TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

HANNU ANTTILA

MEASUREMENTS AND ANALYSIS OF YOUTUBE TRAFFIC PROFILE AND ENERGY USAGE WITH LTE DRX MODE

Licentiate Thesis

TIIVISTELMÄ

TAMPERE UNIVERSITY OF TECHNOLOGY
Tieto- ja sähkötekniikan tiedekunta
ANTTILA, HANNU: YouTube-liikenteen mittaaminen, tutkiminen ja energiankulutuksen laskeminen LTE DRX:n kanssa
Lisensiaatintyö: 102 sivua, 9 liitesivua
Lokakuu 2016
Ohjaaja: Tekniikan tohtori Toni Levanen
Tarkastajat: Professori Mikko Valkama ja tekniikan tohtori Marko Helén
Avainsanat: YouTube, DRX, LTE, liikennemalli, energia

Videoiden lataaminen muodostaa suurimman osan Internetin liikenteestä, ja sen osuus on koko ajan kasvamassa. Toisaalta yhä suurempi osa Internetin liikenteestä siirretään matkapuhelinverkkojen kautta. Matkapuhelinverkkojen optimointi videosiirtoa varten voisi pienentää tarvittavaa taajuuskaistaa ja säästää puhelimen akkua. Tässä työssä tutkitaan YouTube-videoiden siirtoa ja etsitään liikennemallin avulla tietoa, jota voitaisiin käyttää siirtotehokkuuden parantamiseen. Painopisteenä on Long Term Evolution (LTE) -verkon Discontinuos Reception (DRX) -toiminta ja verkon siirtoajastin, joka lauetessaan siirtää puhelimen RRC_CONNECTED-tilasta RRC_IDLE-tilaan.

Työn alussa perehdytään aikaisempiin tutkimuksiin aiheesta, ja sen jälkeen esitellään mittausjärjestelyt. Mittaukset tehdään sekä paikallisverkossa että LTE-verkossa käyttämällä verkkoselaimeen perustuvaa YouTube-videon siirtoa. Mittausten ja tulosten tarkastelun jälkeen luodaan YouTube-siirrosta uusi *Matlab*-malli. Tämän liikennemallin avulla voidaan luoda YouTube-siirron kaltaista dataa testejä varten. Toinen *Matlab*-malli tehdään YouTube-siirron energiankulutusta varten. Sillä tutkitaan erityisesti verkon siirtoajastimen vaikutusta puhelimen energiankulutukseen LTE-verkossa.

Tutkimus osoittaa, että 97 % YouTube-siirrosta tapahtuu kahden rinnakkaisen Transmission Control Protocol (TCP) -yhteyden avulla. Siirron alussa on 10 sekuntia kestävä kiihdytysvaihe, jossa siirretään 20 % videosta. Sitä seuraa tasainen vaihe, jossa lähetykset ja lähetystauot vuorottelevat. Koko video on lähetetty, kun 74 % videon katseluajasta on kulunut. Katselun aikana siirretään myös useita kooltaan pienempiä TCP-yhteyksiä, jotka katkovat lähetystauot vain muutaman sekunnin mittaisiksi. Näitä pienempiä TCP-yhteyksiä viivästyttämällä saadaan aikaiseksi pidempiä lähetystaukoja ja parannetaan siten DRX:n hyödyntämismahdollisuuksia. Laskelmat osoittavat, että puhelimen energiankulutuksessa voidaan säästää jopa 30 % pienillä verkon siirtoajastimen arvoilla, kun TCP-yhteyksiä viivästytetään. Tutkimuksessa osoitetaan myös yleisesti siirtoajastimen merkitys puhelimen energiankulutukselle.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Faculty of Computing and Electrical Engineering
ANTTILA, HANNU: Measurements and analysis of the YouTube traffic profile and energy usage with LTE DRX
Licentiate Thesis: 102 pages, 9 Appendix pages
October 2016
Instructor: Doctor of Science Toni Levanen
Examiners: Professor Mikko Valkama and Doctor of Science Marko Helén
Keywords: YouTube, DRX, LTE, data profile, promotion timer, traffic model, energy

Video streaming forms a major part of the traffic on the Internet and its share of the traffic keeps increasing. On the other hand, more and more data is delivered in mobile networks. Optimizing a mobile network for video transmission could provide benefits of both decreasing the needed bandwidth and saving battery power in mobile equipment. In this thesis, YouTube data profile is examined to see if there are transmitting patterns which could be used for increasing transmission efficiency. The emphasis is on Discontinuous Reception (DRX) and on the promotion timer which is in control when a Mobile Station (MS) moves from the RRC_CONNECTED state to the RRC_IDLE state in Long Term Evolution (LTE) networks.

First, previous studies are explored and then a measurement setup is described. Measurements are done both in Local Area Network (LAN) and in LTE network using a YouTube implementation based on a web browser. After the measurements and the result analysis, a new *Matlab* model for YouTube data transmission is created. This traffic model can be used for simulating YouTube video transmission. Additionally, another *Matlab* model for YouTube energy calculations in LTE network is derived. This model is used to examine the energy usage in an MS and especially the effect of the promotion timer.

The studies indicate that 97 % of YouTube traffic is transmitted in two parallel Transmission Control Protocol (TCP) streams. There is a 10-second speedup phase where 20 % of the video is transmitted at the beginning of the transfer. The speedup phase is followed by a steady phase where idle and transmission periods alternate. The whole of the video data has been delivered when 74 % of the viewing time has elapsed. During the viewing, there are also dozens of small TCP streams that break idle periods into a few seconds. Delaying transmission of these small TCP streams gives a greater opportunity for longer idle periods and thus for DRX. It is calculated that delaying the small TCP streams can bring up to 30 % energy savings with small promotion timer values. Additionally, the importance of promotion timer values to the MS energy consumption is shown.

CONTENTS

ABBREVIATIONS

3GPP	3rd Generation Partnership Project
ADSL	Asymmetric Digital Subscriber Line
ARQ	Automatic Repeat-reQuest
BIDI	Bi-Directional, data transmitted in both directions like UL and DL
ECDF	Empirical Cumulative Distribution Function
DL	Downlink, data is transmitted from the server to the terminal
DRX	Discontinuous Reception
DTX	Discontinuous Transmission
eNB	E-UTRAN NodeB
E-UTRAN	Evolved Universal Terrestrial Radio Access Network
FDD	Frequency Division Duplexing
HD	High Definition
HTTP	Hypertext Transfer Protocol
HW	Hardware
IMS	IP Multimedia Subsystem
IP	Internet Protocol
IPv6	Internet Protocol version 6
kBytes	Kilo Bytes, 1024 bytes. This traditional definition is used here in all calculations
LAN	Local Area Network
LTE	Long Term Evolution
MAC	Medium Access Control
MBMS	Multimedia Broadcast/Multicast Service
MME	Mobility Management Entity
MS	Mobile Station
NAS	Non-Access Stratum
PDCCH	Physical Downlink Control Channel
PDCP	Packet Data Convergence Protocol
PHY	Physical Layer
QoS	Quality of Service
RLC	Radio Link Control
RMSE	Root-Mean-Square Error
RRC	Radio Resource Control
SDU	Service Data Unit
SI	System Information
STD	Standard Deviation
SW	Software
TCP	Transmission Control Protocol

TCP/IP	Transmission Control Protocol/Internet Protocol
TDD	Time Division Duplexing
TX	Transmitting / Transmitter
UDP	User Datagram Protocol
UE	User Equipment
UL	Uplink, data is transmitted from the terminal to the server
USB	Universal Serial Bus
WLAN	Wireless Local Area Network
WWW	World Wide Web

1 INTRODUCTION

Nowadays, video streaming dominates the Internet traffic. In 2012, video streaming covered 57 % of the whole traffic and it could be up to 69 % in 2017. Cisco has estimated that the growth rate of video streaming in fixed networks is 32 % and 90 % in mobile networks per year between 2012-2017 [1]. Similarly, International Telecommunication Union has estimated that there could be 10 billion smartphone subscriptions and the total of 13.8 billion mobile subscriptions in 2025. They also estimated strong growth in data traffic and especially in video traffic, the amount of which might be 4.2 times greater than that of non-video in 2025 [2], [3], [4].

During the first half of 2015 YouTube had 15.6 % and Netflix had 36.5 % share of the download traffic in the fixed line in North America. Surprisingly, in the upstream the winner was BitTorrent with 26.8 % share whereas YouTube and Netflix together only had 10.7 % share of the traffic. This shows that YouTube and video traffic generally is very downlink oriented. In Europe, the situation is slightly different in the fixed line: YouTube had 24.4 % share of the downstream traffic whereas the share of Netflix was only 4.8 % in 2015. The traditional Hypertext Transfer Protocol (HTTP) still takes 15.4 % share in the downstream in Europe. In the upstream BitTorrent dominates with 21.1 % share and YouTube is third with 7.5 % share of the total traffic. In Europe, the mobile traffic downstream winner was YouTube with 21.4 %, HTTP came second with 19.9 % and Facebook came third with 15.6 %. In North America, mobile users downloaded YouTube with 21.2 %, Facebook with 15.8 % and HTTP with 10.8 % [5].

The relative values presented above show the importance of YouTube and Netflix especially in the downstream Internet traffic. Their importance will probably grow in the future. Optimizing network algorithms and routers for video streaming can be useful in order to save network resources. The world is constantly moving towards mobility in data sharing, and the role of the mobile networks as the de facto Internet access is increasing. Especially with mobile networks, the problem is that most of the networks are bandwidth limited and every allocated resource should be fully utilized. This is important if energy efficient, very high bit rate networks are planned.

One way to save network resources is to use Discontinuous Reception (DRX). During DRX a network is sending nothing to a Mobile Station (MS) and an MS is receiving nothing. The MS can save battery power and the network can use the resources for other MSs. This is particularly significant for the MS where the battery power is limited. DRX can be used when there are pauses in data transmission patterns. A side effect of DRX is that packet transmissions may be delayed and this can affect user experience.

Additionally, the real implementation of the network and MS is more complex because both parties must take DRX into account.

In this thesis, YouTube data transmission profile is studied to see if there are patterns in YouTube traffic which could be used for optimizing network parameters and DRX parameters in particular. The result is a new traffic model for YouTube traffic. As a special case, the effect for LTE and for LTE DRX is briefly examined, but the general results are not targeted only for LTE networks. More detailed attention is paid to the promotion timer in LTE network and new results about timer values and energy usage with YouTube traffic shaping are presented.

This thesis is organized as follows. Chapter two presents general video transmission principles and provides an overview of some previous studies. Chapter three describes a DRX feature in general and also DRX as used in LTE. Chapter four approaches YouTube traffic profiles in Local Area Network (LAN) and Chapter five in mobile networks with emphasis on LTE networks. Chapter six introduces a new YouTube traffic model derived from the measurements and analysis discussed in the previous chapters. Chapter seven combines the YouTube traffic model and the LTE DRX model and analyzes the effect of DRX and an LTE network promotion timer on MS energy efficiency. Also, traffic shaping for YouTube data is done and the results are presented. Finally, chapter eight sums up the final conclusions.

2 YOUTUBE VIDEO TRANSMISSION AND EARLIER STUDIES

This chapter briefly describes video transmission in general and presents earlier studies about YouTube video transmission data profiles.

2.1 Video transmission in general

Video transmission and viewing can be divided into two different methods based on their nature: real-time and non-real time video viewing. Real time video viewing imposes high requirements for video latency and transmission systems. Pauses and disruptions in transmission are easily visible to a viewer. Requirements for transmission systems are very similar to real time voice systems and perhaps the best example can be found from videotelephony systems. In videotelephony not only voice but also video is transmitted between two or several parties and conversation is possible between the attendants. Real-time quality of services parameters have been defined for modern radio networks to ensure low latency and thus high quality for videotelephony. These parameters are e.g. resource type (guaranteed bit rate or non-guaranteed bit rate), priority, packet delay budget and packet error loss rate in LTE networks [6]. As an example, videotelephony requires lower packet delay budget than normal Word Wide Web (WWW) traffic.

On the other hand, non-real time video systems are easier to implement and video can be transmitted without emphasizing latency and in most cases, the traditional best-effort class used in IP networks is enough to guarantee satisfactory viewing experience. Normally, these systems use one-way video traffic from the sender to the viewer and the viewer has the possibility of controlling viewing. Video is typically buffered to avoid interruptions in case of errors in transmission medium, so latency or guaranteed bit rate is not so important. Examples of non-real time video transmission systems are YouTube, Netflix and other services which allow users just to look at the videos on demand.

2.2 Studies about YouTube video data profile

Ameigeiras et al. analyzed the basics of YouTube traffic in [7]. They claimed that YouTube used Flash Video as the default media format (92 % of traffic) for non-High Definition (HD) video clips and their study concentrated on this traffic. They characterized how YouTube servers downloaded data to users. Their test setup consisted of regu-

lar university network, *Wireshark* protocol analyzer [8], a playback monitor and a clip surveyor; the last two were developed by themselves.

Ameigeiras et al. [7] showed that traffic generation rate of the media server depended on the video encoding rate and the basis of their study was to examine the amount of the accumulated data in the player end. Based on the change of accumulated data, they identified that YouTube transmission strategy consisted of 2 phases: initial burst phase and followed by a so called throttling phase. They got the following cumulative probability distribution for initial burst length for video data measured in seconds as shown in Table 1. [7] (Video data viewing length can be longer than actual time used for data transmission). The actual transmitted data amount is the size of the initial video burst in seconds multiplied by the video encoding rate.

Table 1: Cumulative probability density function of the initial burst size [7]

Size of initial video burst (s)	37	38	39	40	41	42	43	44	45	46	47
Cumulative probability (%)	1.2	1.2	3.6	69.9	89.2	91.6	94	97.6	97.6	98.8	100

A clear reason why the initial burst size was close to 40 s is unknown, but it can be guessed that based on the user behaviour studies most users watch the movie clip less than 40 seconds before deciding whether to continue the clip or to move on to the next clip. The initial burst also provides sufficient buffer for short interruptions in the connection without causing unwanted pauses in the movie playback. It should be noted that the actual initial burst and throttling behaviour also depends on the operation system used [9]. This will be discussed in more detail in the latter part of this chapter.

The throttling phase started after the initial burst. During the throttling phase data was not sent continuously but in short bursts which Ameigeiras et al. [7] referred to as ‘a chunk’. In their opinion, Transmission Control Protocol/Internet Protocol (TCP/IP) packets belonged to the same chunk when the time difference between TCP/IP packets was less than 200 ms. Between the chunks there were periods when data was not sent at all. The media server controlled traffic generation rate with a so called throttle-factor which was 1.25 with YouTube. The information rate is throttle-factor multiplied by video clip encoding rate. They claimed that the chunk size is almost always exactly 64 kBytes and the period between chunks was approximately 64 kBytes/(1.25·V_r), where V_r is video encoding rate. This kind of throttling saves bandwidth for files which might not be played to the end, because not all of the video data is sent immediately to the receiver. So if viewing is stopped, it might be that last chunks are never sent from the video server. According to Finamore et al. [10], only 10 % of YouTube videos are viewed longer than 50 % of the actual video duration.

Additionally, Ameigeiras et al. [7] studied what happens if there is network congestion during a video clip downloading. It seems that the video server always tried to send data with a constant throttling speed and if the congestion only lasted for a short period

of time, it was not visible to the end-user. However, if the congestion lasted long enough, the buffer finally drained out and there was a break in video viewing and the end-user had to wait for new data to arrive. Finally, they provided a YouTube server traffic generation model which can be used for simulating YouTube traffic.

Ramos-Munoz et al. [9] - actually, the same team that wrote [7] - also studied mobile YouTube traffic. They used Android IOS and Apple mobile stations in a 3G network. They tried to find out the characteristics of YouTube traffic when used over mobile network and compared the results with wired line studies (like [7]). Tests were performed in the early 2013 and they used packet sniffers installed both in Android and Apple mobile stations. During testing they used the native YouTube application in a mobile station to download videos and not web browsers. Their tests used three different kind of mobiles: Apple, Android-M (middle priced) and Android-H (high priced) models. Their study showed that in Apple video between 1-12 TCP connections (1 of them being most used: 66.8 %) were downloaded. Android-H also used several TCP connections and Android-M only used one TCP connection.

Ramoz-Munoz et al. [9] discovered that a throttling factor equalled to 2.0 for video encoding rates higher than 200 kb/s and that the chunk size was 64 kBytes. Thus, the chunk size was the same as for wired networks but the throttling factor differed. In Android-H they observed the terminal sent TCP RESET when the amount of data in the buffer was close to 100 s and data transmission was paused. After this, application asked the server to send more data when there was no more than 40 s of video left. For Android-M they noticed that TCP window was used to control the amount of data. For Apple they claimed the results being the same as in wired networks presented in [7] and TCP window control was not noticed.

Rao et al. [11] studied both Netflix and YouTube characteristics. They did the measurements both in wired and in WLAN (Wireless Local Area Network) networks using Apple IOS and Android operating systems. They used different data for videos: HTML5, Flash and Microsoft Silverlight. As a result they found three different streaming strategies depending on the browser, application and data set:

1. No ON-OFF cycles: all data was transferred as fast as possible
2. Short ON-OFF cycles: there were small periods (2-4 seconds according to the figures) when data was not transferred. According to their definition the transmitted block size was less than 2.5 Mbytes
3. Long ON-OFF cycles: there were larger periods (even 60 seconds) when data was not transferred. The transmitted block size was larger than 2.5 Mbytes.

They used Internet Explorer, Google Chrome and Firefox browsers and in terms of YouTube they compared Flash, HTML5 and HD videos. They used *tcpdump* and *windump* programs to capture traffic in PC. In mobiles (Android and iPhone) they used native mobile applications. They only captured the first 180 seconds of traffic and did the measurements in 4 different locations:

1. 100 Mbps wired network connected to the Internet with 500 Mbps
2. WLAN with typical 7.7 Mbps download and 1.2 Mbps upload rate

3. 100 Mbps wired network connected to the Internet through 1 Gbps link
4. Wired network with a cable modem, which had typical 20 Mbps in downlink and 3 Mbps in uplink performance

The used networks were located in France and in the United States of America. The results showed that for Flash videos YouTube used Short ON-OFF cycles whereas HTML5 used short cycles, no cycles or long cycles depending on the browser. In addition, according to the results, YouTube sent approximately 40 seconds of video data during the start in the buffering phase. The researchers defined the end of buffering phase when there was the first OFF period in the traffic.

For the steady-state transfer after the buffering phase Rao et al. [11] observed that YouTube servers sent data periodically in 64 kBytes blocks. They observed larger than 64 kBytes blocks only when the retransmissions caused several 64 kBytes blocks to merge. They also noticed that Google Chrome sometimes used long ON-OFF periods with YouTube. OFF periods were typically in the order of 60 seconds. They found out that during the buffering phase Chrome typically downloaded 10-15 MB of data.

Supposedly, Rao et al. [11] assumed that all the YouTube video data is transmitted in a single TCP session. This can be concluded from their figures and text where they discussed the TCP transmission and reception window size. Because every single TCP session has its own window, several TCP sessions would mean several TCP windows, one for each of the TCP sessions.

