at the next five characters. There is an exact match for those five characters in the characters that have already gone into the data stream, and it starts exactly five characters behind the point where we are now. This being the case, we can output special characters to the stream that represent a number for length L, and a number for distance D.

The data so far:

Blah blah b

The compressed form of the data so far:

Blah b[D=5,L=5]

The compression can still be increased, though to take full advantage of it requires a bit of cleverness on the part of the compressor. Look at the two strings that we decided were identical. Compare the character that follows each of them. In both cases, it is “l” - so we can make the length 6, and not just five. But if we continue checking, we find the next characters, and the next characters, and the next characters, are still identical - even if the so-called “previous” string is overlapping the string we're trying to represent in the compressed data. It turns out that the 18 characters that start at the second character are identical to the 18 characters that start at the seventh character. It is true that when we are decompressing, and read the length, distance pair that describes this relationship, we don't know what all those 18 characters will be yet - but if we put in place the ones that we know, we will know more, which will allow us to put down more - or, knowing that any length-and-distance pair where length is more than distance is going to be repeating (distance) characters again and again, we can set up the decompressor to do just that.

Blah b[D=5, L=18]!

Combination of Huffman Coding and LZ77 for 3GPP UDC

The DEFLATE compressor is given a great deal of flexibility as to how to compress the data. There are three modes of compression that the compressor has available:

- Not compressed at all. This is an intelligent choice for example for the data that's already been compressed. Data stored in this mode will expand slightly, but not by as much as it would if it were already compressed and one of the other compression methods was tried upon it.
- Compression, first with LZ77 and then with Huffman coding. The trees that are used to compress in this mode are defined by the DEFLATE specification itself, and so no extra space needs to be taken to store those trees.
- Compression, first with LZ77 and then with Huffman coding with trees that the compressor creates and stores along with the data.

The data is broken up in “blocks”, and each block uses a single mode of compression. If the compressor wants to switch from non-compressed storage to compression with the trees defined by the specification, or to compression with specified Huffman trees, or to compression with a different pair of Huffman trees, the current block must be ended and a new one must begin.

In case of UDC, Static Huffman coding is used as a DEFLATE compression strategy. A standard dictionary for SIP and SDP and one operator defined dictionary can be used as pre-defined dictionaries. The standard dictionary for SIP and SDP consists of the first 3468 bytes of the dictionary for SigComp defined in RFC 3485. When UDC is configured, at most one dictionary, configured by upper layers, is put into the tail of the compression buffer. Also, the compression buffer acts as a FIFO and hence the content of the dictionary is to be totally pushed out of the compression buffer after the size of transmitted uncompressed packets compressed by UDC exceeds the compression buffer size. If the size of dictionary is larger than the compression buffer size, only the tail of the dictionary is inserted in the compression buffer. UDC Data Block should be byte-aligned. Sync flush method is used as the DEFLATE byte-alignment. It performs 2 following tasks:

- If there is some buffered but not yet compressed data, then this data is compressed into one or several blocks
- A new block with empty contents is appended.

UDC Operation and Procedures

UL-only RoHC utilizes the same ideas already existing in UE and eNB, inherited from VoLTE RoHC, at the PDCP layer, while RRC layer is the side that is impacted with a new configuration change. UDC, on the other hand, is introduced in 3GPP with several changes to the protocol layers in both UE and eNB. In practice, the RRC and PDCP layer are impacted by UDC, where new signaling IEs are introduced to negotiate the data compression in addition to having new PDCP PDU types. This section describes the protocol layer changes and new entities introduced to handle UDC operation.

UDC Configuration in PDCP Layer

UDC is configured under PDCP layer parameters. Several PDCP entities may be defined for a UE, each entity carrying user plane data may be configured to use UDC or header compression. Each PDCP entity is carrying the data of one radio bearer. In the current version of 3GPP TS 36.323 specification, the robust header compression protocol (RoHC) and UDC, are supported. Every PDCP entity uses at most one RoHC or one UDC compressor instance and at most one RoHC or UDC decompressor instance. RoHC and UDC are not supported simultaneously for the same radio bearer. **Figure 6** represents the PDCP layer functional view with regard to UDC configuration point.

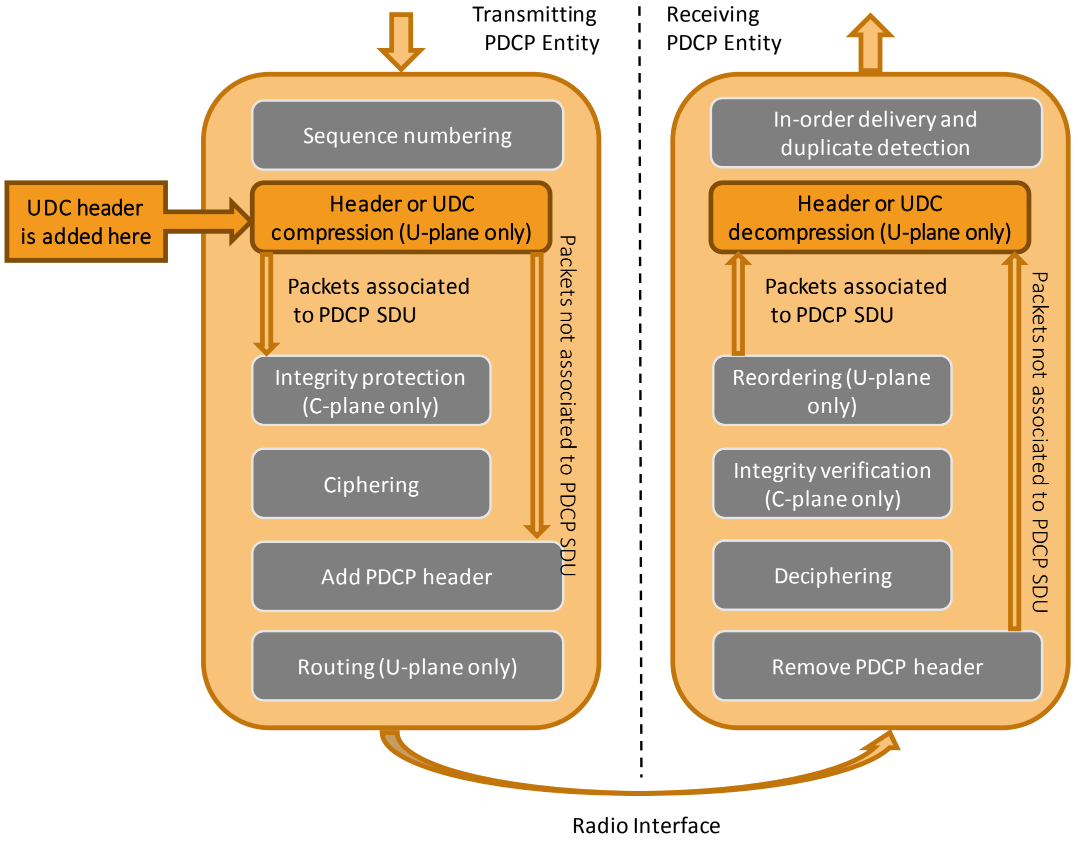
Transmitting PDCP Entity
Receiving PDCP Entity
UDC header is added here
Sequence numbering
Header or UDC compression (U-plane only)
Packets associated to PDCP SDU
Packets not associated to PDCP SDU
Integrity protection (C-plane only)
Ciphering
Add PDCP header
Routing (U-plane only)
In-order delivery and duplicate detection
Header or UDC decompression (U-plane only)
Packets associated to PDCP SDU
Packets not associated to PDCP SDU
Reordering (U-plane only)
Integrity verification (C-plane only)
Deciphering
Remove PDCP header
Radio Interface

Figure 6: PDCP Layer Functional View

The PDCP entities associated with DRBs can be configured by upper layers to use UDC. If UDC is configured, the UE shall apply UDC compression function to process the received PDCP SDU from upper layers corresponding to the configured DRB. If pre-defined dictionary is configured by upper layers, the UE shall prefill the configured pre-defined dictionary in the compression buffer upon configuration of UDC. If pre-defined dictionary is not configured by upper layers, UE shall set the compression buffer to all zeros. A UDC packet consists of a UDC header and a UDC data block.

UDC header (1 byte) contains the information about whether the current PDCP SDU is compressed by UDC or not represented by the FU bit where “0” stands for “not compressed” and “1” for “compressed”. Only the compressed packets are stored in the memory buffer. The UDC header also contains a reset bit FR to inform the decompressor that the compression buffer has been reset. The validation bits (checksum) of the compression buffer are also contained in UDC header. Checksum mechanism could be used to resolve miss-match (if any) between the compression and de-compression buffers.

A UDC data block contains either DEFLATE compressed blocks generated by UDC protocol or original PDCP SDU for SDU not compressed by UDC; the type is specified in FU bit in UDC header. The FR bit and Checksum field in UDC header are used only if FU bit is set to 1. User Plane PDCP data PDU structure can be found in **Figure 7** (example for PDU format for DRB using 12 bit SN).