Prados-Garzon et al. [12] simulated YouTube traffic over LTE network, which they called as '3G Long Term Evolution'. Some of the researchers were the same as in [7]. They evaluated the performance of YouTube service for Flash videos downloaded from Personal Computer over the LTE network. First they analyzed TCP traffic traces from YouTube streaming servers. They used 10 Flash video for the traces. For YouTube traffic generation they used the model presented in [7]. Naturally, they also had a model for LTE's E-UTRAN NodeB (eNB) simulation. The results were as follows:

1. The throughput reached by an UE is limited by the server traffic generation rate during the throttling phase. Thus, in the most cases the UE did not use the maximum data rate achievable in the LTE interface
2. Most of the TCP packet losses occurred during the initial burst due to the TCP adaptation. These packet losses were independent of the radio link quality, because losses were mainly caused by TCP. The packet loss depended on the link quality during the throttling phase and the loss was greater in poor radio link conditions.
3. The probability of suffering pauses in viewing increased with higher load of the cell, especially in poor radio link conditions. The more there were users the less there was bandwidth for a single user.
4. The number of pauses experimented by the users during video downloads were heavily influenced by the cell load because bandwidth available per user was reduced. The same applied for pause duration, but the load of the cell had less impact on pause duration than for the number of pauses.

If the server traffic generation rate had been higher, the available LTE bandwidth would have been better utilised. On the other hand, this might have led to more pauses during the viewing because it increases the load of the cell. Packet losses during the initial burst appeared partly because the nature of the TCP and it can be difficult to improve this on radio link level, but proper TCP parameters could help. Not surprisingly, poor radio link conditions with high load in a cell gave the worst user experience in the form of pauses in video viewing. But it also seems that the pause duration did not grow in the same way as the number of the pauses during high cell load. This can depend on the used LTE eNB model and especially on the way the model allocates resources to different users. It looks like the model used here wanted to give short radio resources yet frequently, which means that during the simulations considerable number of pauses was observed, but their length was not growing. So the eNB scheduler can have a significant role for a user experience. If the eNB scheduler had precise knowledge of the traffic patterns, it could use this information in the scheduling decisions. Now the 3GPP standard [6] differentiates video traffic services in very high level, and it does not take account different data profiles, which can exist inside the same service category. As was already seen, even the same provider, like YouTube, can provide the same video transmission in several different ways depending on the used equipment. On the other hand, the network equipment vendors have quite free hands to develop their scheduling algorithms for eNB, because 3GPP specification sets only boundaries for optimization ideas.

Li et al. [13] studied how entropy theory could be used to predict traffic dynamically in cellular networks. They used a network with 7000 Base Stations (BS) to collect data in the cells. If the radio network controller knew what kind of traffic is expected next, it could change the network parameters accordingly and route the traffic in an optimal way. They showed that traffic prediction is feasible both theoretically and practically. Their study did not conclude how much prediction would benefit a network, but then again, showed some examples of how prediction could be used.

Multimedia traffic model for videos was introduced in [14]. The model uses Poisson process to generate data and is intended for modelling multimedia traffic in the IP Multimedia Subsystem (IMS). Baugh et al. [15] defined the similar Poisson based model for 3GPP standardisation for Medium Access Control (MAC) and Physical Layer (PHY) performance metrics calculations. Both models are based on a Star Wars movie capture and assume constant data transmission rate during the viewing.

Tanwir et al. [16] classified and studied several VBR video traffic models. They divided each model to five different groups: Autoregressive models, models based on Markov processes, Self-similar and fractional ARIMA models, Wavelet models and other approaches. They presented the features of all of these groups. For example, the Autoregressive models are based on autocorrelation of the video. All the groups and models are based on statistics and expect data rate to be the same as the viewing rate. The models do not include transmission characteristics but only video codec output.

2.3 Targets in this thesis

In this thesis, it was studied if the results mentioned above concerning YouTube data profile and models are still valid. The emphasis was on verifying patterns in video transmissions, especially behaviour in the initial and throttling phase with 64-kByte blocks, which were described in [7], [9] and [11]. The results could then be used with DRX studies. The measurements and results of YouTube data profile are presented in Chapters 4, 5 and 6.

3 DRX IN MOBILE NETWORKS

This chapter gives an overview of DRX. As a special case DRX in 3GPP (3rd Generation Partnership Project) LTE network is introduced briefly. Finally, some other scientific studies about LTE DRX are discussed.

3.1 DRX in general

Several modern mobile networks enable having very high bit rates for a single MS. A release 8 LTE terminal with four antennas can achieve 300 Mbits/s in downlink as maximum. Nowadays, a common category 3 LTE MS can reach 100 Mbps in DL and 50 Mbps in UL. The next category 4 increases a DL throughput to 150 Mbps, e.g. the popular Samsung Galaxy S4 LTE is an MS capable of category 4 [17]. Furthermore, these high rates mean a high current consumption in baseband and RF circuitry of the MS, where values over 1.6 Watts have been measured with early LTE implementations [18]. The high current consumption means high battery drainage and high heat emission, as well. With DRX an MS can switch off the RF circuitry and parts of the baseband when there are breaks in the transmissions and receptions. This saves battery power and additionally cools down the MS. DRX also saves network resources, because network does not have to reserve radio resources for the MS during the DRX period. Although the benefits of DRX are clear with high speed networks, it is not a new invention. Already 2G GSM voice MSs used DRX to save battery power.

DRX is always a trade-off between power consumption and delay. During DRX an MS cannot receive any data, not signalling nor paging information from the network to start data reception. The longer the DRX period is the higher possibility there is that some signalling or data packets are delayed. This means that on the network side there must be buffering capacity to store the data that is coming during the DRX period. Additionally, because the MS can move, it must have a possibility to measure network conditions and parameters in certain intervals. Otherwise, it could happen that the MS moves out of the cell range and drops from the service. The faster the MS can move the shorter the DRX period should be. In addition, depending on the network standard, an outgoing packet in UL can interrupt the DRX period. This happens e.g. in 3GPP LTE networks in Frequency Division Duplexing (FDD) mode. Figure 1 shows DRX functionality in general. The upper part of the Figure 1 shows the buffers of the network and in the lower part the Receiver (RX) functionality of the MS can be seen. The first transmission may be sent directly to the MS but the second transmission needs buffering, because the MS is in the DRX mode and cannot receive data when it arrives. RX power can be switched off during the DRX period and it is switched on only during data

reception or when the MS has to listen to possible paging messages, monitor cell information or undergo other measurements.

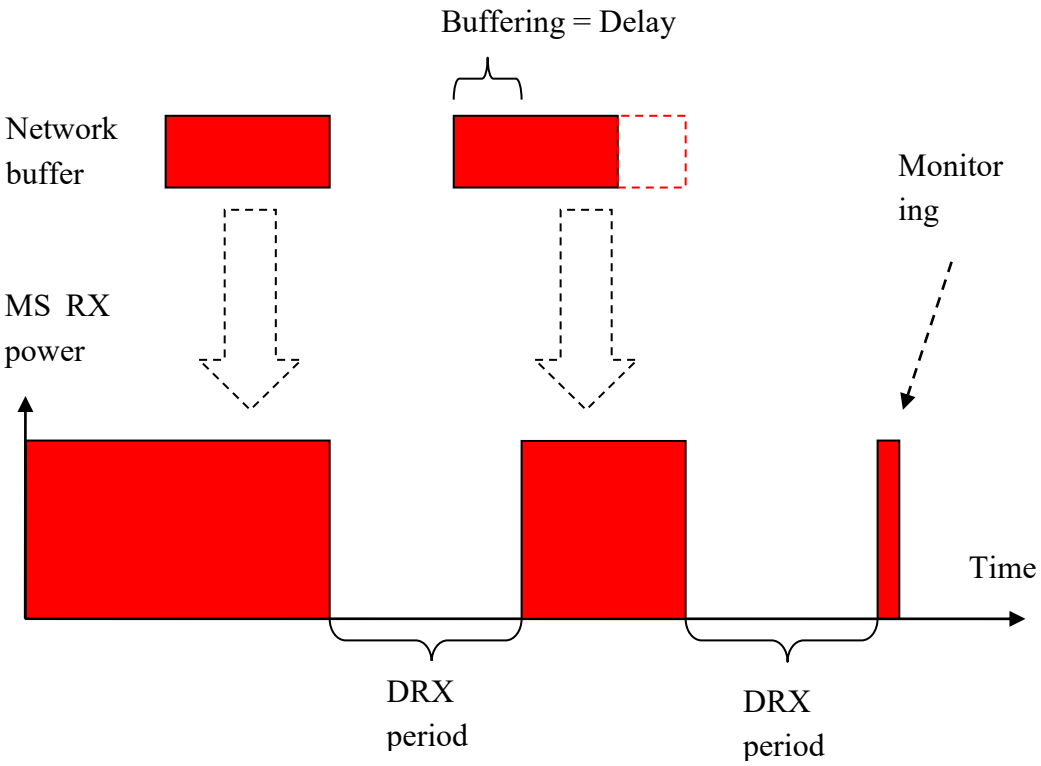

Figure 1: DRX in general

Discontinuous Transmission (DTX) is the same thing for UL as DRX is for downlink. In DTX, a device has pauses in transmissions and the network recognizes that no data is incoming during the DTX period. There is no DTX mode in the 3GPP LTE FDD system.

3.2 DRX in LTE

DRX functionality for LTE in general level has been specified in 3GPP Stage 2 standard [19] and a more detailed explanation of the DRX can be found in the same standard's MAC layer specification [20]. The LTE control plane protocol stack is shown in Figure 2. The control plane protocol stack consists of Non-Access Stratum (NAS), Radio Resource Control (RRC), Packet Data Convergence Protocol (PDCP), Radio Link Control (RLC), MAC and PHY protocol layers. NAS signalling is done between an MS and a Mobility Management Entity (MME) while the rest of the signalling is between an MS and an eNB. The LTE data plane stack is very similar except NAS and RRC layers are missing and in higher level either the TCP or the User Datagram Protocol (UDP) and the IP protocol are used between the MS and the remote host.

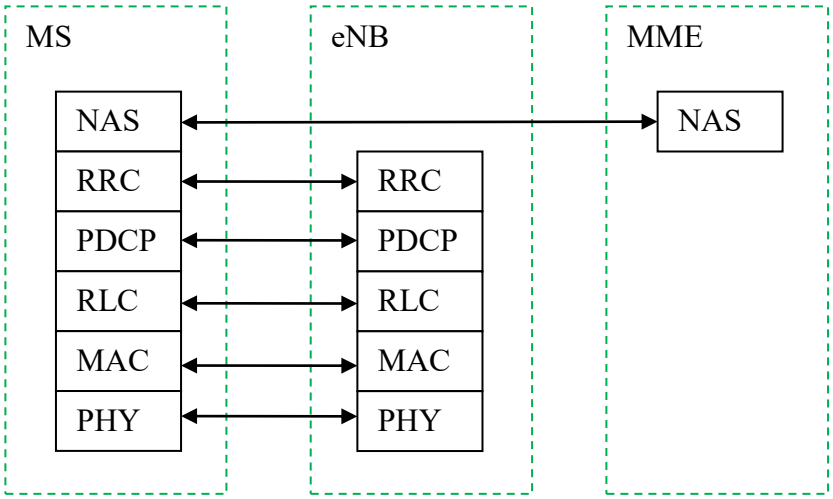

Figure 2: LTE control plane protocol stack [19]

First of all, to understand LTE DRX functionality, it is good to understand the basics of RRC in LTE. RRC sublayer is part of the LTE network control plane and it controls high level operations between an MS and an eNB such as the following:

- Broadcasting System Information (SI)
- Paging
- Establishment, maintenance and release of an RRC connection between an MS and an Evolved Universal Terrestrial Radio Access Network (E-UTRAN)
- Security functions
- Establishment, configuration, maintenance and release of point-to-point Radio Bearers
- Mobility functions
- Notification of Multimedia Broadcast/Multicast Service (MBMS)
- Quality of Service (QoS) management functions
- Reporting and control of the measurement reporting of an MS
- NAS direct message transfer to/from NAS from/to an MS

The RRC controls DRX operation by configuring the timers on the MAC layer [20]. These timers are listed in Table 2. In the first column, there is the name of the timer. The second column presents an explanation for the timer and, the third column indicates the maximum timer values. The timer values are defined in 3GPP RRC specification [21] as subframe lengths and one subframe lasts 1 ms. It is up to the RRC in the network whether the short DRX is configured or not. When a short DRX is configured, an MS first uses a short DRX cycle before it starts using a long DRX cycle. The short DRX is to reduce MS wakeup time in case of unexpected data arrival immediately after DRX is enabled [22]. A network can command an MS to start a DRX operation immediately when needed.

Table 2: Timers for DRX in LTE

Timer name	Purpose	Maximum timer value
onDurationTimer	To define how long MS is active during DRX cycle to receive paging messages	200 ms
drx-inactivity Timer	To specify time after MS starts DRX; timer is restarted if MS receives something	2560 ms
drx-RetransmissionTimer	To indicate the number of consecutive PDCCH (Physical Downlink Control Channel) subframes which MS will listen if retransmission is expected; if retransmissions are expected, the time (onDurationTimer) of MS being active increases	33 ms
longDRX-Cycle	To indicate cycle period of long DRX; includes both active and inactive time	2560 ms
drxStartOffset	To specify the subframe when DRX cycle starts after DRX is active; in case of long DRX this is the same as longDRX-Cycle	2560 ms
drxShortCycleTimer	To determine time after Long DRX cycle is started; this is expressed as multiples of shortDRX-cycle	16
shortDRX-Cycle	To determine cycle period of short DRX; includes both active and inactive time	640 ms

RRC has two main states to control the functionality of the MS: RRC_IDLE and RRC_CONNECTED [19]. These states are seen in Figure 3. These states are briefly explained because they affect heavily the MS's current consumption and radio activity.

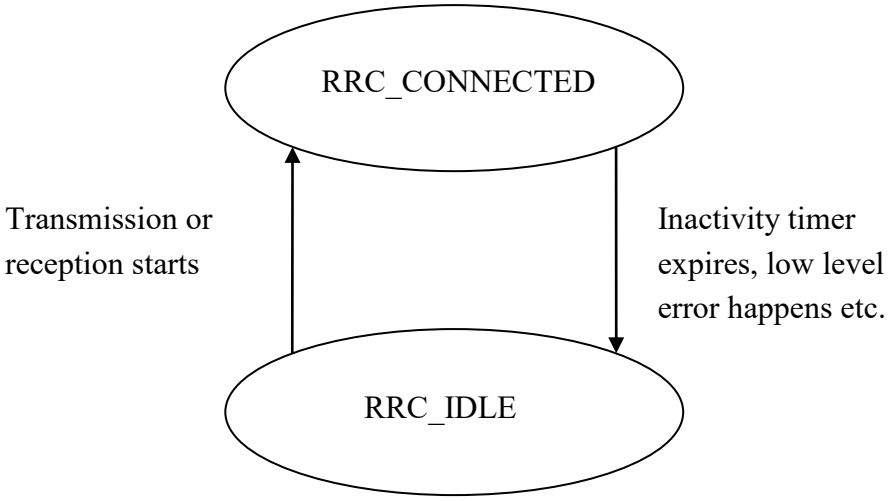

Figure 3: RRC states in LTE

During the RRC_IDLE state the network knows the MS location only at a tracking area level, which can include several cells. Cell reselection decisions are made by the MS, which listens to the paging messages occasionally. During the RRC_IDLE state data transmission is not possible and moving from the RRC_IDLE state to the RRC_CONNECTED state requires extra signalling between the MS and the eNB [23]. Moreover, the eNB must allocate radio bearers for the MS. For these reasons there will be an extra delay to start data transmission when the MS is in the RRC_IDLE state. During this state the MS is inactive most of the time and it resembles the DRX operations during the RRC_CONNECTED state.

During the RRC_CONNECTED state the network knows the MS location at a cell level and data transmission and reception are possible between the MS and the eNB. In this state the MS reports channel quality information to the network, and the network controls and orders cell reselections. In the RRC_CONNECTED state the network has reserved radio bearers in the eNB and those are released when the MS moves to the RRC_IDLE state. During the RRC_CONNECTED state the DRX can occur if no data is transmitted. Additionally, one can assume that there is a timer on the network side which forces the MS into the RRC_IDLE state after some inactivity in packet transfers. This timer is not defined in the 3GPP specification and it is network vendor implementation dependent. In some networks this timer was found to be around 11.5 seconds [18]. MS moves to the RRC_IDLE state also, when an error occurs in the lower protocol layers, which is unrecoverable and requires actions from the RRC layer.

3.3 DRX and energy usage studies

Bontu et al. [22] analyzed LTE DRX power save mechanism and they described the LTE DRX timers at a detailed level. They assumed simply that when TX or RX is not

ON, 75 % of the energy is saved and did not explain the background for this assumption. The researchers estimated power savings for VoIP and for video streaming. For video streaming Bontu et al. used a very simple model, which sent packets continuously with a certain interval. Their conclusion was that for video streaming DRX may save 40-45 % of battery power and with VoIP the savings can be up to 60 %.

Huang et al. [18] studied LTE network performance and compared it with 3G and WLAN with real data collected from several users. They got 13 Mbps in DL and 6 Mbps in UL as median throughput values for LTE. They also derived empirical power model for LTE, which modelled energy usage of the MS. They noticed that LTE used more energy for short transmissions (e.g. one TCP packet) than WLAN or 3G, but it was more power efficient with larger transfers. The researchers also measured how long a time it took to change the state e.g. from the RRC_IDLE state to the RRC_CONNECTED state. They measured power consumption for different DRX states, which can be seen in Table 3. In the first column, there is the state name, the second column gives the amount of measured power consumption in that state, the third column shows the time used in that state and the fourth column explains about the meaning of the state in more detail.

Table 3: Measured power levels in different states [18]

State	Power (mW)	Duration (ms)	Explanation
LTE promotion	1210.7±85.6	260.1±15.8	MS moves from RRC_IDLE to RRC_CONNECTED
LTE Short DRX On RRC_CONNECTED	1680.2±15.7	1±0.1	Data transmission/reception during DRX
LTE Long DRX On RRC_CONNECTED	1680.1±14.3	1±0.1	Data transmission/reception during DRX
LTE tail base on RRC_CONNECTED	1060.0±3.3	11576±26.1	No data transmission but MS is ready and listening to channel, DRX is possible and after this duration MS moves to RRC_IDLE
LTE DRX On RRC_IDLE	594.3±8.7	43.2±1.5	MS listens to paging during DRX

For UL transmission a device uses much more energy than for DL reception. They derived data transfer power model to illustrate this. Formulas in their model were as follows:

$$P_u = \alpha_u t_u + \beta \quad (1)$$

$$P_d = \alpha_d t_d + \beta \quad (2)$$

Where P_u is UL power, P_d is DL power, t_u is UL throughput and t_d is downlink throughput. The instant power level which combined both UL and DL is

$$P = \alpha_u t_u + \alpha_d t_d + \beta \quad (3)$$

and the constants α_u, α_d and β are listed in Table 4.

Table 4: Data transfer power model constants

α_u (mW/Mbps)	α_d (mW/Mbps)	β (mW)
438.39	51.97	1288.04

Huang et al. also counted the total energy consumption for different networks and different states. Promotion energy contributed below 4 %, data transfer energy 47 % and tail energy consumption 48 % for LTE. In their measurements both 3G and LTE tail energy rations were surprisingly high, almost half of the total energy. The results did not differentiate in LTE tail period whether the device was on DRX or whether it was listening to the channel. They showed that the promotion timer, which controls the movement of the MS from the RRC_CONNECTED state to the RRC_IDLE state affected heavily the total energy consumption whereas the DRX-inactivity Timer, which controls the start of the DRX of the device after the last transmission or reception, had only a minor effect on the energy consumption of the device.