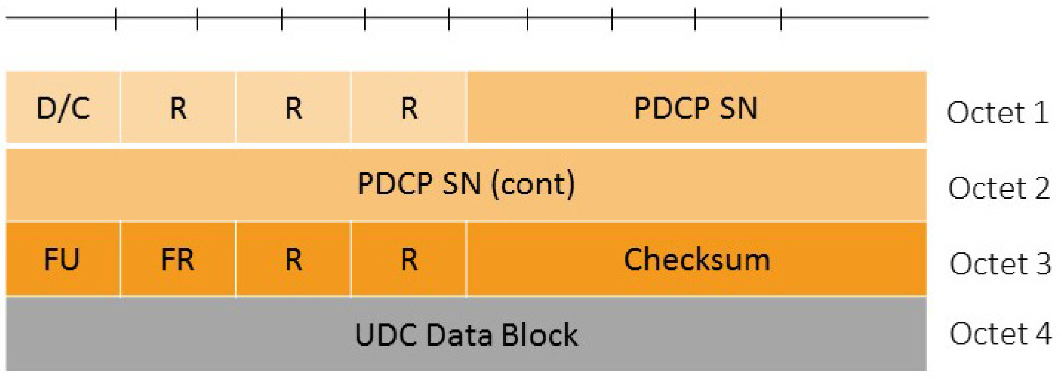

Figure 7: User Plane PDCP Data PDU for UDC (12 bit SN)

The UDC configuration summary is shown in **Table 5.**

Table 5: UDC Configuration Summary

Configurations	Settings
Compression strategy	“fixed Huffman-tree”
Byte-alignment option	“Sync_Flush” (RFC1979)
Compression flow	“optional UDC skipping” agreed
UDC buffer	2/4/8K, configured by eNB
Checksum in PDCP header	4-bit format
Pre-defined dictionary	- SigComp (RFC 3485) - One additional operator-defined dictionary (ODD) - UE Capability: associated PLMN ID + version ID

UDC Operation Flow in RRC Layer

The UE UDC capability is advertised in the RRC UE Capability Information message. For that matter 3 IEs are added to 3GPP TS 36.306 specification that are:

- SupportedUDC-r15 This field defines whether the UE supports the uplink data compression operation as specified in TS 36.323 A UE that supports the uplink data compression operation shall support 8192 bytes for compression buffer per UDC DRB and support up to 2 UDC DRBs.
- SupportedStandardDic-r15 This field defines whether the UE supports UL data compression with SIP static dictionary as defined in RFC 3485.
- SupportedOperatorDic-r15 This field defines whether the UE supports UL data compression with operator defined dictionary. If UE supports operator defined dictionary, the UE shall report versionOfDictionary, the version number of the dictionary, and associatedPLMN-ID, the associated PLMN ID of this operator defined dictionary as defined in TS36.331. This parameter is not required to be present if the UE is in VPLMN. In the current specification release, UE can only support one operator defined dictionary.

Once the network is aware that the UE is capable of UDC the eNB can configure it with the following new UDC related PDCP parameters within IE uplinkDataCompression-R15 of dedicated RRC signaling (PDCP-Config):

- BufferSize Indicates the buffer size applied for UDC specified in TS 36.323. Value kbyte2 means 2048 bytes, kbyte4 means 4096 bytes and so on. E-UTRAN does not reconfigure bufferSize for a DRB except for handover cases.

- Dictionary Indicates which pre-defined dictionary is used for UDC as specified in TS 36.323. The value standardForSIPandSDP means that UE shall prefill the buffer with standard dictionary for SIP and SDP defined in TS 36.323, and the value operator means that UE shall prefill the buffer with operator-defined dictionary.

The UDC configuration flow is represented at **Figure 8.**

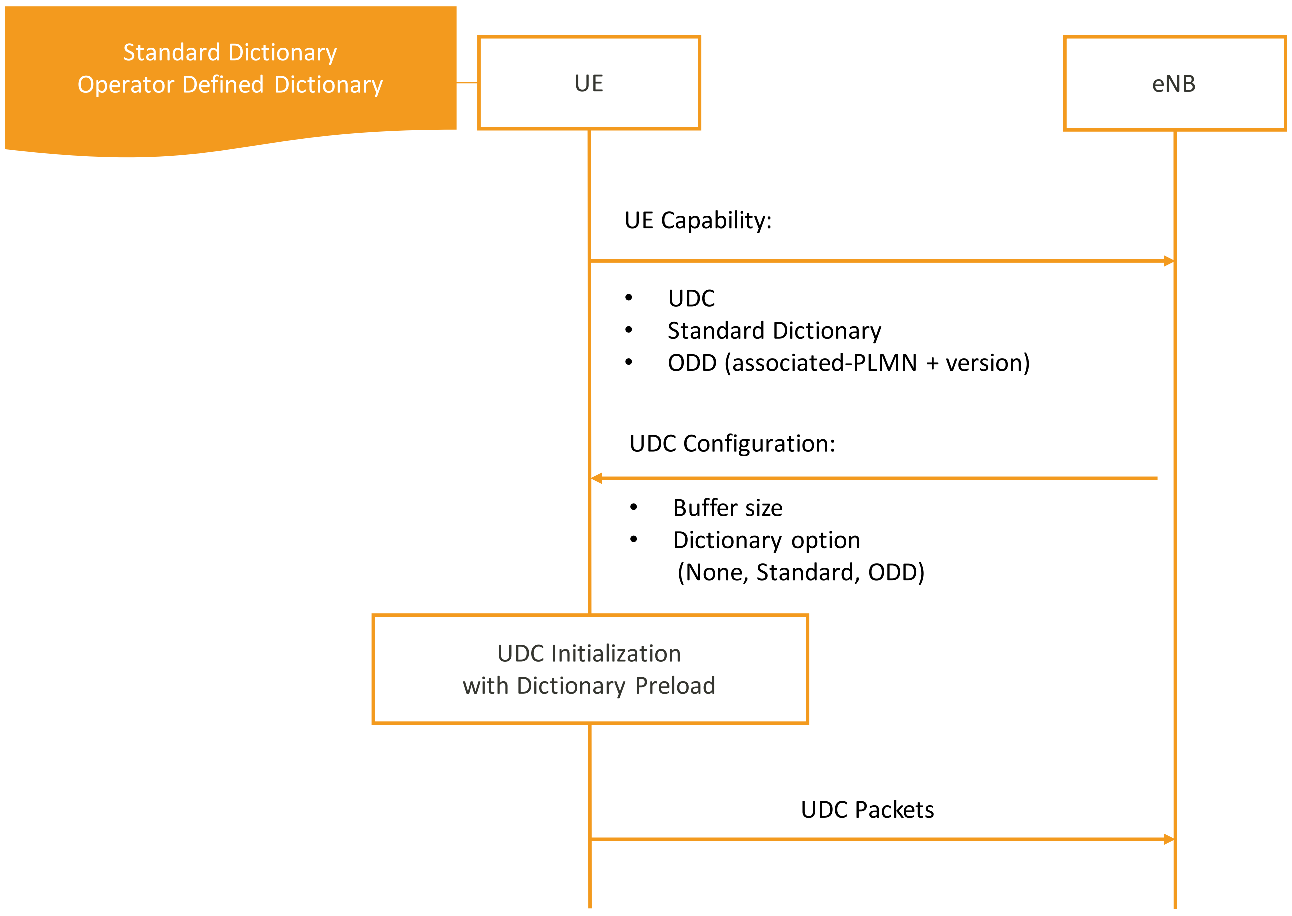

Figure 8: UDC Configuration Flow

Following restrictions must be taken into consideration for UDC deployment:

- UDC is used only in RLC Acknowledged Mode (AM),
- Maximum number of UDC DRB is 2,
- UDC is not compatible with RoHC thus only one of them can be applied at a time,
- UDC is not used on split or LWA DRB,
- Processing time is required for the compression thus certain thresholds will be applied by the UE depending on performance parameters. This can cause limitations of the maximum throughput of compressed traffic.

UDC is implicitly released and reset during handover, upon the initiation of RRC connection re-establishment and upon RRC connection resume. It will be re-configured though after HO/RRC re-establishment with all options described earlier (enabled/disabled, buffer size, dictionary).

Compression Error Handling

To detect and eliminate any possible errors in compression mechanisms, the error handling procedures were designed and described in specifications. They fall into 2 categories:

- Error handling triggered by the network.
- Error handling triggered by the UE.

In the first case the eNB will detect an error by checksum error. The checksum has a length of 4 bits. This field contains the validation bits for the compression buffer content: the checksum is calculated by the content of current compression buffer before the current packet is put into buffer.

The checksum is derived from the values of the first 4 bytes and the last 4 bytes in the compression buffer. The calculation is described as follows:

- Each byte is divided into two 4-bit numbers.
- The 16 4-bit numbers are added together to obtain a sum;
- The checksum is one's complement of the right-most 4 bits of the sum.

3GPP TS 36.323 contains an example of the checksum calculation:

The current UDC compression/decompression buffer has the following binary values for example:

Header<1,1,0,0,0,1,0,1,0,0,1,1,1,1,1,1,0,0,0,1,1,0,0,1,0,1,0,1,0,0,0,1,......,0,1,1,1,1,1,0,1,1,0,0,0,1,0,1,0,1,0,0,1,1,1,1,1,1,1,0,0,1,1,1,0,0> Tail

The sum of the first 4 bytes and the last 4 bytes can be calculated:

1100+0101+0011+1111+0001+1001+0101+0001+0111+1101+1000+1010+1001+1111+1001+1100 = 10000110;

And checksum value will be one's complement of the right-most 4 bits (i.e. 4 LSB) of the above sum. Hence checksum is 1001.