Kolding et al. [24] studied the LTE DRX impact on power saving and user throughput. They showed that 95 % reduction of the MS power can be reached with only 10-20 % loss in experienced throughput. They used web browsing traffic model [25] for the simulation and simplified models for different MS power states. The power value in their model seemed very low, e.g. for active data, it was only 500 mW. This value can be compared to an actual measured value which shows over 1600 mW [18].

Polignano et al. [26] studied DRX/DTX effects on Voice over IP QoS performance. First they started with a short introduction to how DRX is defined in LTE and explained the main features of dynamic and semi-persistent scheduling. They used the simplified power model for MS power consumption and the simulation based on *Matlab* model, which had several users in a cell. They calculated power usage with VoIP with different scheduling strategies and estimated how scheduling and DRX affects QoS of VoIP. They showed that the best way to save energy can be achieved with the semi-persistent scheduling but, as a side effect, more spectral resources were used for a VoIP call.

Aho et al. [27] studied battery saving opportunities and LTE network performance with VoIP. After a quick review to related studies, they explained in timer level how DRX works in LTE. The researcher group used simulator made with C++ programming language to simulate LTE network and the traffic therein. They simulated 21 active cells but the statistics were collected only from the six middle cells. Finally, Aho et al. included VoIP capacity measurements with different DRX parameters in their studies. They proposed adaptive discontinuous reception with the channel quality preamble to improve capacity in the cell. They pointed out that short DRX cycle timers are an attractive choice for LTE energy efficiency.

Herrería-Alonso et al. [28] proposed a new DRX mechanism where eNB queues downlink traffic until the queue size reaches certain threshold. So, data is not transmitted immediately and the scheme introduces some delay for the packets. Energy is saved because an MS can make use of DRX longer continuous periods of time. The proposed mechanism does not require any changes for standards and nor does it require any extra signalling. The mechanism resembles the packet coalescing technique introduced for Ethernet networks [29].

Hoque et al. [30] studied different multimedia streaming techniques and emphasized energy and quality of experience. They made several high level energy measurements and noticed e.g. that longer DRX cycles gave greater energy savings. They also compared different streaming strategies and noticed, not surprisingly, that delivering content continuously during the whole viewing time was much more power hungry than using e.g. throttling to deliver content in chunks.

Chen et al. [31] proposed a buffer aware scheduler for LTE eNB. The scheduler in eNB tries to allocate data so that an MS can receive video data as much as possible while in the RRC_CONNECTED state and stay in the RRC_IDLE state for as long as possible to save power. The scheduler uses buffer length and channel conditions as basis for scheduling. They simulated their scheduler with two video traffic models, but the traffic model details have not been revealed. Power model is based on a simplified model presented in [18]. Their simulations contain 5-40 UEs in the cell and the best power savings are received when there are many UEs in the cell. Their results do not tell what happens to video quality while scheduled this way.

Siekkinen et al. [32] used closed 3G and LTE network and shaped the streaming traffic profile into bursts before sending over the wireless network to the mobile. The shaping was done by a special proxy server. They used YouTube only with 3G network and noticed that Lumia 800 YouTube client downloads the whole video fast in “all-at-once” manner. So there is yet another streaming strategy found for YouTube. LTE was used only for audio streaming and the results show that shaping can give even 60 % energy savings when DRX is used in LTE network. It must be noted that their audio stream seems to have been only in one TCP stream without any other TCP streams causing traffic in the channel.

Another kind of traffic shaping was done by Lee et al. [33]. They changed HTTP GET headers sent by the browser with special SW. The header change caused video

server to send large chunks every 60 seconds. They measured traffic in a real LTE network and got around 35 % energy savings for an MS without DRX in use. In their network DRX was not active so they calculated that with DRX the savings could be up to 70 %. Their study did not include whether such large chunks are reasonable for network bandwidth usage.

Hoque et al. [34] made a survey that examines different solutions to improve the energy efficiency of wireless multimedia streaming in hand-held mobile devices. They categorize the research work according to different layers of Internet protocol stack the research utilizes. Most of the studies concern WLAN but LTE and 3G were also studied. They noticed that comparing the effectiveness of different solutions is difficult. The results depend on the hardware (HW) used and most studies used different devices. Additionally, it is difficult to measure the power consumption of individual components of commercial devices.

Deng et al. [35] proposed traffic aware technique to lower MS energy consumption. They developed a technique where an MS tries to predict when to move from RRC_CONNECTED to RRC_IDLE state and vice versa. They explained how an MS can request an LTE network to release RRC_CONNECTED state. However, they did not explain how an MS can request the network to move from RRC_IDLE state to RRC_CONNECTED state if the MS does not have any packets to transmit. Because existing networks did not support the used features, they simulated the results. Some traffic statistics and MS power values are measured in the real networks. Their study shows that the method could give 67 % energy savings in LTE networks.

Foddis et al. [36] studied the effect of RRC promotion timer to MS energy consumption and traffic overhead on the control plane. They used a simple energy model whose key parameters like timer values and traffic profiles are from the real LTE network. They monitored 8 users during one day. In their test network the promotion timer was originally 60 seconds, i.e. a very high value. In their simulations they also used the following values: 70.449, 12.154, 3.275 and 2.065 seconds. Using a 70.449-second promotion timer did not give any energy changes, but reducing the value to 12.154 seconds caused energy savings from 30 % to 50 % for all but one user who only had one video streaming session. It was not explained what kind of video streaming that was, but the other users had a lot more variety during the day, e.g. Twitter and Facebook traffic. Using a 3.275-second timer gave additional 20 % energy saving but using a 2.065-second timer did not give any extra benefits. With small promotion timer values they noticed significant increase in signalling overhead. They did not carry out any traffic shaping for the data.

Aqil et al. [37] developed a framework which helps user to choose a lower quality video and thus save energy because less transmission is needed. For this purpose they made a mathematical framework, which simulated lower LTE layers and the results were verified using simulations.

In a rather old study (2008) Xiao et al. [38] studied YouTube energy consumption in Nokia MS. They measured the energy consumption in both 3G and WLAN networks.

The results show that WLAN was more power efficient. They did not give details about YouTube traffic profile but buffering is used in the device. It could be estimated that there is no throttling and, e.g. in WLAN, transmission stops when there is still over 50 % of the viewing time left.

Lee et al. [39] proposed an algorithm which tries to maximise overall video playback time of an MS as a function of remaining data quota and battery energy. The algorithm finds an optimal interval between the chunks the video server is sending. This interval is different for every MS and the server should know battery and data status of the MS. The usage would require changes in video sending servers.

3.4 Targets in this thesis

This thesis analyses how video transmission and especially YouTube transfer behaves from an MS energy consumption point of view with LTE DRX. The emphasis is on different promotion timer values and how different timer values alter MS energy consumption. The power levels and equations used are from [18]. The results are presented in Chapter 7.

4 YOUTUBE TRAFFIC PATTERNS IN LAN

This chapter presents the measurements which were done within a commercial LAN network. In the beginning, the first measurement setup is briefly explained along with the reasons for choosing the methods used. Thereafter the general findings are presented and the latter part of the chapter consists of statistical analysis.

4.1 LAN measurement setup

Measurements were carried out in a commercial Sonera LAN network in Tampere, Finland with an Asymmetric Digital Subscriber Line (ADSL) modem and using a Windows XP computer. The measurements were done during May 2014. According to the observations, the network could give quite steady 20 Mbps DL throughput and 2 Mbps UL throughput. All possible background software (SW) was turned off during the video traffic pattern measurements.

Wireshark network analyzer [8] was used to capture TCP/IP data which was then filtered out using own proprietary *Python* software to get timestamps, data amount and the direction of the packets (DL or UL). Next the compressed measurement data was fed to *Matlab* to carry out the final analysis. For the packet timestamps the accuracy of 1 ms was used.

The measurements were performed by first starting *Wireshark* analyzer to capture the log and then using the Firefox Web browser to start video playback from YouTube, the video page was clicked. When the whole video playback was ready, the *Wireshark* capturing was stopped. During capture a few Internet Protocol version 6 (IPv6) packets belonging to non video related background processes were seen and those were filtered out before the analysis. The number of discarded packets was limited to only tens of packets (hundreds of bytes) whereas the measurements contained megabytes of data.

4.1.1 First set of measurements

The measurements were started simply by taking some YouTube logs (around 20 different YouTube videos) and analyzed using *Wireshark* options. Shortly, some regular patterns in the data profile were noticed: the YouTube video server sent data in certain intervals, not continuously. It was also noticed that the log files contained several independent TCP streams, two of which were the most dominant ones: over 90 % of data was transferred in these two TCP streams. This first rough analyze phase was done

simply with looking at the screen and using pen and paper when calculating packet delays and differences.

Next, *Matlab* was used to analyze the measurement data. In the beginning, packet sizes versus time were plotted. One example can be seen in Figure 4, which presents video of 300 seconds. This figure contains both UL and DL data. In X-axis, there is time in seconds and in Y-axis there is the amount of data in kBytes. This is a typical example figure of YouTube video data traffic. This very raw picture alone indicates that at the beginning of data transfer there occurred high activity (first 10 seconds of video) and later can be seen symmetrical data transmission peaks in regular intervals.

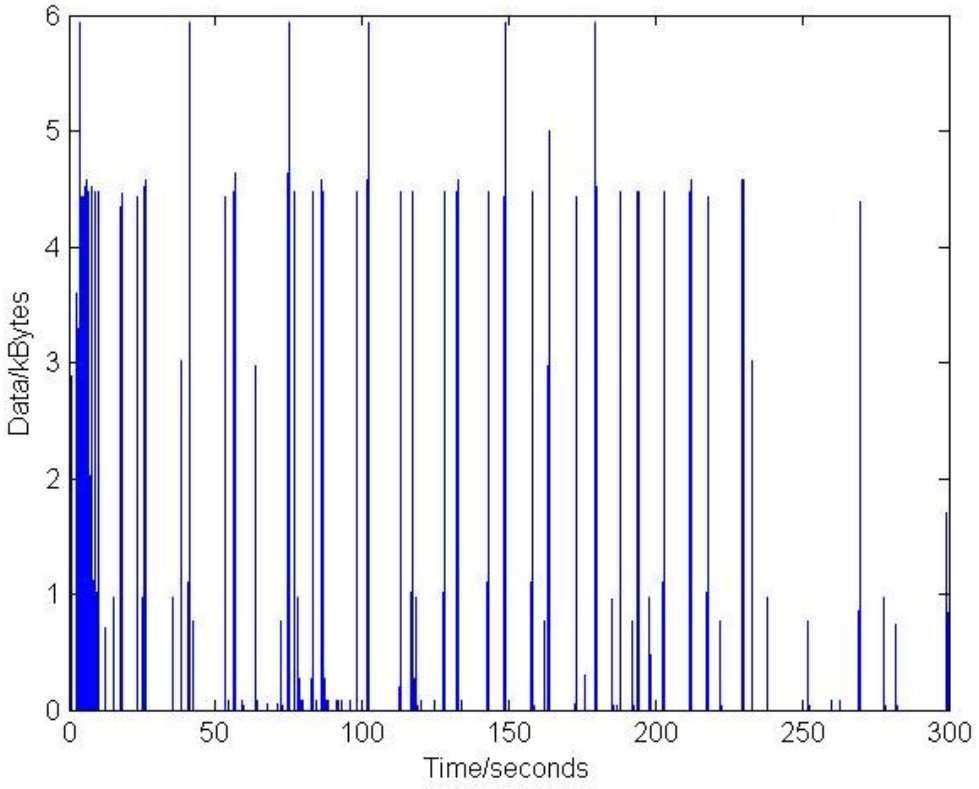

Figure 4: Example of raw TCP/IP data sent and received during YouTube video downloading

In the next phase of the analysis, the regularity was studied in more detail. The next experiments involved autocorrelation and cross correlation to provide a better picture of the time correlation in the data bursts. Studies were made with autocorrelation of UL traffic, DL traffic, Bidirectional (BIDI – both UL and DL data contained in same log) traffic and cross correlation of UL and DL traffic. Before calculating correlations, data was smoothened with 1 second mean integrator. As one could assume, both UL and DL correlated very heavily with each other. As an example, DL autocorrelation is presented in Figure 5 (this is the same data as in Figure 4 but it contains only DL data). Here can additionally be seen clear regular correlation peaks.

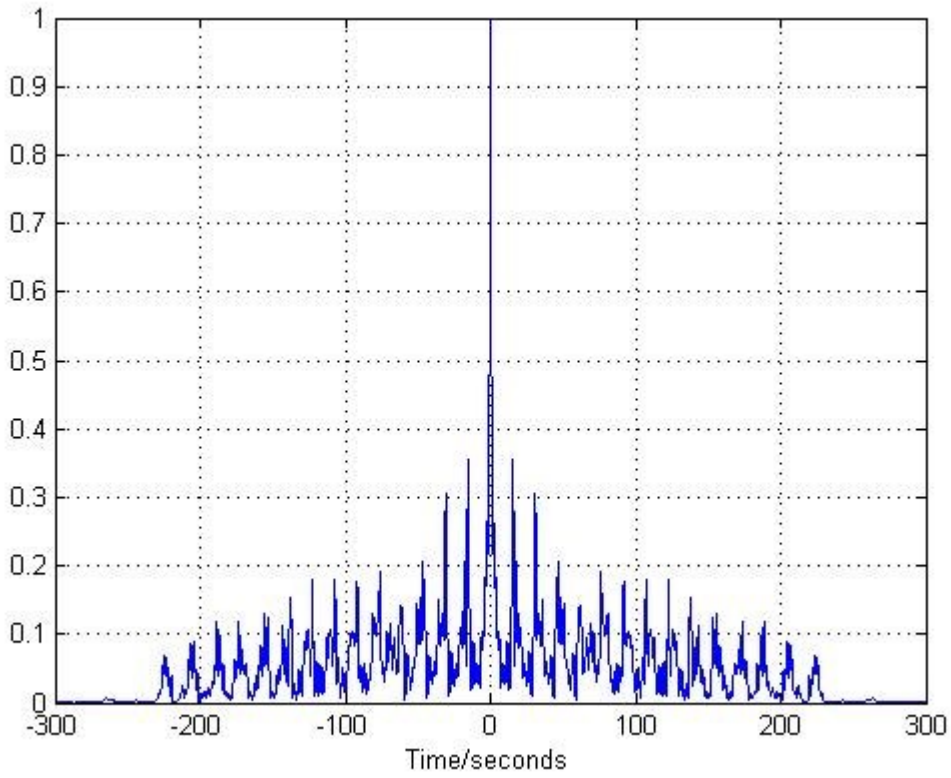

Figure 5: Example of autocorrelation of DL data

Next was studied the width of the main correlation peak around 0 second. According to the theory this should give the length of the initial burst in time. The autocorrelation results were compared with the results calculated directly from *Wireshark* log with pen and paper. The bursts were so short in time that no accurate results were obtained from autocorrelation: the results depended very heavily on at which correlation point the width was calculated: e.g. in Figure 5 at point 0.9 correlation value was around 0.23 s which is much shorter a time duration than at correlation value 0.5 where the correlation value was 0.67 s. Examining *Wireshark* log for the same file, the initial burst length was around 0.07 – 0.41 seconds depending on which of the packets were included in the calculations. So it was difficult to estimate the end of the initial burst.

The next task was to analyze the distance between the side peaks in the autocorrelation. This distance should tell the time between transmissions of data bursts noticed e.g. in Figure 4. There was observed that the first side peak appears at around 15 seconds and when comparing to the original *Wireshark* log, it matches very well. The second side peak appears to be around 30 seconds, so it is the multiple of the first side peak.

4.1.2 Final measurement setup

After the first analysis it became quite evident that traffic patterns described in [7], [9], [11] were not observed. In all of those studies were seen 64 kBytes data bursts sent by YouTube server, and usually the pauses between the bursts were quite short, i.e. 0.5

seconds or less. Besides, it was noticed that using the autocorrelation did not give information that is accurate enough for this study. Originally, it was only planned to verify and use the results seen in [7], [9], [11] but now it was decided to study the YouTube traffic patterns in more detail.

For further study, *Matlab* analysis scripts were changed into every TCP/IP packet being bundled to the same "chunk" if the time difference between the adjacent packets was less than 200 ms. This is the same method as in [7]. The time stamp of the chunk was the timestamp of the first packet in the chunk and raw data from *Wireshark* was used as basis. This means that every chunk consisted of one or more TCP/IP packets and the size of the chunk was the sum of the IP packet sizes in bytes. Figure 6 shows how IP packets were converted into chunks.

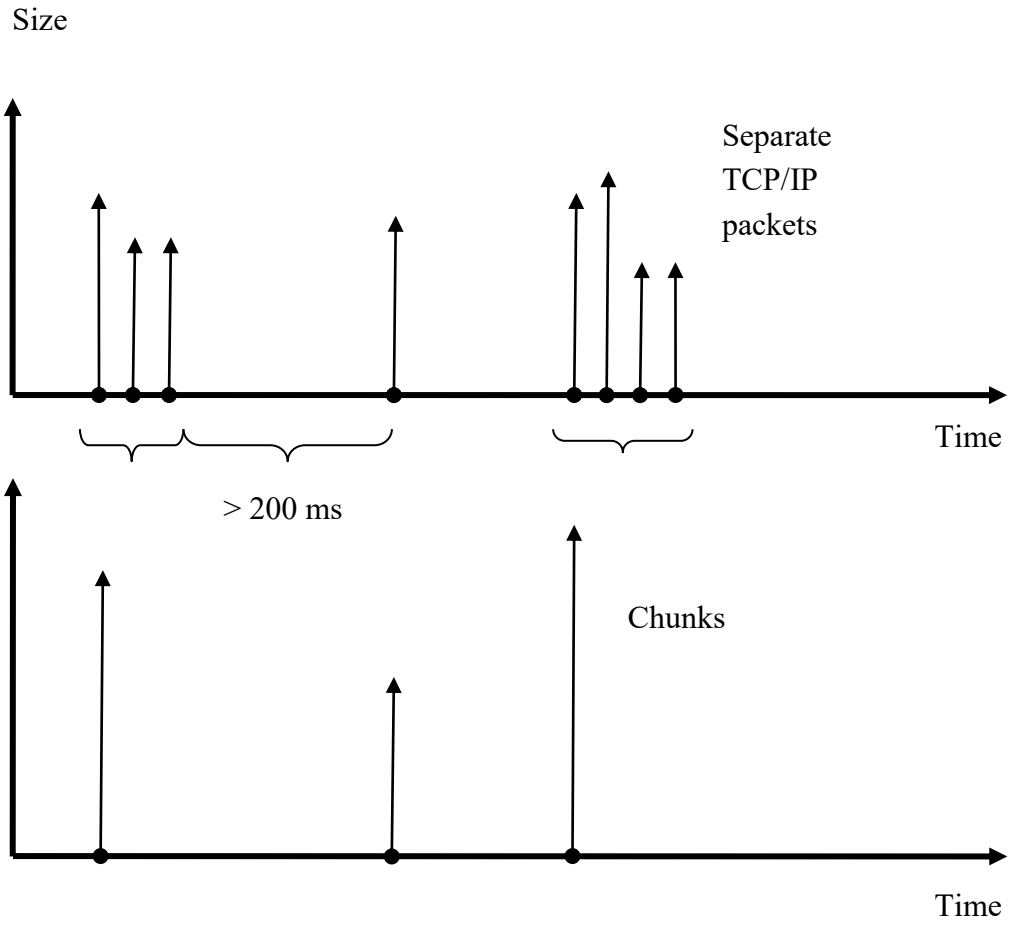

Figure 6: Converting TCP/IP packets to chunks

To reduce the possibility of video coding causing variations to measurements, 10 arbitrary videos were chosen with 360p video coding quality. In each case no video settings were changed before playback. Four of the videos were sports related, two were music videos, two were TV shows, one was a street view and one was a video gaming video. The chosen videos lasted several minutes in order to have proper statistics, so the lengths of the chosen videos are between 272-468 seconds (4 minutes 32 seconds – 7 minutes 48 seconds).