UDC checksum error notification PDCP control PDU indicates the compression buffer and de-compression buffer are out of synchronization. When receiving the notification, the UE shall trigger UDC buffer reset procedure to resynchonize the compression buffer. The error handling flow is shown in **Figure 9.**

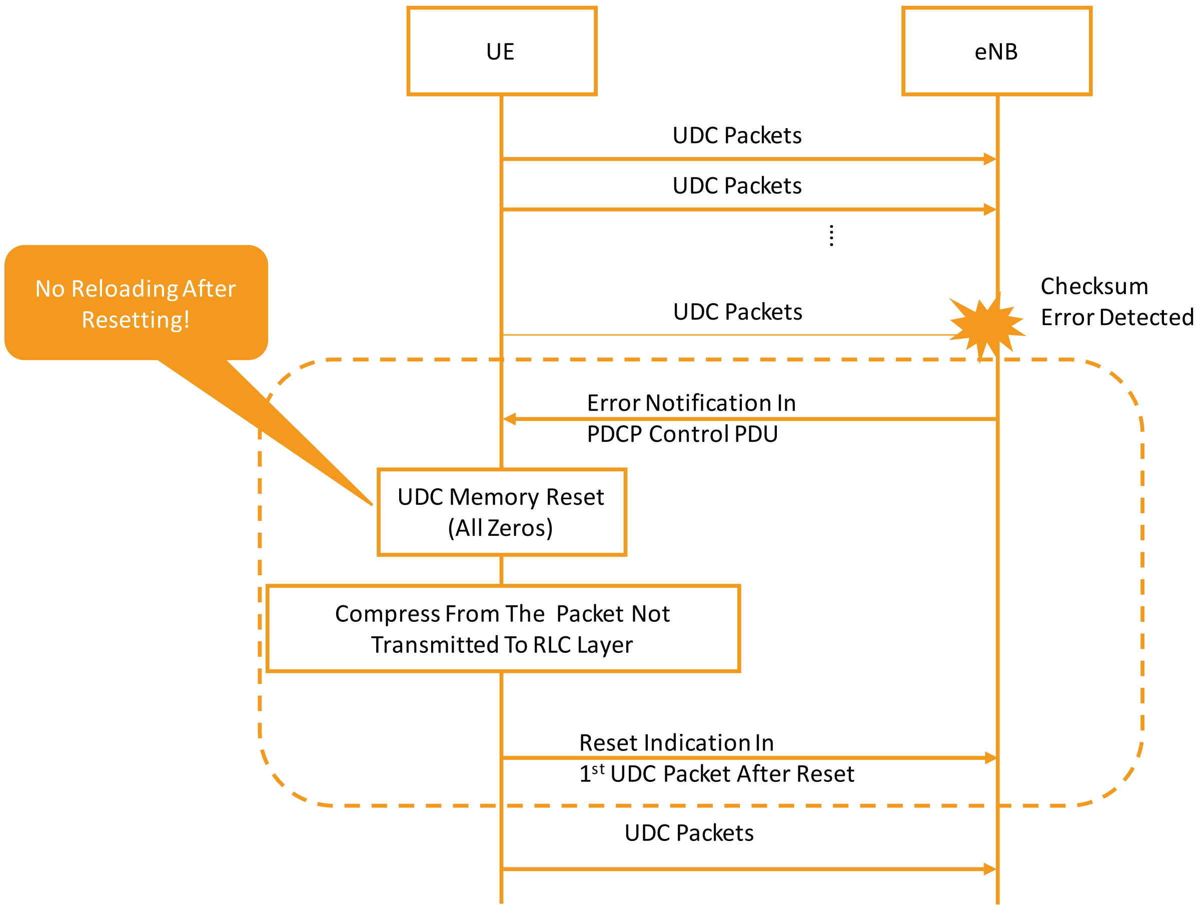
UE
eNB
UDC Packets
UDC Packets
⋮
UDC Packets
Checksum Error Detected
No Reloading After Resetting!
Error Notification In PDCP Control PDU
UDC Memory Reset (All Zeros)
Compress From The Packet Not Transmitted To RLC Layer
Reset Indication In 1st UDC Packet After Reset
UDC Packets

Figure 9: Network-Initiated Error Handling

The UE can detect an error on its own. For example it can encounter the situation when the compressed packet is bigger than the maximum PDCP SDU size. This can be the case when UDC gets a PDU with a size equal to maximum PDPC PDU size and after the compression procedure it exceeds the mentioned limit. When such things happen the UE will take a decision to reset its UDC buffer, further actions will follow the behavior from the network-initiated flow.