4.2 General findings regarding the traffic patterns in LAN

The same calculations were made to all of the ten video clips. The analysis also included visual checking from both *Wireshark* logs and *Matlab* figures to find any measurement or setup errors. Using "chunk method" presented in Chapter 4.1.2, one can easily spot nice regular patterns in all of the videos. Figure 7 presents a typical view of YouTube data traffic. It shows the chunks found versus their timestamps. This is the same data set that was used in Figure 4.

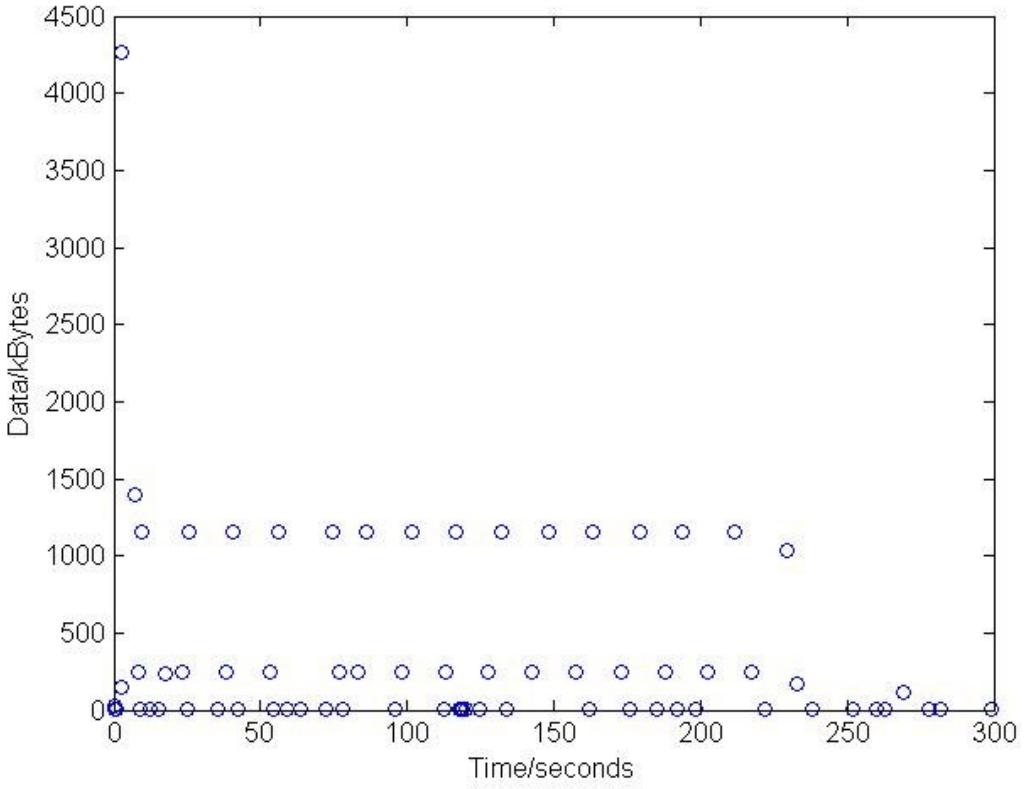

Figure 7: Transmission and reception plotted as chunks

As can be seen in Figure 7, there are the following three characteristics in YouTube videos:

1. There is a clear *speedup phase* at the beginning of transfer. This phase lasts only a few seconds but can include several chunks in a short period of time and some of them can be very large in size, e.g. in this particular video there is a single chunk over 4000 kBytes visible. The magnification of the speedup phase can be seen in Figure 8, which shows the chunks between 0-25 seconds.

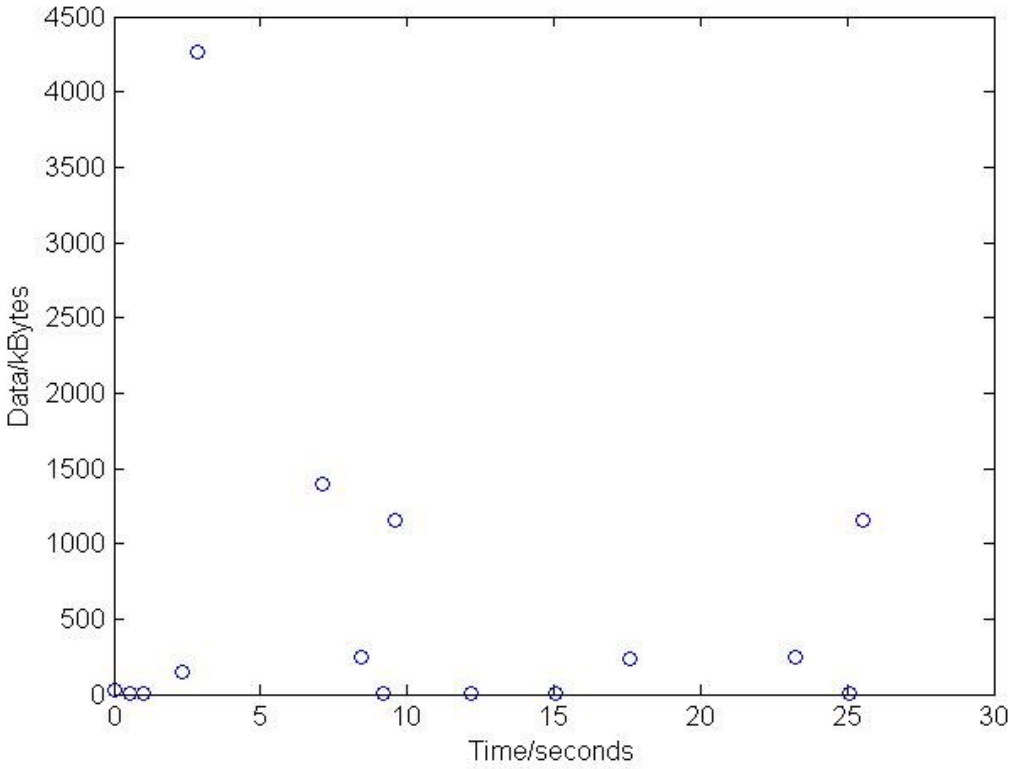

Figure 8: Magnification of the speedup phase

2. The speedup phase is followed by a *steady phase* which contains three different streams:
    - One higher stream with regular intervals, in this example chunks over 1100 kBytes
    - Another lower stream with regular intervals, in this example under 500 kBytes but over 200 kBytes
    - Several irregular small chunks normally under 200 kBytes

    The magnification of this phase is seen in Figure 9, which shows the chunks after 25 seconds.
3. After the steady state both the regular high stream and the low stream fade out and only irregular chunks remain in *video tail* phase. In Figure 7 this can be seen roughly after 250 seconds.

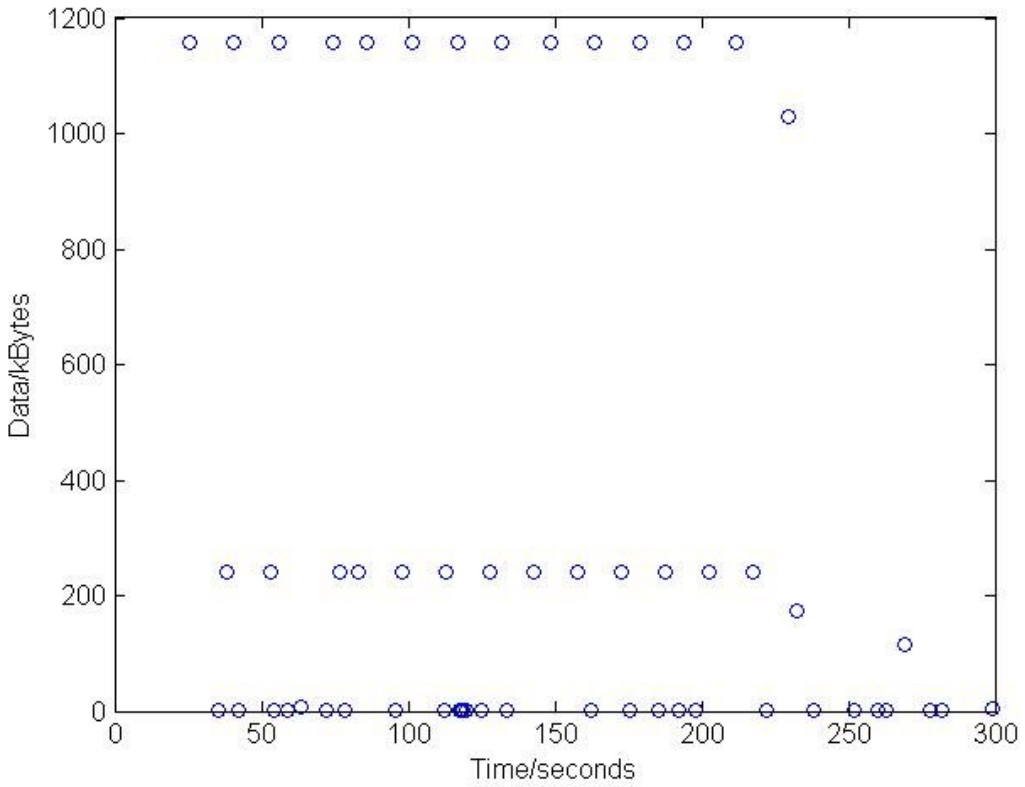

Figure 9: Magnification of the steady phase, starting from 25 seconds

Because the high and the low stream were very regular, one could assume that it was caused by two major TCP streams that were noticed in *Wireshark* logs. For this reason, the rest of the other traffic was filtered out and only these two TCP streams remained. In addition, it could be anticipated that the high chunk stream could be caused by one single TCP stream and the low chunk stream by another TCP stream. To verify this, the chunks belonging to the different TCP streams were separated. This situation is plotted as an example in Figure 10. The chunks belonging to the TCP stream with more data is in blue colour and the chunks belonging to the TCP stream with less data are in red colour. There can be seen that these two major TCP streams were both involved in the speedup phase and they also formed the high and the low streams in the steady phase and contributed to the fading out phase, too. Surprisingly, the TCP streams did not match perfectly with the low and the high chunk streams. Some of the chunks did not match clearly with the stream (high or low) that they were supposed to belong to. Instead, they were obviously from the other stream. This becomes even more visible in Figure 11. This figure is from a different YouTube video clip than the earlier examples. So, evidently, the chunks in the high and the low chunk stream consisted of a mix of two TCP streams.

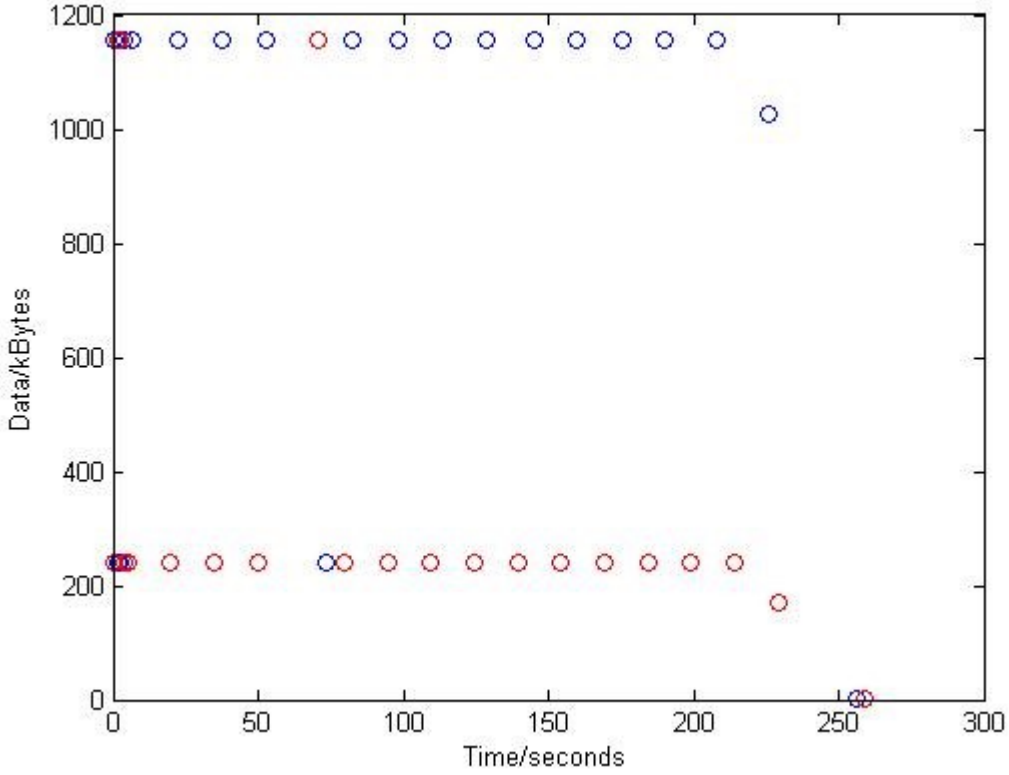

Figure 10: Two major TCP streams form the high and the low stream. TCP stream with more data in blue colour and TCP stream with less data in red colour

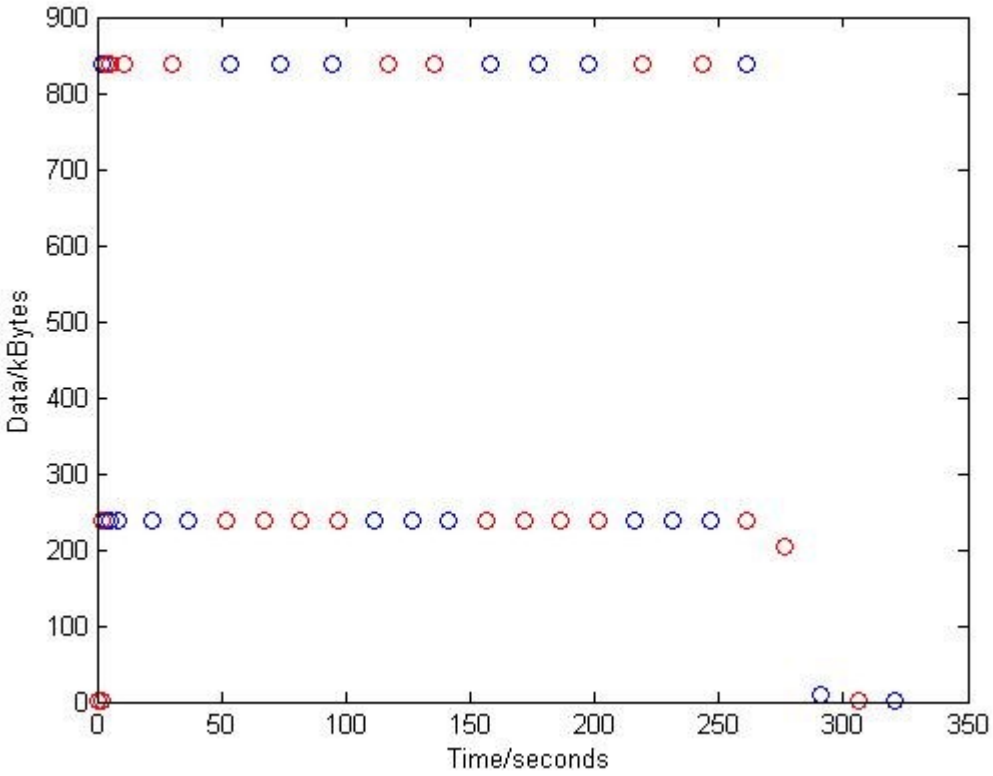

Figure 11: TCP stream with more data (another example). TCP stream with more data in blue colour and TCP stream with less data in red colour

Because these two major TCP streams seemed to contain most of the data, it was interesting to see how chunk patterns look like if these two streams are filtered out. The remaining of the data could be called as "noise" and they consisted of several small TCP streams. In some of the logs, one can spot almost a hundred small streams, in other logs only ten small streams. The majority of these streams were traffic that the browser has with Google servers (N.B. YouTube is owned by Google). There may be a few streams caused by Windows XP e.g. to check if any updates should be available. Figure 12 shows an example of this background noise. It is visible that most of the chunks were very small but there were also some chunks over 100 kBytes.

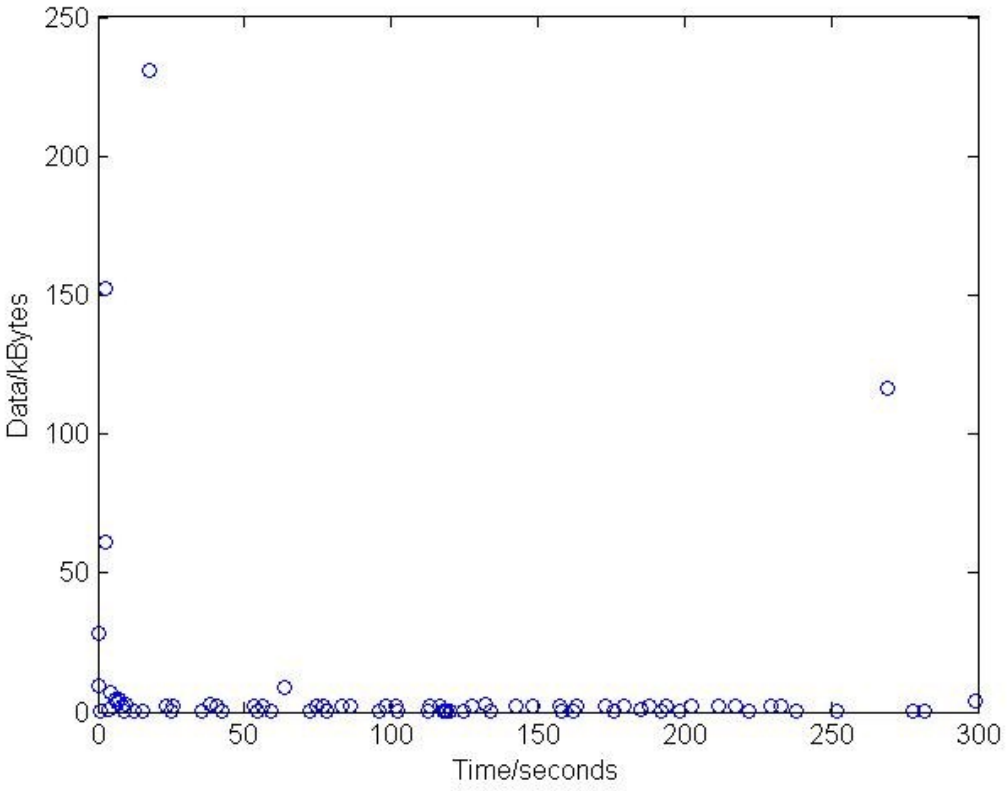

Figure 12: Major TCP streams filtered out, only background "noise" TCP streams remained

4.3 LAN statistical examination

The same ten different YouTube log files from the previous chapters were examined and statistically analyzed using *Matlab*. The analysis covered the following parts: full original file, two major TCP streams alone and only "noise" part – where the two major TCP streams were filtered out.