Performance Evaluation of UDC and UL RoHC

UDC gains per application were evaluated in the China Mobile practice tests in 3GPP RP-160042. These results are provided in **Figure 10.**

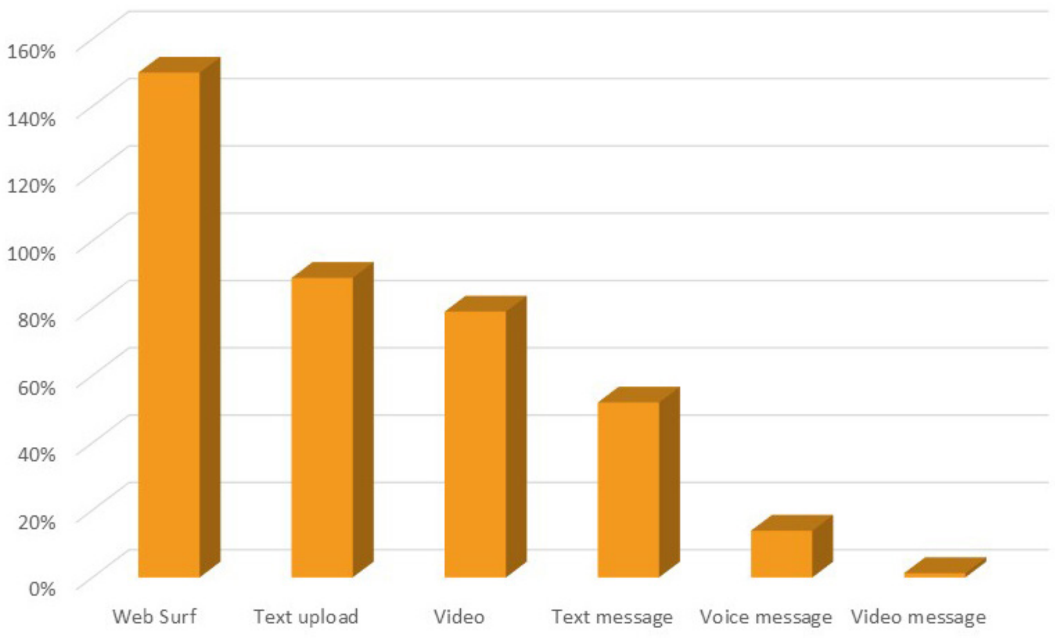

Figure 10: UDC Gains by Application

The UDC bid to improve the uplink performance is limited though due to several considerations:

- Only non-ciphered traffic can be compressed – this is the biggest limitation so far as the whole internet is moving towards HTTPS and other secure protocols. Evaluation of compressible traffic according to Qualys SSL Pulse as of June, 3 2018 can be found in **Figure 11.**
- VPN usage is popular in some markets making the traffic of such subscribers unsuitable for UDC.
- Many services use their own compression mechanisms at application level.
- Many operators are deploying their own traffic optimization systems and this is yet another compression opportunity at application level. Although this is much more relevant to the downlink traffic rather than the uplink.

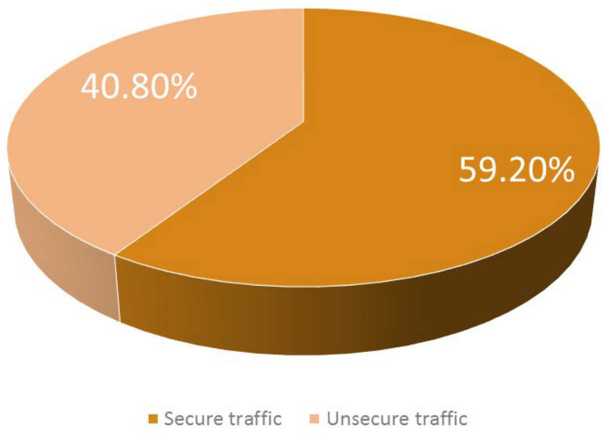

Figure 11: Share of Compressible Payload

Applications that are not very suitable for UDC can use UL-only RoHC instead. Although the compressed payload and headers together is of course saving much more resources, header-only compression also brings big value in some cases. For this purpose UL RoHC technique is more efficient than UDC. The idea to use RoHC in non-VoLTE applications was already approved by 3GPP in TEI14 as WI #680099.

3GPP TR 36.754 studied compression performance against nine scenarios for different uplink compression solutions. **Figure 12** illustrates the evaluation result for DEFLATE-based UDC and UL-only RoHC.

Not surprisingly, DEFLATE-based UDC outperforms RoHC for web browsing traffic, SIP, Mixed-IP flows scenarios, and performs generally good compression performance (~46% and above) elsewhere. On the other hand, UL-only RoHC has 20% better performance for video streaming and FTP.

As a result, UL-only RoHC can be mostly used for traffic where TCP/IP header ratio is very high, like online video streaming. For online video streaming, 94% traffic consists of TCP ACK without payload. For other applications like Text Chat or Web Browsing, DEFLATE-based UDC can be a better choice for compressing the uplink packets.