4.3.1 Statistical evaluation of the full file

The sum of the lengths of the videos was 3429 seconds (57 minutes 9 seconds) and a total of 278606017 bytes (approximately 272076 kBytes) was transmitted or received in TCP/IP level. These figures also include IP and TCP headers. In DL, 271604219 bytes (approximately 265238 kBytes) were received and in UL 7001799 bytes (approximately

6838 kBytes) were transmitted. This means that UL takes only 2.5 % of the total transmission/reception and a YouTube video viewing is very DL dominated. This is an expected result because UL mostly consists of TCP acknowledgements. If all these videos were transmitted in steady speed during the whole viewing time, it would make 79208 bytes per second. Because the reception capacity was around 20 Mbits/s these 79208 bytes could be transmitted in 0.03 seconds and RX could sleep for 0.97 seconds (or 97 % of reception time) per every second. But since YouTube uses video encoding all of the videos do not contain the same amount of data per second. This can be seen in Figure 13, which shows all of the ten video clips and the amount of DL data every viewed second contains.

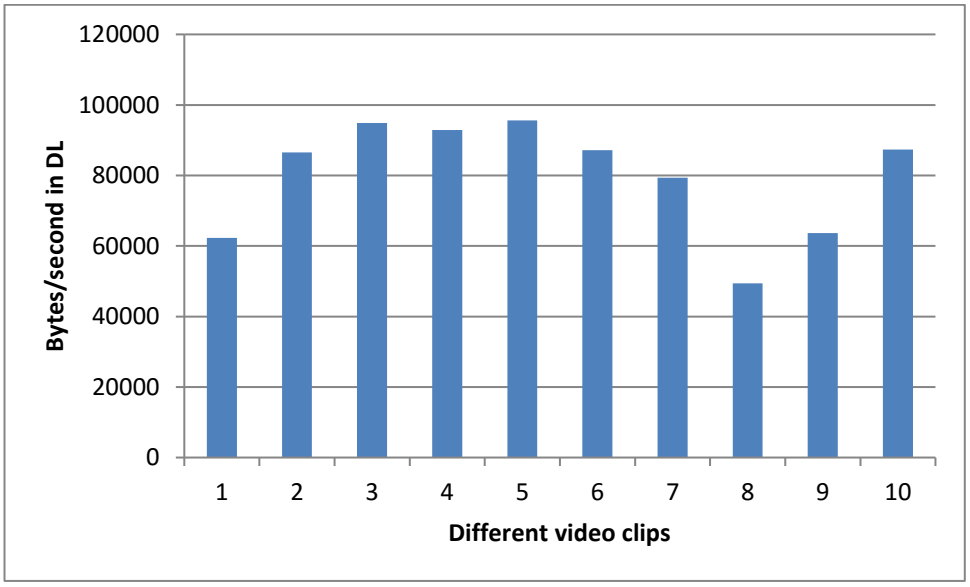

Figure 13: Video clip bytes per every viewing second

The most data intensive clip contained 95636 bytes per second while the least intensive clip only 49414 bytes per second, which is approximately 50 % less. These were both music videos, so the type of the video does not explain the difference.

The speedup phase was defined to contain all the chunks until there was the first chunk in the two major streams at the same level as all the rest of the chunks in the major streams in the steady phase. Additionally, in the speedup phase the data amounts between the clips varied in the same way as the total data amounts between the clips. To find out if there was any regularity in the speedup phase, the proportion of DL bytes received in the speedup phase was compared to the total received amount of DL bytes. Figure 14 shows the results. All the results are close to each other and the average value is 0.20.

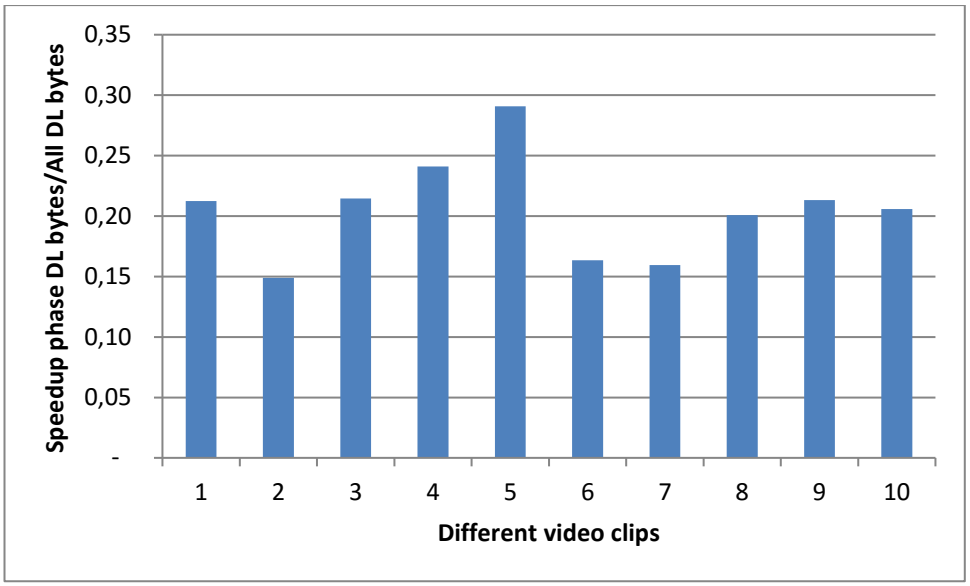

Figure 14: Number of speedup phase DL bytes divided by all DL bytes

Finally, all the DL speedup phase bytes of all the video clips were added up, which gives the total of 54569395 bytes. When this is divided by all the DL bytes, it is 0.2009, which reveals that, on the average, 20 % of DL data of video clip is transmitted during the speedup phase.

The length of the speedup phase was also measured. The length of the speedup phase is defined as the last timestamp of the chunk still belonging to the speedup phase. The results for the different video clips are shown in Figure 15, which shows the lengths of the speedup phase for the different video clips. The average value of the speedup lengths is 9.97 seconds.

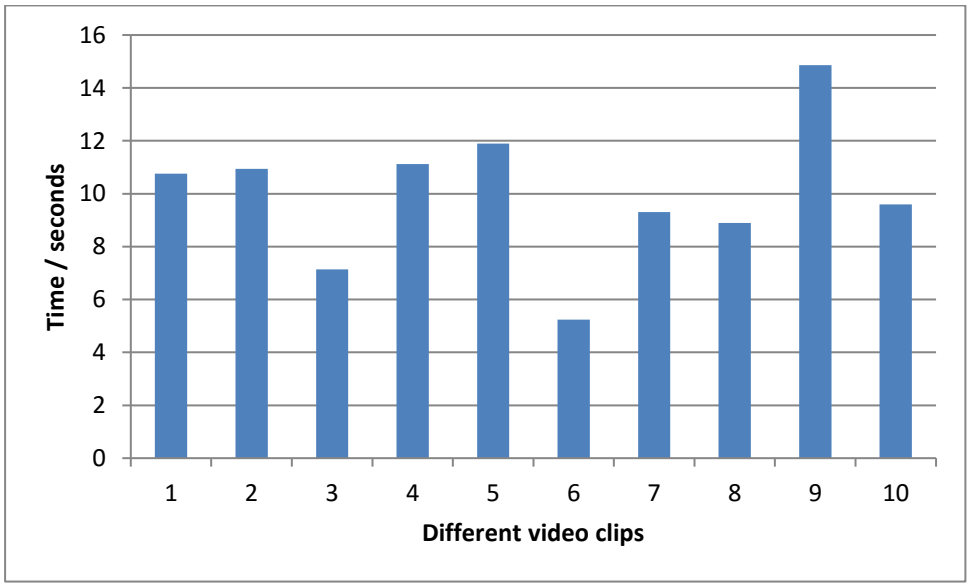

Figure 15: Speedup phase length in seconds for different clips

4.3.2 Statistical evaluation of the major TCP streams

The rest of the data was filtered out except for the two most dominant TCP streams. The reason for this can be seen in Table 5, which presents the portion of the two TCP streams per video clip.

Table 5: Portion of 2 major TCP streams per video clip

Video clip number	1	2	3	4	5	6	7	8	9	10
Percentage from all data in clip	91	98	97	96	97	97	98	97	97	98

The average value for portion is 97 %. It is quite clear that the majority of the data comes from these two TCP streams.

In DL, 264263677 bytes were transmitted and, in UL, it was 5500172 bytes, so the UL traffic byte amount was 2.1 % of the DL amount. In general, it was observed that TCP servers sent two 1500-byte IP packets in DL, which UL then acknowledged with one 40-byte IP packet. In DL, 176776 IP packets were received and, in UL, 111260 IP packets were transmitted. This makes approximately 1.58885 DL packets for a single UL packet in the major streams. The values above indicate that an average package size in DL was 1495 bytes and, in UL, it was 49 bytes.

Next TCP/IP packet delays inside the chunks were compared. These results included both DL and UL packets. All the results from the ten different video clips were added up and the total of 287607 difference values was compared. To examine how time differences were distributed, it was used Empirical Cumulative Distribution Function (ECDF), which is defined as:

$$F_n(x) = \frac{1}{n}\sum_{i=1}^{n} 1\{X_i \leq x\} \tag{4}$$

where

$$1\{X_i \leq x\} = \begin{cases} 1\,, if\ X_i \leq x \\ 0, if\ X_i > x \end{cases} \tag{5}$$

and X_i is random variable and n is number of samples [40].

Figure 16 shows the ECDF of the time differences inside the chunks. It is evident that the TCP/IP packets inside the chunks were very densely grouped.

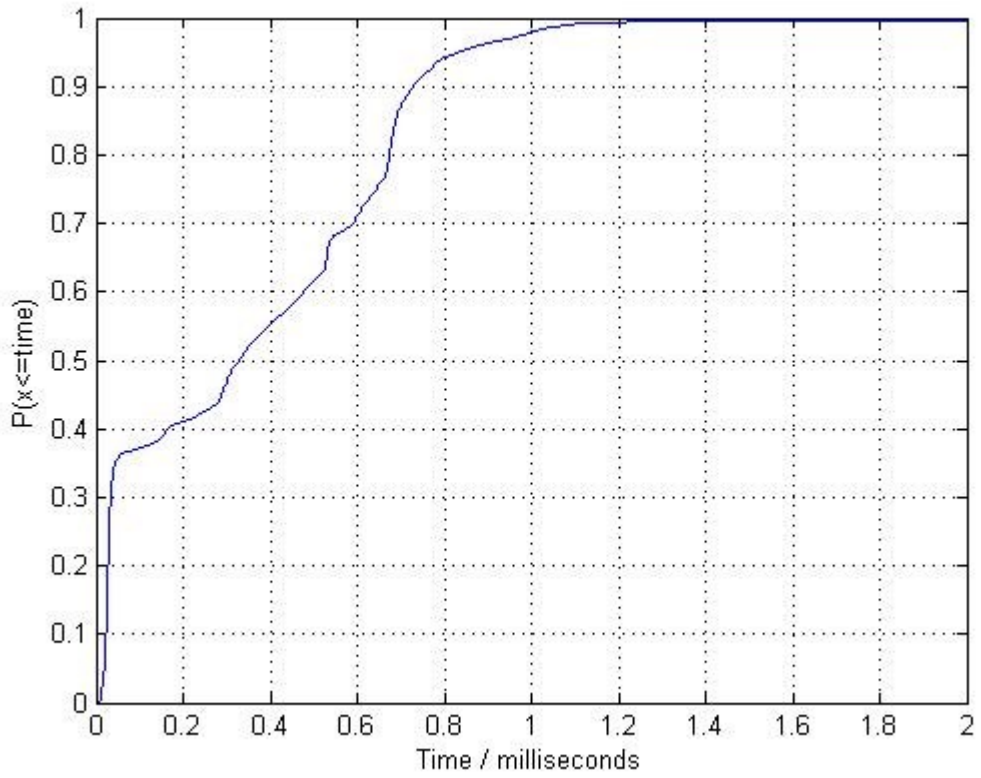

Figure 16: Empirical CDF of TCP/IP packet time differences inside chunks

For 94 % of the TCP/IP packets the difference between them was less than 0.8 ms. The median value was 0.33 ms and the average value was 0.42 ms. The maximum value of time difference was 199 ms. This was expected because packets were defined to belong to the same chunk if the difference was less than 200 ms. Time difference was larger than 150 ms only for 17 out of 287607 values. Cumulative probability values are presented more detail in Table 6. These values are rounded off to the integers.

Table 6: Cumulative probabilities of IP packet time differences inside chunks

Time equal or below (ms)	0.03	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.2	1.3
Probability (%)	25	37	41	47	55	61	71	87	94	96	99	100

For the two major TCP streams the speedup phase data amounts were recorded and compared to the total amount of the data received in the two major TCP streams. As with full data file, the average proportion was 0.193, so 19.3 % of data was sent during the speedup phase.

Next, time differences between DL chunks were calculated. These calculations contained both high and low stream (see Figure 10). Table 7 reveals the statistics of the

time differences between the chunks in the two major TCP streams. Median, average, Standard Deviation (STD) and minimum difference are shown in seconds.

Table 7: Time differences of chunks in two major TCP streams

Video clip	1	2	3	4	5	6	7	8	9	10
Median (s)	8.51	3.74	6.02	5.24	7.19	6.52	7.22	6.83	8.40	6.62
Average (s)	8.44	6.65	7.2	7.13	6.97	7.08	7.74	7.83	8.62	6.91
STD (s)	5.17	5.83	5.57	5.79	4.12	6.05	5.63	6.19	5.01	5.25
Minimum	0.287	0.352	0.857	0.590	0.864	0.344	0.303	0.430	0.997	0.394

One could discover that the average delay between the chunks varied from 6.65 seconds to 8.62 seconds while the median delay varied from 3.74 seconds to 8.4 seconds. The median value of all the median values was 6.7 seconds and the average was 7.5 seconds. The median of the standard deviation was 5.6 seconds. The standard deviation was quite high and other values varied significantly, too. One reason for this variation was that the measurement values also included the speedup phase chunks, which seemed to appear in irregular intervals and in very short intervals, as well.

Because there were clearly two chunk streams (higher and lower) and the combined results (see Table 7) did not give any clear regularity, values were calculated separately for these streams. Chunk sizes over 500000 bytes (488 kBytes) were considered to belong to the high stream whereas the rest of the chunks to the low stream. This separation matched very well for all the video clips. The time differences in the steady phase between the high stream chunks and the low stream chunks varied from 0.33 seconds to 15.033 seconds. The median value was 7.2 seconds and the average value was 7.3 seconds.

4.3.2.1 High stream statistics

Figure 17 shows median and average values for the higher stream chunk sizes in DL. These values also included the chunk sizes from the speedup phase, so, in terms of the approximation of the chunk sizes during the steady phase, the median values give better results than the average values. The median of median values was 1147 kBytes. It can be seen that with video clips 1 and 8 the values are clearly different from the other values. Both of these clips contained the least data per viewing second and YouTube clearly controls the high stream chunk size depending on the video coding rate and total video file size.

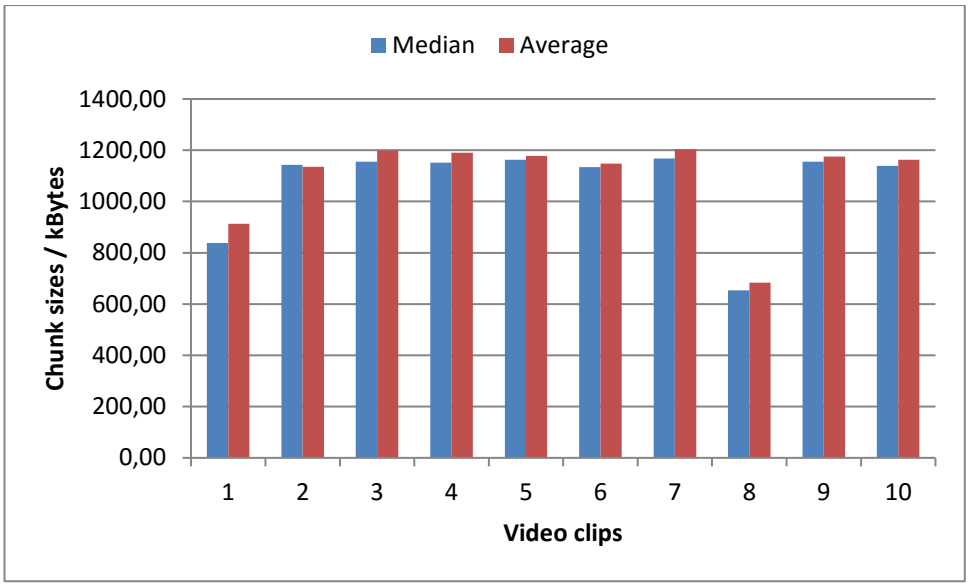

Figure 17: Median and average values of high stream DL chunk sizes

To illustrate this, the median of DL high stream chunk sizes were compared to the total DL video clip size. This is seen in Figure 18. The average value of these was 0.041, which means that approximately 4 % of total data was sent in one higher stream chunk during the steady phase. The value was 0.042 when compared to the total DL major stream size. Also, video clips 1 and 8 which had smaller chunk sizes than the other clips were now very close to 4 %. The clip 2 chunk size was under 3 % although the average and median values did not differ from the other clips. This could mean that YouTube also limits the maximum value of the chunk size. Later can be seen that for clip 2 also transmission lasted longer than for the rest of the clips, see Figure 21.

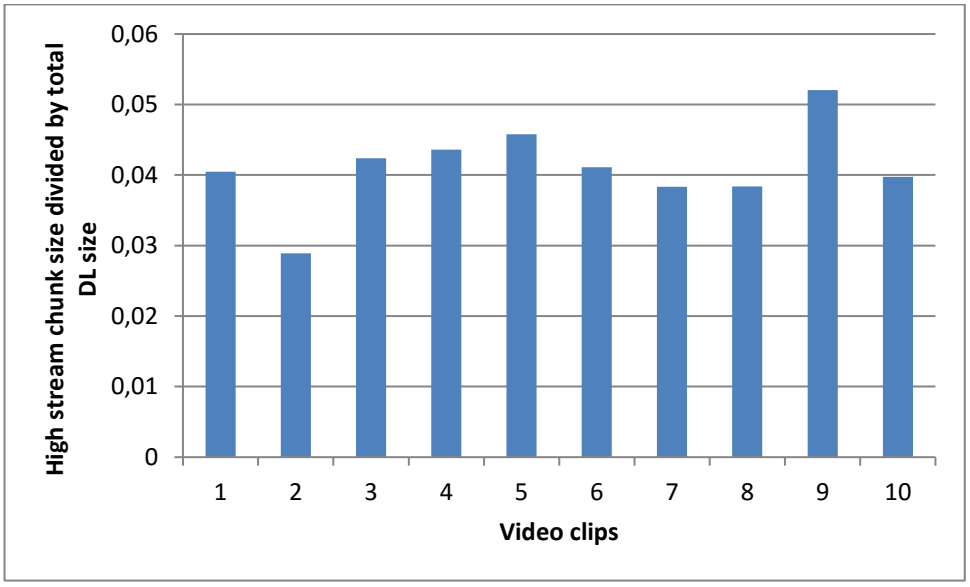

Figure 18: High stream DL chunk size median divided by total DL video clip size

Data from all the video clips were added up and empirical cumulative distribution was calculated. Here the speedup phase was not included in the calculations, so the distribu-

tion contains only the chunks from the steady phase. The results are seen in Figure 19. In blue colour is printed the distribution from all the clips and in red colour the distribution when clips 1 and 8 are excluded. Clip 1 had all values exactly 838.8 kBytes except one value was 1077.6 kBytes and clip 8 had all values exactly 653.9 kBytes.