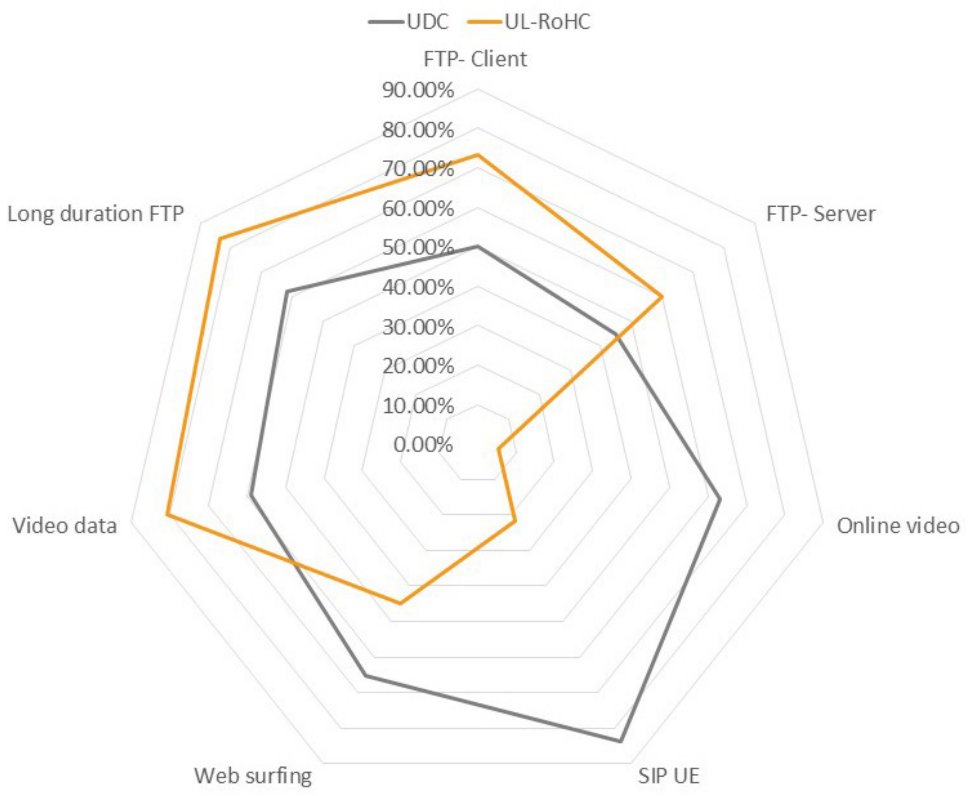

Figure 12: UDC vs. RoHC Compression Efficiency

Since UL-only RoHC cannot compress data payload, it is mostly useful if header is larger than payload. However, looking into a real network traffic distribution shown in **Figure 1,** video streaming and instant message only account for 28% uplink traffic volume. That is, more than 70% traffics may not be served well by RoHC deployment. As a result, DEFLATE-based UDC is required to serve the other 70% traffic. Therefore, any performance gain of the compression features is highly dependent on the traffic profile.

On the other hand, certain types of encrypted data files cannot be compressed very much. In practice, there are several ways of encryption techniques used in HTTP type of traffic: application layer encryption or transport layer encryption, and both are applied depending on the type of web application used. Encryption turns data into high-entropy data, usually indistinguishable from a random stream. Compression relies on patterns in order to gain any size reduction. Since encryption destroys such patterns, the compression algorithm would be unable to give much (if any) reduction in size if you apply it to encrypted data. Therefore, it is expected to have very minimal gains of compression to such traffic even if the compression is done at air interface (probably limited to the compression of GET Request and Response of the HTTP protocol only).

Conclusion

Efficient spectrum usage and utilization for various types of user application are getting more and more important for the future LTE deployments in order to cope with the increased data rate and capacity demands. This increases the need for supporting new features and methods such that spectrum, resources and latencies can be improved in order to enhance the system performance and user experience. The exponential increase in the demand for data connectivity can put disproportionate pressure on either uplink or downlink performance, depending on the usage of the Smartphones and applications. Users enjoy data-intensive streaming and social media services which are the most common practice for mobile internet. Therefore, network operators are shifting the focus into how to improve the end-user experience and the network capacity with as many different types of data and application usage as possible.

3GPP has developed several advanced and reliable mechanisms to improve the uplink performance for LTE networks. The mechanism use compression methods for headers and data payload in order to reduce the overhead on the air interface when uplink data is sent. As the standardization process will be completed soon, we are expecting network infrastructure vendors as well as chipset manufacturers to start implementing these compression features at a commercial level. This will allow to perform live trials and evaluate the benefits for the commercial deployments.