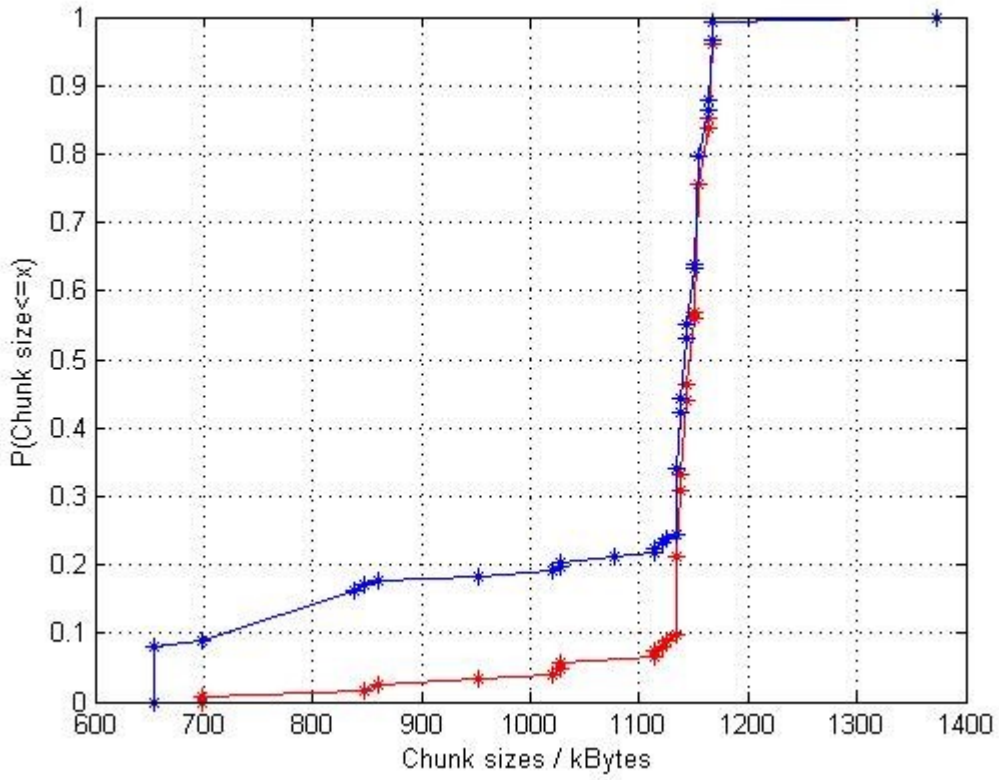

Figure 19: Empirical cumulative distribution of high stream chunk sizes in the steady phase. The results are from all the clips in blue colour and without clips 1 and 8 in red colour.

Cumulative probabilities for the high stream chunk sizes without clips 1 and 8 are also presented in Table 8. The median value was 1151 kBytes and the average was 1138 kBytes. Over 90 % of the chunk sizes are between 1130 – 1170 kBytes. It is quite clear that there is not much variation in the high steam chunk sizes during the steady phase, so the conclusion is that, YouTube tries to keep the size as a constant.

Table 8: Cumulative probabilities for the high stream chunk sizes in the steady phase without clips 1 and 8

Chunk size below (kBytes)	699	700	800	900	1000	1100	1130	1135	1140	1145
Probability (%)	0	0.8	0.8	2.4	3.3	5.7	8.9	21.1	33.3	46.3
Chunk size below (kBytes)	1150	1155	1160	1165	1170	1200	1300	1400		
Probability (%)	46.3	56.9	75.6	85.4	99.2	99.2	99.2	100		

Next studies handled the time differences between the high stream chunks. In the earlier figures one can notice that, during the steady phase, the chunks appeared very regularly in all log files, and, for this reason, the median values of the time difference in the chunks were analyzed. The results are visible in Figure 20. In seven video clips, the median value of difference was very close to 15 seconds and three clips had median values close to 20 seconds. The median of medians was 15.41 seconds.

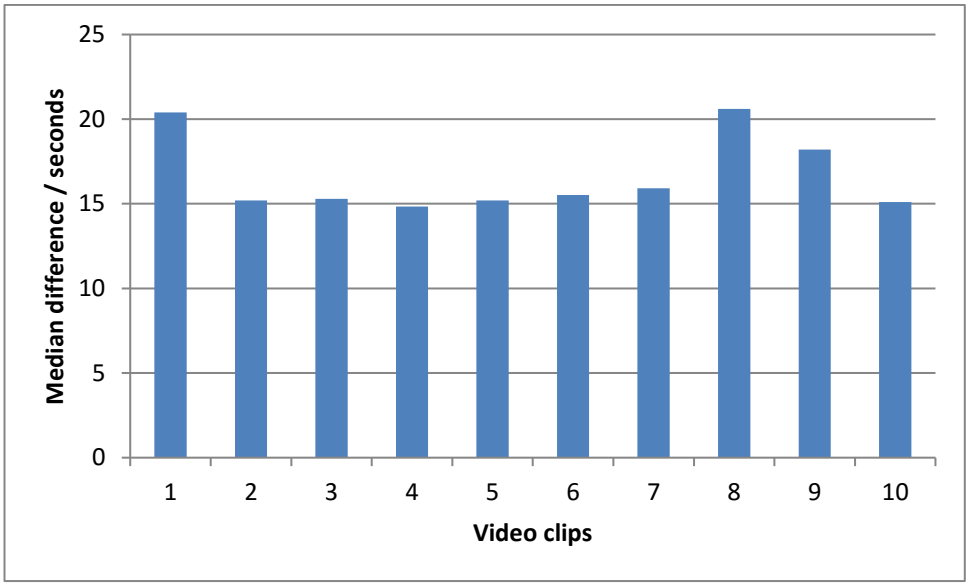

Figure 20: Median of time difference in high stream chunks

It is interesting to see that video clips 1, 8 and 9 had the least data per viewing second (see Figure 13) and these clips also have the highest median values for high stream steady state time difference, the equivalent values being 20.4 s, 20.6 s and 18.2 s. The rest of the clips were very close to 15 seconds. But when comparing the chunk sizes of the clips per total DL transmission amounts, clips 1 and 8 also have the proportion of 4 % and only clip 9 shows some difference of over 5 % value. In order to see how YouTube controls the median time difference of chunks in the high stream, the proportion of the last full size chunk time was compared to the total transmission time of the

clip. The last full chunk stands for the chunk still belonging to the high stream and the size of which is very close to the median of the high stream chunk size. This proportion can be seen in Figure 21. This proportion varies from 0.67 – 0.82 and the average value is 0.74. It can be observed that the high stream of the clips 1 and 8, which had the greatest median differences between the chunks, still finishes smoothly after 70 % of the video has elapsed. Therefore, it seems that YouTube tries to limit the high stream chunk size to 4 % of the total DL transmitted size. The time difference between the high stream chunks is 15 seconds but, if the clip size is quite small, a YouTube server uses a longer time difference allowing the last full chunk to appear at the point where 74 % of the full video has elapsed.

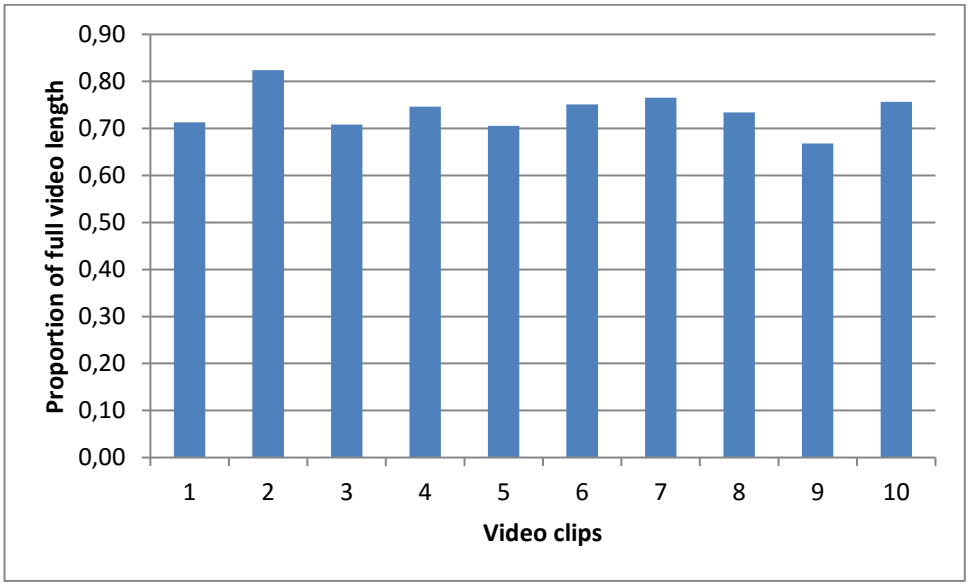

Figure 21: Proportion of the last high stream chunk of total video length

Empirical cumulative distribution of the time differences was also calculated. Here only the high stream chunks belonging to the steady phase were used. The results are in Figure 22, where in blue colour is ECDF of clips 1, 8, 9 and in red colour is the distribution of the rest of the clips. In green colour is plotted the corresponding normal distributions with the average value of 21.963 seconds and the standard deviation of 4.8995, and with the average value of 16.2945 seconds and the standard deviation of 2.4425. Average and standard deviation values were calculated directly from the measurements. With clips 1,8,9 the median was 21.078 seconds and with clips 2,3,4,5,6,7,10 the median was 15.467 seconds. Normal cumulative distribution can be calculated using the equation:

$$p = F(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{x} e^{\frac{-(t-\mu)^2}{2\sigma^2}}dt \qquad (6)$$

where p is the probability, μ is mean and σ is standard deviation [41], [42].

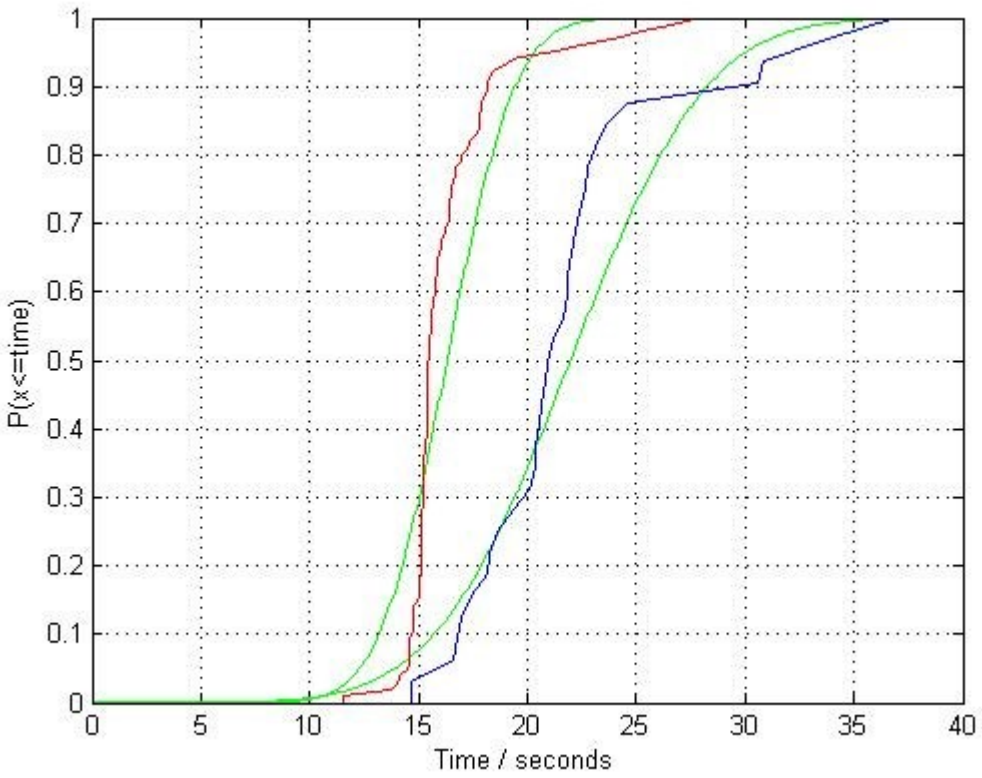

Figure 22: Empirical cumulative distribution of the high stream chunk time differences in the steady phase. In blue colour are clips 1, 8 and 9. In red colour are clips 2,3,4,5,6,7 and 10. In green colour are the corresponding normal distributions.

Average and standard deviation values are directly from the measurements. Root-Mean-Square-Error (RMSE) can be used to measure the differences between two time series. RMSE can be calculated using the equation:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(x_{1,t} - x_{2,t})^2}{n}} \quad (7)$$

where $x_{1,t}$ and $x_{2,t}$ are samples from two different series and n is the number of samples [43]. The RMSE of the cumulative normal distribution for clips 1, 8, 9 was 0.107 and the RMSE for the rest of the clips was 0.139. The exact cumulative probabilities are in the Table 9. It can be deduced that the most of the time differences are very densely packed.

Table 9: Cumulative probabilities of the high stream chunk time differences in the steady phase.

Time difference below (seconds)	Probability for clips 1,8,9 (%)	Probability for clips 2,3,4,5,6,7,10 (%)
11.55	0	0
12	0	1.0
12.5	0	1.0
13	0	1.0
13.5	0	1.0

14	0	2.9
14.5	0	4.8
15	3.1	16.2
15.5	3.1	50.5
16	3.1	65.7
16.5	3.1	75.2
17	9.4	78.1
17.5	15.6	81.9
18	15.6	87.6
18.5	21.9	92.4
19	25	92.4
19.5	28.1	93.3
20	28.1	94.3
20.5	37.5	94.3
21	50.0	94.3
21.5	53.1	95.2
22	62.5	95.2
22.5	71.9	95.2
23	78.1	96.2
23.5	81.3	96.2
24	84.4	96.2
24.5	84.4	97.1
25	87.5	97.1
25.5	87.5	98.1
26.5	87.5	98.1
27	87.5	99.0
28	87.5	100
28.5	93.8	100
34	96.9	100
37	100	100

4.3.2.2 Low stream statistics

As with the high stream chunk sizes, the same values were also calculated for the lower stream and the results can be seen in Figure 23. Unlike the high stream median values, these low stream median values were always exactly the same for all the video clips: 239 kBytes (exactly 244498 bytes).

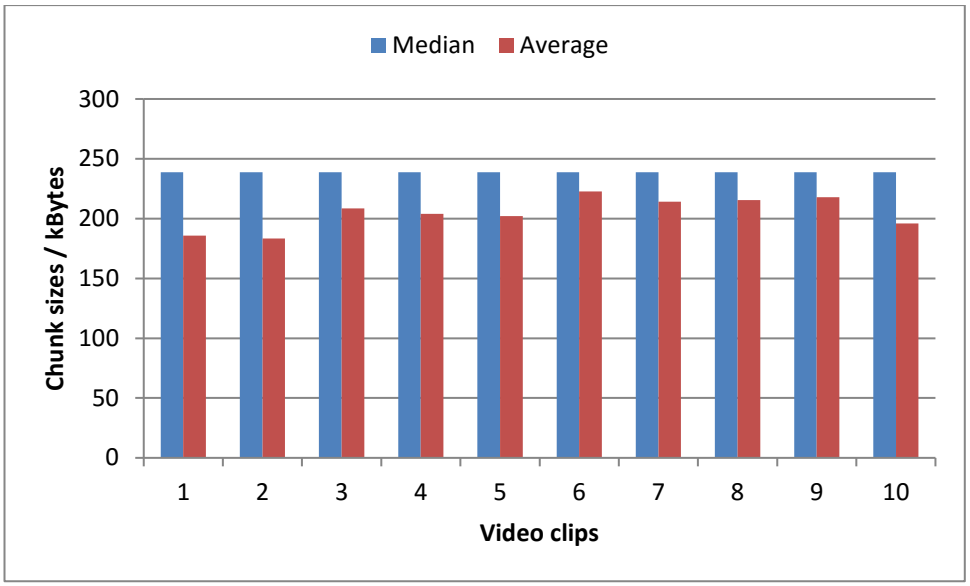

Figure 23: Median and average values of low stream DL chunk sizes

Empirical cumulative distribution for the low stream chunk sizes during the steady phase can be seen in Figure 24 and a more detailed distribution in Table 10. It still seems that YouTube tries to keep this low stream chunk size constant in the steady phase and the chunk size does not depend on the total video DL size. An average value in the steady phase was 203.1 kBytes and median values were 238.8 kBytes. In fact, over 78 % of the chunks were between 238-239 kBytes and 68.5 % of the values were exactly 244498 bytes (238.8 kBytes). The minimum value was 40 bytes, i.e. the size of one TCP acknowledgement.

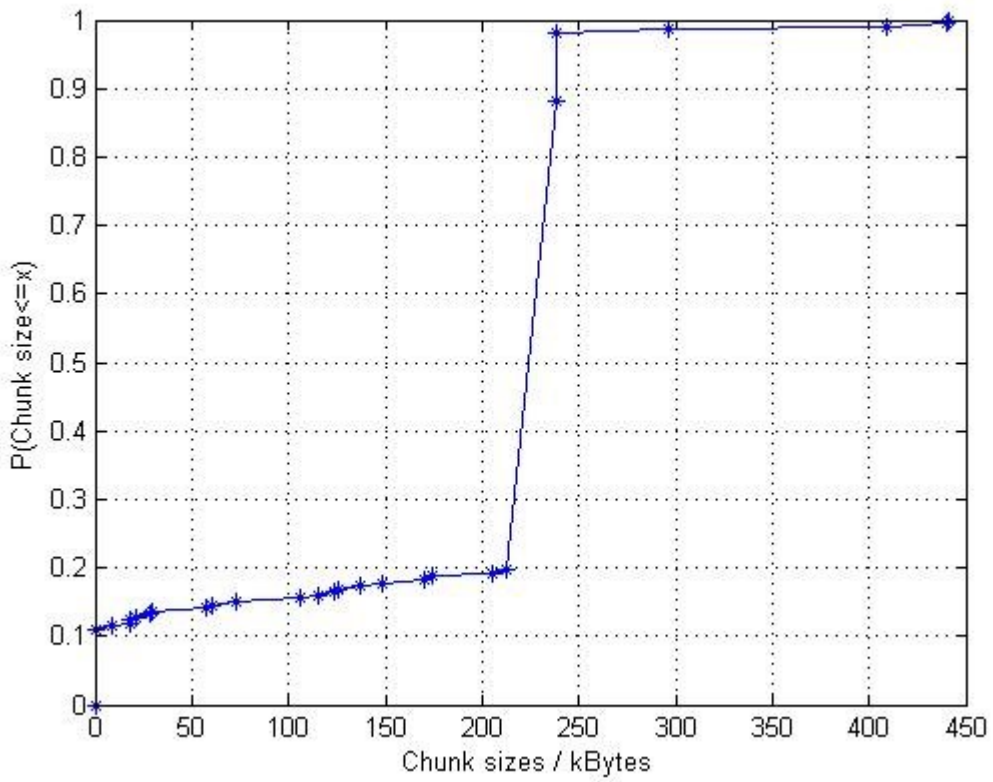

Figure 24: Empirical cumulative distribution of the low stream chunk sizes in the steady phase.

Table 10: Cumulative probabilities for the low stream chunk sizes in the steady phase

Chunk size below (kBytes)	50	100	150	200	238.7675	238.7676	240	300	400	450
Probability (%)	13.7	15.1	17.8	18.7	19.6	88.1	98.2	98.6	98.6	100

Next were analyzed the time differences between the low stream chunks. The results are shown in Figure 25. All the median values were very close to each other and the median of medians was 14.91 seconds. The median value for clip 2 was 13.97 seconds which slightly differs from the general trend but, nevertheless, the values were much more tightly packed than the median difference values for the high stream chunks. It was additionally observed that, although clips 1 and 8 had longer high stream median values, the same did not appear in the low stream chunks.

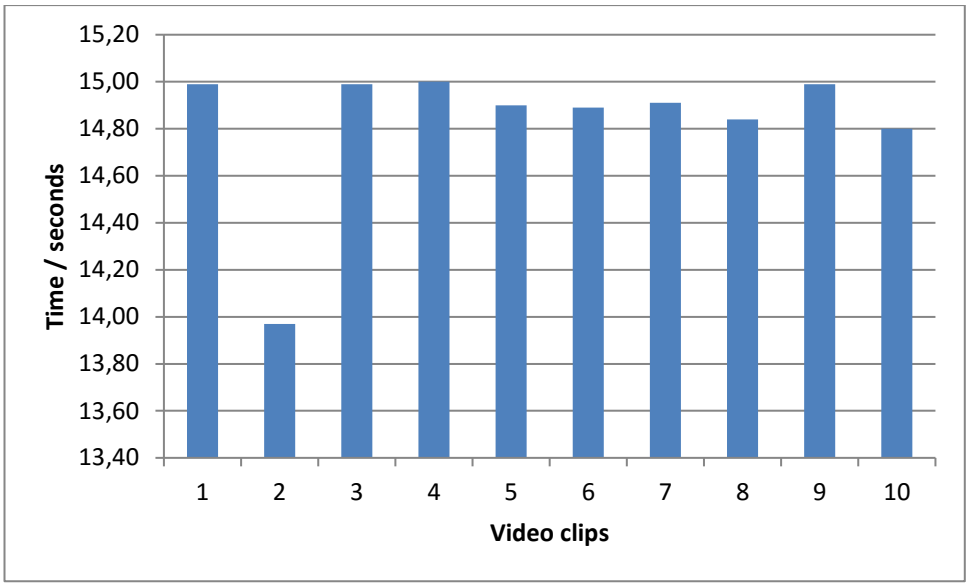

Figure 25: Median of time difference of low stream chunks

Because clip 2 median differed from the other values by 1 second, it was good to observe how two major TCP streams looked like in this clip. This can be seen in Figure 26.

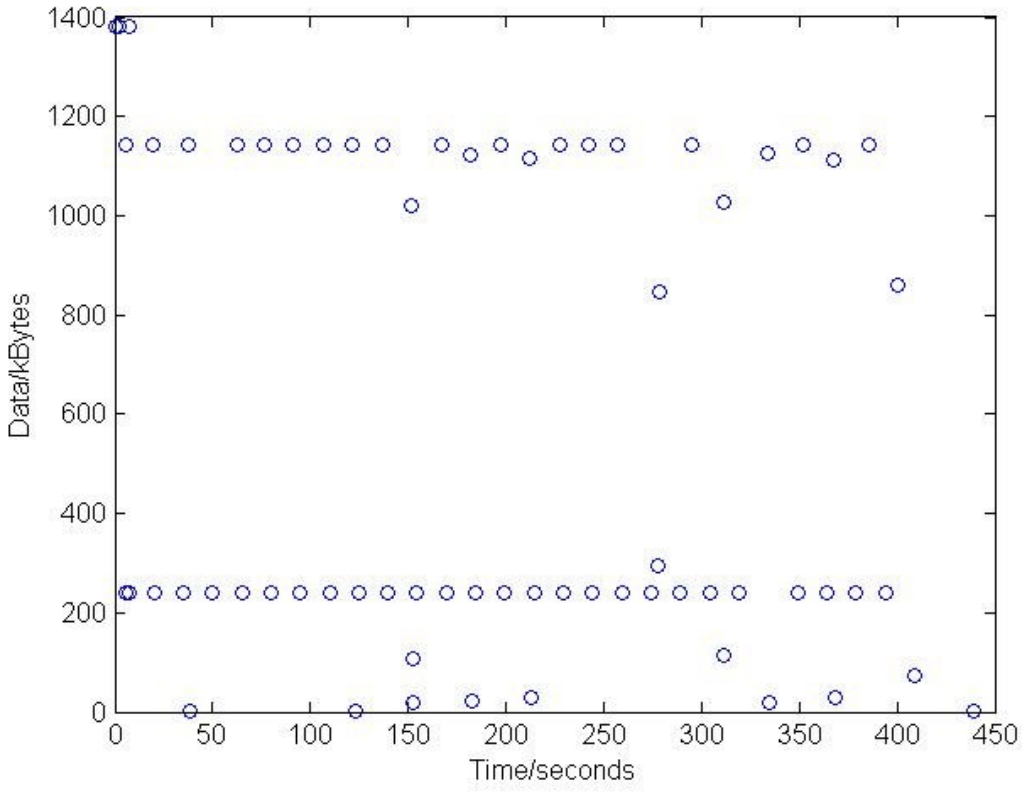

Figure 26: Clip 2 chunks for two major TCP streams

There were quite many variations both in the high and the low streams. A new median difference value for the clip 2 low stream was calculated using only the chunk sizes between 488 kBytes and 195 kBytes. This was done in order to remove unwanted small chunks visible in Figure 26. This way the median value was 14.9 seconds (average was

14.4 seconds). So also in video clip 2 the low stream median was 14.9 seconds, but it additionally contained some chunks with less data that caused the original median value to differ from the rest of the values. The last low stream chunks appear at the same time as the last high stream chunks, so the same formula can be used as for the high stream chunks: the last low stream chunk appears approximately when 70 % of the video has been viewed.

The empirical cumulative distribution of the time differences during the steady phase in the low stream chunks is in Figure 27 and is marked with red colour. Approximately 70 % of the time differences were very close to the median 14.9 seconds and the average 14.1 seconds. In the same figure the normal cumulative distribution is marked with blue colour and quite a large mismatch to the ECDF is visible. RMSE was 0.2.

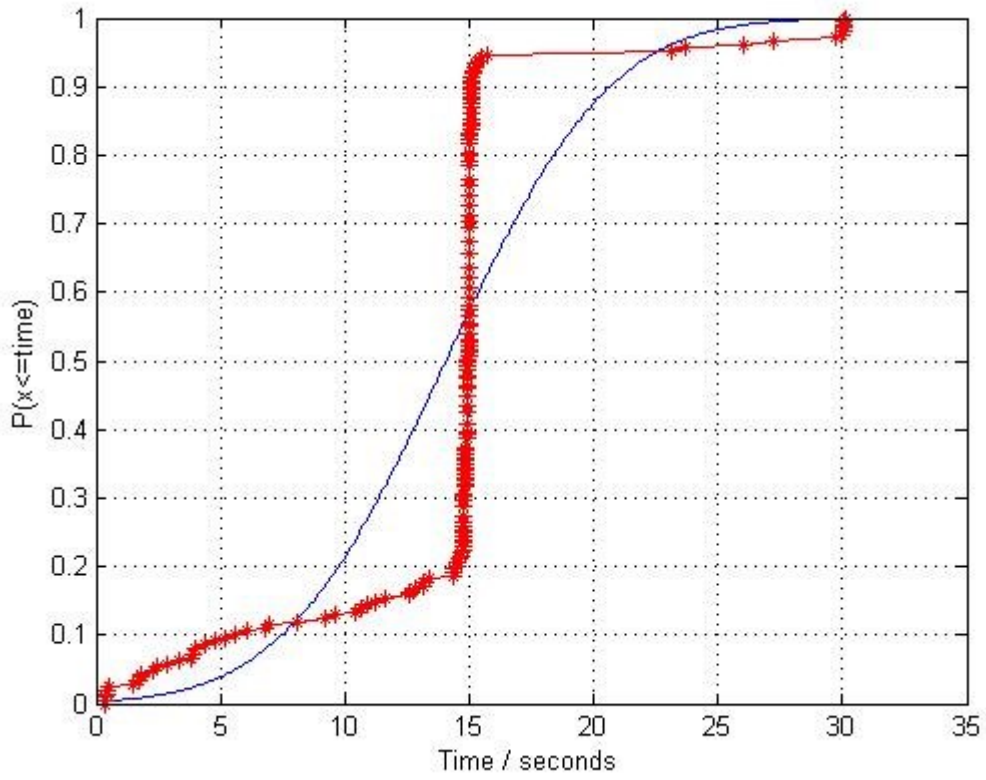

Figure 27: Empirical cumulative distribution of the low stream chunk time differences in the steady phase in red colour. Normal cumulative distribution is in blue colour with average value of 13.942 and standard deviation of 5.17325.

In addition, between 16-23 seconds there is not a single value. More detailed values of the empirical cumulative distribution are shown in Table 11.

Table 11: Empirical cumulative distribution of the time differences between the low stream chunks in the steady phase.

Time difference below (seconds)	0.325	2.5	5	10	14	14.7	14.8	14.9	15.0	15.1	15.2
Probability (%)	0	5.3	9.1	12.9	18.2	22.0	26.3	40.7	67.5	86.1	91.9
Time difference below (seconds)	15.3	16	23	24	26	27	28	29	30	30.2	
Probability (%)	93.3	94.7	94.7	95.7	95.7	96.2	96.7	96.7	97.6	100	

Because the two streams of data chunks (high and low) were clearly visible, it could be possible that one of them is for audio and another one for video. The bit rate of the lower stream was roughly 244498 bytes/14.91 seconds = 131186 bits/seconds. This is very close to the audio bit rate of 128 kbps for 360p resolution videos used by YouTube after March 2011 [44], [45]. Additionally, video clip 2 had a different sound than the other videos because it contained only outdoor and indoor background noise and no direct speech or music. This could explain why the clip 2 low stream data looked a little different from the rest. For the high stream the bit rate was 1147 kBytes/15.41 seconds=610 bits/seconds in the steady phase. This is much lower than 1000 kbps which is a maximum rate for video codec in this resolution [45]. It could be assumed here that our video samples didn't make use of all the video bandwidth and video codec can heavily compress the video data.

4.3.2.3 Speedup phase statistics for major streams

The study included observing speedup phase statistics for the two major streams. Because the chunks here were large, the higher and lower streams were not differentiated but all the values during the speedup phase were added up. First was examined the time differences between the chunks and the results are shown in Figure 28. Empirical cumulative distribution is in red colour, normal cumulative distribution is in green and exponential cumulative distribution in blue colour. The normal cumulative distribution gives quite good match with RMSE of 0.0569 while the exponential cumulative distribution had RMSE of 0.1284.

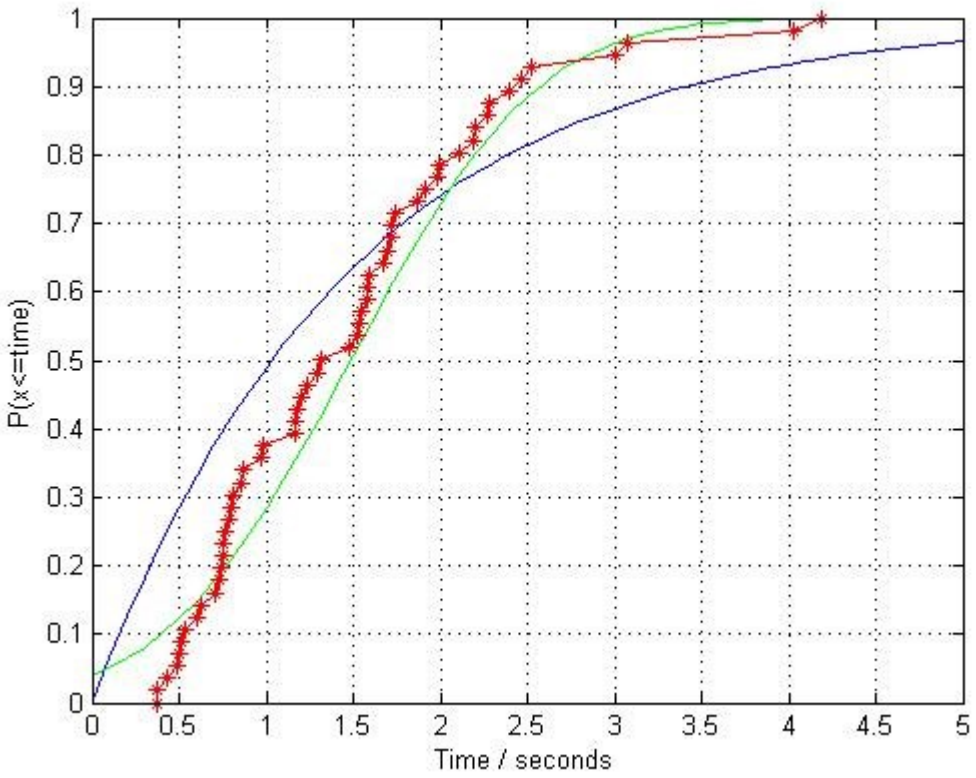

Figure 28: Major streams time differences between the chunks in the speedup phase. Empirical cumulative distribution in red colour, normal cumulative distribution (average=1.4828, standard deviation=0.8470) in green colour and exponential cumulative distribution (average=1.4828) in blue colour.

And exponential cumulative distribution can be calculated using the equation:

$$p = F(x|\mu) = \int_0^x \frac{1}{\mu} e^{\frac{-t}{\mu}} dt = 1 - e^{\frac{-x}{\mu}} \quad (8)$$

where p is the probability, μ is mean and σ is standard deviation [46], [47]. More detailed probability distribution is presented in Table 12. Over 78 % of the time differences were below 2 seconds.

Table 12: Empirical cumulative distribution of the time differences between all the chunks in the speedup phase.

Time difference below (seconds)	0.37	0.5	0.75	1	1.25	1.5	1.75	2	2.25	2.5
Probability (%)	0	5.4	21.4	37.5	46.4	51.8	71.4	78.6	83.9	91.1
Time difference below (seconds)	2.75	3	3.25	3.75	4	4.19				
Probability (%)	92.9	92.9	96.4	96.4	96.4	100				

The empirical cumulative distribution of the chunks sizes in the speedup phase is in Figure 29 in red colour. Normal and exponential theoretical distributions are also plotted in that figure. RMSE for normal cumulative distribution was 0.1193 and for exponential cumulative distribution 0.1371.

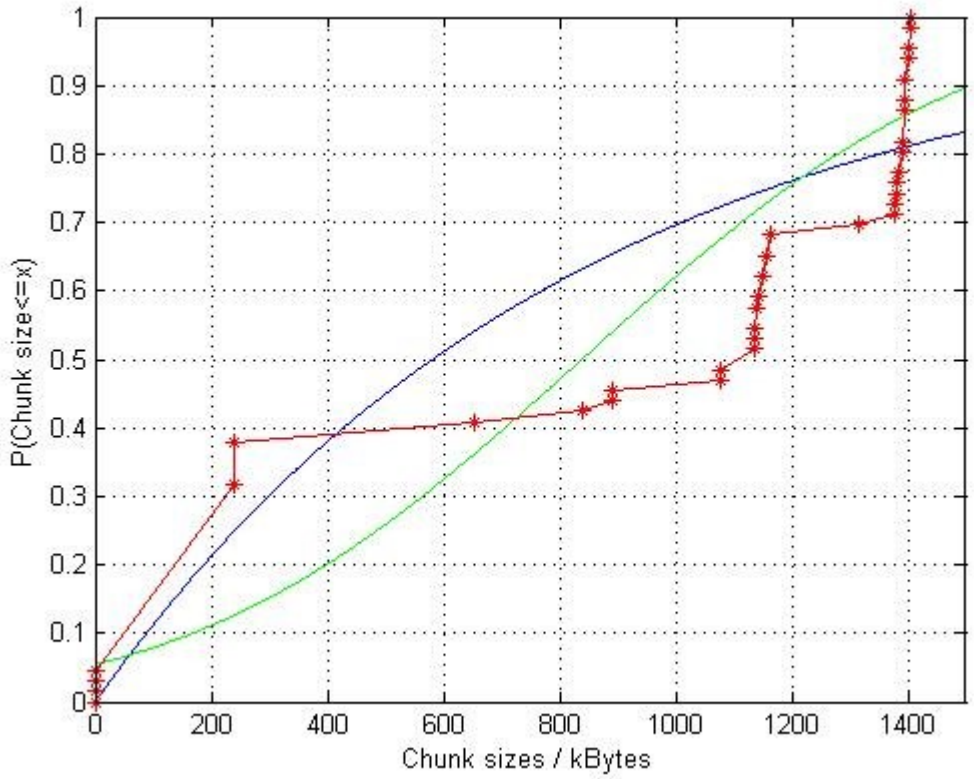

Figure 29: Empirical cumulative distribution of the chunk sizes in the speedup phase in red colour. In blue colour exponential cumulative distribution (average=839.686) and normal cumulative distribution (average=839.686, standard deviation=523.5998) in green colour.

A more detailed distribution is in Table 13. There can be seen three distinctive size patterns: the first between 238 and 239 kBytes, the second one between 1130 and 1170 kBytes and the last one after 1380 kBytes. The first and the second pattern match well the earlier findings about sizes in the low and the high stream. The third pattern is the sum of the low and high stream.

Table 13: Empirical cumulative distribution of the major streams chunk sizes in the speedup phase.

Chunk size below (kBytes)	5	238	239	653	654	838	839	892	893	1070
Probability (%)	4.5	4.5	37.9	37.9	40.9	40.9	42.4	42.4	45.5	45.5
Chunk size below (kBytes)	1080	1130	1140	1150	1160	1170	1310	1320	1370	1380
Probability (%)	48.5	48.5	57.6	59.1	65.2	68.2	68.2	69.7	69.7	72.7
Chunk size below (kBytes)	1390	1400	1410							
Probability (%)	80.3	90.9	100							

4.3.3 Statistical evaluation of the background noise streams

All the other data except the two major TCP streams are regarded as “background noise”. This noise consisted of several small TCP streams, both UL and DL biased. YouTube web-pages use these streams e.g. to receive data for web page updates. In order to study these noise streams statistically, all the noise TCP streams from each of the 10 video clips were added together. The analysis data contained both UL and DL TCP/IP packets. There were a total of 1139 noise chunks in the data. The median value of all the chunks was 1.25 kBytes (1280 bytes). 7340542 bytes were transferred in DL and 1501626 bytes in UL. UL/DL proportion was 20.4 %, which is much higher than the median of the two major TCP streams, which was 2.1 %. This also reveals that in UL something else but TCP acknowledgements were sent, or DL packets are much smaller than the normal maximum TCP/IP packet size 1500 bytes. In UL, there were transmitted 8378 TCP/IP packets and in DL were received 9404 TCP/IP packets. This means that for every UL packet 1.1225 DL packets were received. An average size for a UL packet was 179 bytes, and for a DL packet an average size was 781 bytes.