As illustrated in this paper, any performance gain of the compression features is highly dependent on the traffic profile. This might cause distinction of the gains observed in different markets. China for example tends to be less HTTPS-enabled while combined with a wide spread of LTE TDD networks make this country a primary target for deployment. On the other hand, Europe mainstream network deployment is LTE FDD and the protected traffic dominate the landscape there, which means the UDC gains will be more modest.

Nevertheless, data compression is an interesting feature that is expected to promptly gain potentials for future deployment in 4G and 5G. As studied in this paper, the gains of such compression algorithms are substantial and implementing the features to the data channels are expected to improve the end-user experience, with the following benefits:

- Increase uplink capacity when the amount of data sent in uplink is reduced.
- Increase the uplink and downlink data rates by sending less uplink data same amount of information (on downlink the benefits come from shorter TCP ACKs, needed for the same amount of downlink data).
- Improved uplink interference and coverage by reducing the uplink transmit power (Tx power) for the data channels when the amount of uplink data is reduced.
- Improved latencies by sending the same amount of information with less round trip time.

References

[1] 3GPP RP-161535 TSG RAN WG Meeting #73 Motivation on Introduction of UL Data Compression in LTE. China Academy of Telecommunication Technology

[2] Qualys SSL Pulse https://www.ssllabs.com/ssl-pulse/

[3] RFC1951 DEFLATE Compressed Data Format Specification version 1.3

[4] An Explanation of DEFLATE Algorithm. Antaeus Feldspar https://zlib.net/feldspar.html

[5] 3GPP TS 36.323 14.5.0 CR 0217 Rev. 2 Introduction of DEFLATE Based UDC Solution

[6] 3GPP TS 36.306 15.0.0 CR 1543 Rev. 2 Introduction of DEFLATE Based UDC Solution

[7] 3GPP TS 36.331 15.1.0 CR 3211 Rev. 2 Introduction of DEFLATE Based UDC Solution

[8] 3GPP TR 36.754 Study on Uplink (UL) data compression in LTE

[9] US20170257796A1 Selective Uplink Only Header Compression Mechanism, MediaTek

Abbreviations

AM	Acknowledged Mode
APDC	Adaptive Packet Data Compression
CA	Carrier Aggregation
D2D	Device-to-Device communication
D2N	Device-to-Network communication
DL	Downlink
DRB	Data Radio Bearer carrying user plane data
E-UTRAN	Evolved UMTS Terrestrial Radio Access Network
eNB	E-UTRAN Node B
IE	Information Element
IODT	Inter-Operability Development Testing
LWA	LTE-WLAN Aggregation
PDCP	Packet Data Convergence Protocol
PDU	Protocol Data Unit
R	Reserved
RLC	Radio Link Control
RoHC	Robust Header Compression
RRC	Radio Resource Control
RTP	Real-time Transfer Protocol
SDU	Service Data Unit
SN	Sequence Number
TCP	Transmission Control Protocol
UDC	Uplink Data Compression
UDP	User Datagram Protocol
UE	User Equipment
UL	Uplink

Aknowledgements

We wish to express our appreciation to our colleagues in MediaTek: Dustin Fan, Randy-YH Liu, Guillaume Sebire, YihShen Chen and MediaTek Carrier Engineering Services team, for their cooperative support and detailed studies on the different aspects of UDC and RoHC.

Authors:

Sergey Maximov

Since he joined MediaTek in 2017 Sergey has been working with the carriers in Russia and Eastern Europe to provide all support of MediaTek modem products in these markets and deliver any required features to the carriers of the region including emerging NB-IoT and 5G technologies. Prior to MediaTek, Sergey spent more than 14 years in operators' O&M and R&D departments working on core network development including an advanced role in IMS services deployment, RAN QoS model implementation, LTE rollouts and many more.

Mohamed A. El-saidny

Mohamed is a leading technical expert in wireless communication systems for modem chipsets and network design. He is leading Carrier Engineering Services Business Unit at MediaTek for Middle East/Africa/Europe, the department responsible for product business development and strategy alignment with network operators and direct customers. He is driving a team responsible for the technology evolution and the alignment of the network operators to the device and chipset roadmaps/products in 3G, 4G, Cellular Internet-of-Things and 5G. His main focus is on expanding MediaTek technologies and technical expertise with the mobile network operators worldwide. He has 15+ years of technical, analytical and business experience, with an international working experience. Mohamed received the B.Sc. degree in Computer Engineering and the M.Sc. degree in Electrical Engineering from the University of Alabama in Huntsville, USA. Prior to MediaTek, he worked at Qualcomm CDMA Technology, Inc. (QCT), San Diego, California, USA. He later transitioned to the network system design in Qualcomm's Corporate Engineering Services division in Dubai, UAE.