To see the chunk sizes the empirical cumulative distribution was plotted and it can be seen in Figure 30. In this figure it is very clear that most of the chunks were under 3000 bytes, and there were actually only 7 chunks which were larger than 200 kBytes.

The largest chunk was 929 kBytes. Actually, 1079 chunks of the total 1139 chunks were smaller than 20 kBytes and still below 2 kBytes were 726 chunks which were 64 % of all the chunks. The average value was 7763.1 bytes. In the same figure can be seen exponential cumulative distribution in red colour with an average value of 1349.8 bytes as well. To get this average value only the chunk sizes below 5000 bytes were included. This distribution does not exactly match the data but it gives a good approximation for smaller packet sizes. RMSE was 0.096.

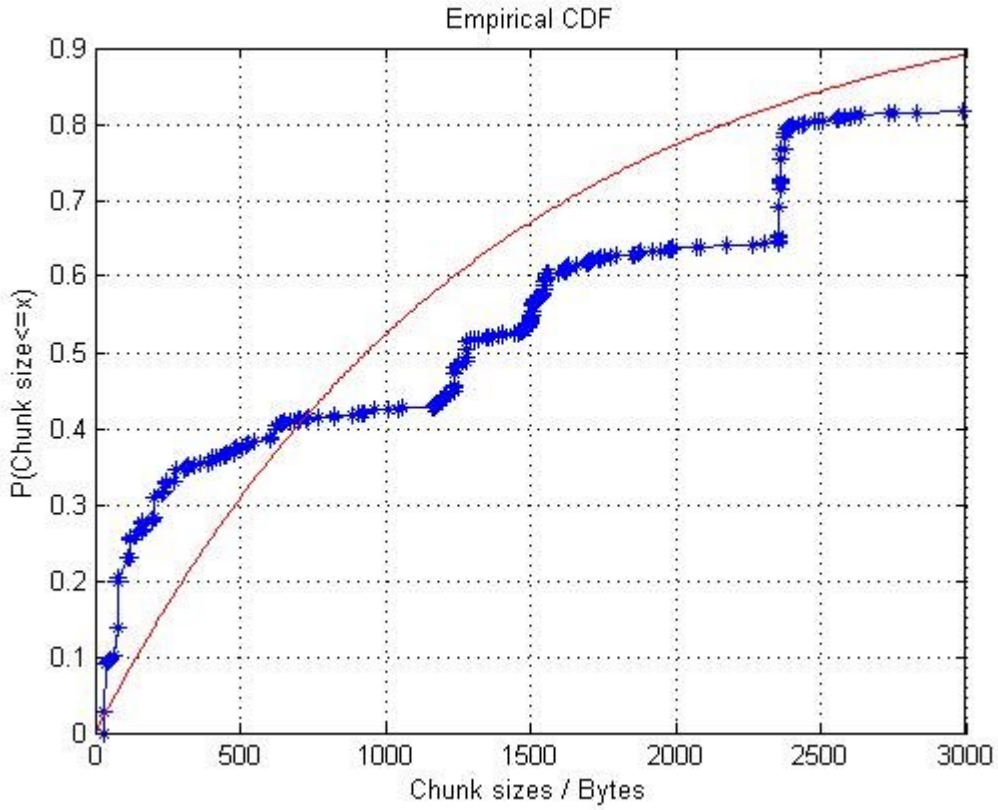

Figure 30: Empirical CDF of the noise chunk sizes between 0 - 3000 bytes in blue. In red exponential cumulative distribution with average value of 1349.8 bytes. Data includes both UL and DL.

To get more accurate probabilities for the different chunk sizes, they were calculated from the measurement results. Probabilities for the first 9000 bytes beginning with 100 bytes differences are presented in Table 14. The probability that chunk size was under 2000 bytes is over 63.7 % and the table shows that over 20 % of chunks are less than 100 bytes. Higher probabilities are also seen for 1200, 2400 and 4800 bytes.

Table 14: Probabilities for chunk sizes 0-9000 bytes

Chunk size (bytes)	0-100	100-200	200-300	300-400	400-500	500-600	600-700	700-800
Probability %	20.3	7.6	6.7	0.8	2.0	0.8	2.8	0.4
Chunk size (bytes)	800-900	900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600
Probability %	0.4	0.6	0.3	0.9	8.1	0.6	1.8	6.4
Chunk size (bytes)	1600-1700	1700-1800	1800-1900	1900-2000	2000-2100	2100-2200	2200-2300	2300-2400
Probability %	1.4	0.8	0.5	0.5	0.2	0.1	0.2	15.5
Chunk size (bytes)	2400-2500	2500-2600	2600-2700	2700-2800	2800-4700	4700-4800	4800-6000	6000-9000
Probability %	0.7	0.5	0.4	0.2	1.2	7.5	1.2	1.5

Table 15 presents probabilities for the chunk sizes between 0 – 400000 bytes. Probability that a chunk size was less than 10000 bytes is 93.9 % and less than 400000 bytes 99.9 %. So there can be some single chunks with a large amount of data but probability for that is very small.

Table 15: Probabilities for chunks sizes 0-1000000 bytes

Chunk size (10000 bytes)	0-1	1-2	2-3	3-4	4-5	5-6	6-7	7-8
Probability %	93.9	0.9	0.4	1.1	0.5	0.1	0.4	0.2
Chunk size (10000 bytes)	8-9	9-10	10-11	11-12	12-13	13-14	14-15	15-16
Probability %	0.2	0.2	0.1	0.1	0.3	0.2	0	0.4
Chunk size (10000 bytes)	16-17	17-18	18-19	19-20	20-25	25-30	30-40	
Probability %	0	0.3	0.1	0	0.2	0.2	0.2	

To study the time differences between the chunks, a histogram was plotted and it can be seen in Figure 31. 82 % of the time differences were under 5 seconds long and

1.7 % were longer than 15 seconds. The maximum time difference was 29.6 seconds and median value was 1.7 seconds.

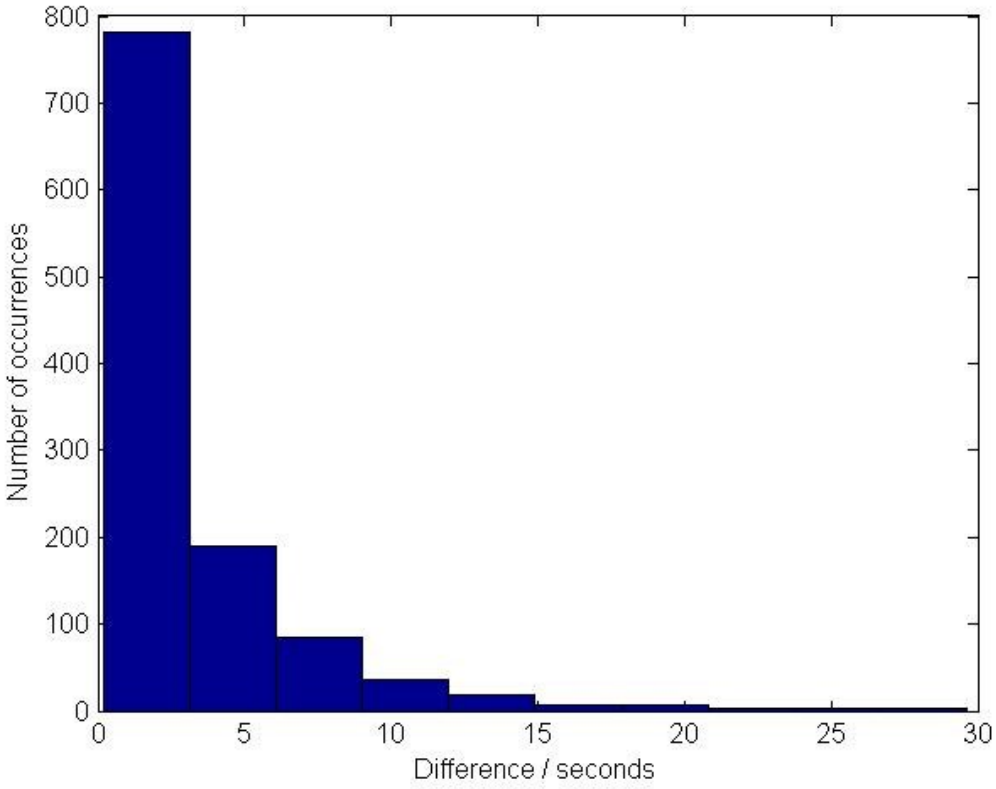

Figure 31: Histogram of time differences between noise chunks

The measurement data of time differences between noise chunks can be treated as exponentially distributed with an average value of 3.08067 seconds. Cumulative distribution function of this exponential distribution is calculated and plotted in Figure 32. In the same figure the colour red symbolizes the empirical cumulative distribution. RMSE was 0.045.

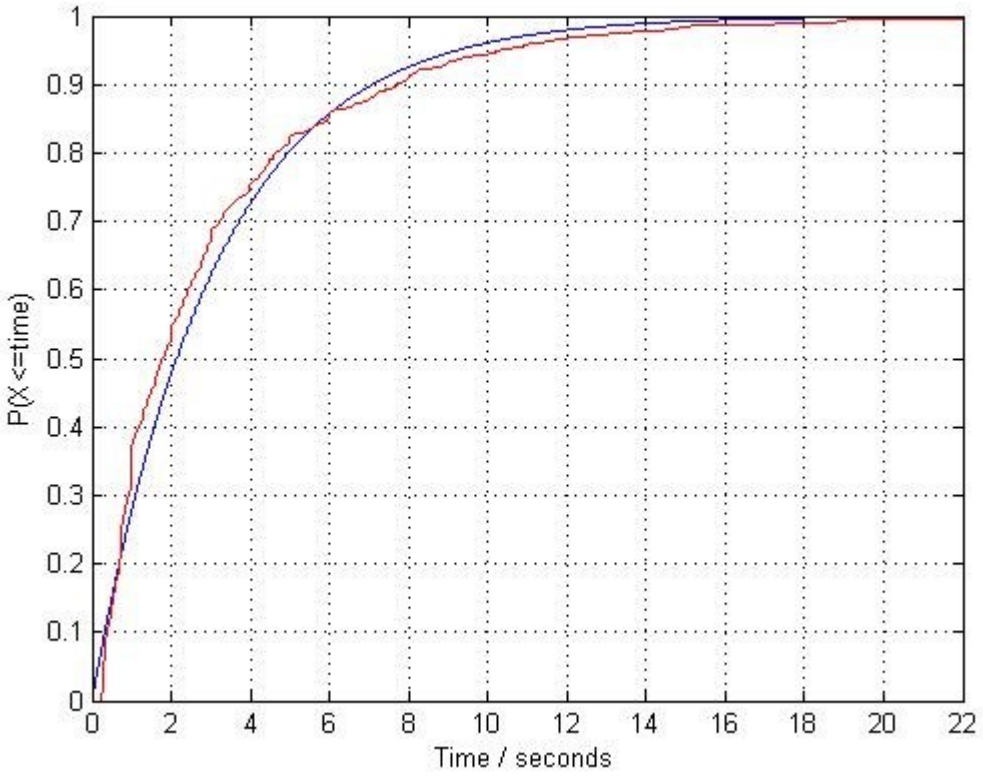

Figure 32: Cumulative distribution of the noise chunk time differences using exponential distribution with average value of 3.08067 seconds in blue. In red is plotted empirical cumulative distribution.

Results from this exponential distribution were compared to the actual measured cumulative probabilities and are presented in Table 16. Values from the measured probability as well as from the exponential distribution are very close to each other as RMSE 0.045 shows. There is a constant difference of 0.06 with small time differences but after 4 seconds the difference is only 0.02 and later only 0.01. So, it seems that exponential distribution gives quite a good approximation of the time differences, but, at the same time, it gives slightly too low probabilities for small time differences and slightly too high probabilities for high time differences. More measurements would be needed to see if this is a trend or just a coincidence.

Table 16: Comparison of measured empirical cumulative probabilities with exponential distribution probabilities

Time difference seconds	Cumulative probability value from measurement	Cumulative probability value of exponential distribution	Difference (measurement – distribution)
1	0.34	0.28	0.06
2	0.54	0.48	0.06
3	0.68	0.62	0.06
4	0.75	0.73	0.02
5	0.82	0.80	0.02
6	0.85	0.86	0.01
10	0.944	0.946	-0.002

4.4 Short summary of LAN

In the measurements were not noticed traffic patterns described in [7], [9], [11] and [32]. In all of those studies were seen 64-kByte data bursts sent by YouTube server, and usually the pauses between the bursts were quite short, i.e. 0.5 seconds or less. Since the results were completely different, the YouTube data profile was studied here extensively including statistical results. A more detailed summary of the YouTube data profile can be found in Chapter 6.1.

The two TCP streams carry 97 % of the whole data. There is a speedup phase at the beginning of the transfer and the transfer covers 20 % of the total data. The chunk sizes are much larger in the speedup phase than during later phases, and the speedup phase lasts around 10 seconds. A steady phase follows next, and there can be seen larger (high stream) and smaller (low stream) chunks. The time differences between the high stream chunks are usually around 15 seconds but they can be higher in case of a small video file size compared to the video length. One high stream chunk contains 4 % of the total DL data. The low stream chunk size is around 244498 bytes and the time differences between the chunks are close to 14.9 seconds. Both the high and the low stream end when approximately 74 % of the video viewing time has elapsed. After that, there can be seen only a few major streams chunks irregular in size. The remaining 3 % of the data is transferred in several small TCP/IP streams. These chunks are very small compared to the two major TCP streams. Most of them are under 2 kBytes. Here the time differences between the chunks are exponentially distributed with the mean value of 3.08067 seconds.

5 YOUTUBE TRAFFIC PATTERNS IN AN LTE TEST NETWORK

This chapter presents the measurements which were done over the LTE test network in Department of Electronics and Communications Engineering in Tampere University of Technology. First, the measurement setup is briefly explained and after that the LTE results are presented. Finally, LAN and LTE measurement results are compared with each other.

5.1 LTE measurement setup

The measurements were done in Nokia's LTE test network in the premises of Department of Electronics and Communications Engineering in Tampere University of Technology at the end of June 2014. A Windows XP laptop recorded the logs and *Microsoft Network Monitor* [48] was used to capture TCP/IP logs. *Wireshark* tool was not used because it did not function with the used LTE Universal Serial Bus (USB) modem. Additionally, special *Nemo Software* from Anite [49] was used to capture the radio link logs directly from the USB stick. The used modem was Huawei LTE USB stick E398 [50]. The measurements were done stationary and radio field conditions were excellent during the measurement to get comparable results with LAN. YouTube measurements in poor radio conditions or during the movement were left for further studies.

The same YouTube videos were recorded as with the LAN measurement except the video clips 9 and 10, because the original video clips 9 and 10 did not play properly in this setup. The reason for this is unknown. The same kind of video clips were searched and used instead. If there were advertisements shown before the actual video playback, the measurements were repeated until clean playback was received.

5.2 LTE statistical examination

Using *Matlab*, 10 different YouTube log files were recorded and analyzed in the same way as it was done with LAN: Analysis was done for the following parts: full original file, two major TCP streams only, "noise" part only, in which the two major TCP streams were filtered out.

5.2.1 Full file

The sum of the lengths of 10 video clips was 3352 seconds and a total of 269436701 bytes were transmitted. There were 263195829 bytes in DL and 6240872 bytes in UL. It

soon became obvious that video clips in LTE followed the same patterns as already noticed during the LAN analysis in Chapter 4.2.

The ratio of DL speedup phase bytes versus all DL bytes was calculated and this is presented in Figure 33. The LTE results are in blue colour and the LAN results in red colour. The presented results match with the LAN results and the average of all DL speedup phase bytes versus all DL bytes is 19 % (it was 20 % with LAN). It is also interesting to notice, that LTE clips 2 and 7 had clearly smaller speedup phase proportions than all the rest of the clips.

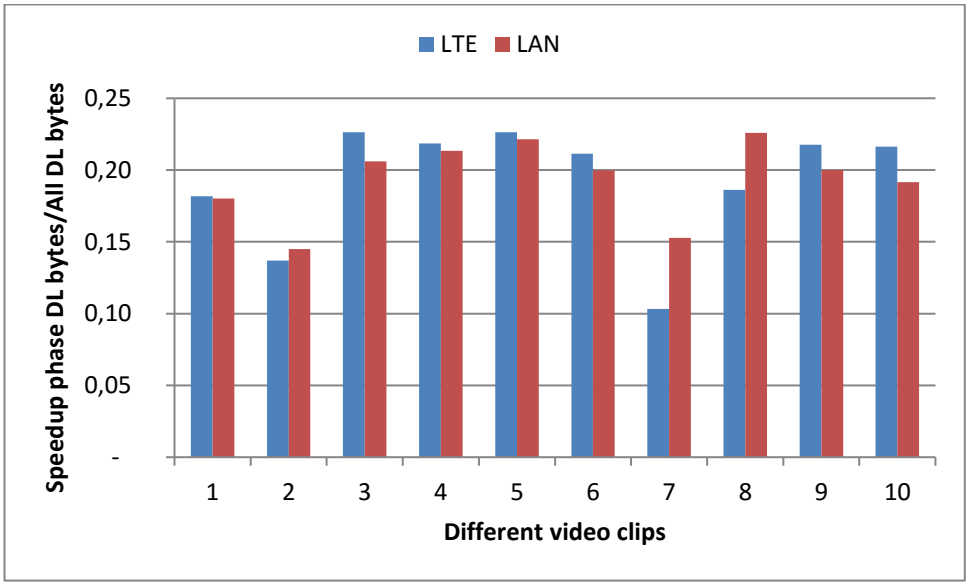

Figure 33: LTE speedup phase bytes versus all DL bytes. LTE results in blue colour and LAN results in red colour.

The average value of the speedup phase length in LTE was 9.10 seconds (in LAN it was 9.97 seconds).

Nemo SW was used to capture PDCP throughput values both in UL and DL from the USB stick. PDCP is one of the higher layers in the LTE network, and the summary of these values is presented in Table 17. It shows average, median and maximum values recorded in UL and DL. Only non-zero values in the *Nemo* log were used for these calculations. The last row in the table shows the average value of all the previous rows. Both in UL and DL the average values were very small and median values were even smaller. Maximum values in DL were at a reasonable level but maximum UL values were very small.

Table 17: PDCP throughputs in LTE. Values are Mbps.

Video Clip	Average DL	Median DL	Max DL	Average UL	Median UL	Max UL
1	1.26	0.003	12.56	0.03	0.002	0.35
2	1.99	0.006	13.19	0.05	0.006	0.45
3	1.77	0.004	11.95	0.04	0.002	0.44
4	1.96	0.55	14.41	0.04	0.004	0.47
5	2.06	0.077	12.19	0.03	0.003	0.30
6	1.66	0.013	11.17	0.04	0.004	0.29
7	1.52	0.004	11.72	0.04	0.003	0.42
8	1.15	0.009	9.50	0.03	0.003	1.12
9	1.40	0.005	14.85	0.03	0.003	0.30
10	1.83	0.119	10.50	0.03	0.003	0.39
Average	1.66	0.079	12.20	0.04	0.003	0.45

As an example, it is good to view throughput of one of these clips versus time. This is presented in Figure 34. This figure shows clip 7 throughputs versus time, and it contains both UL and DL activity.

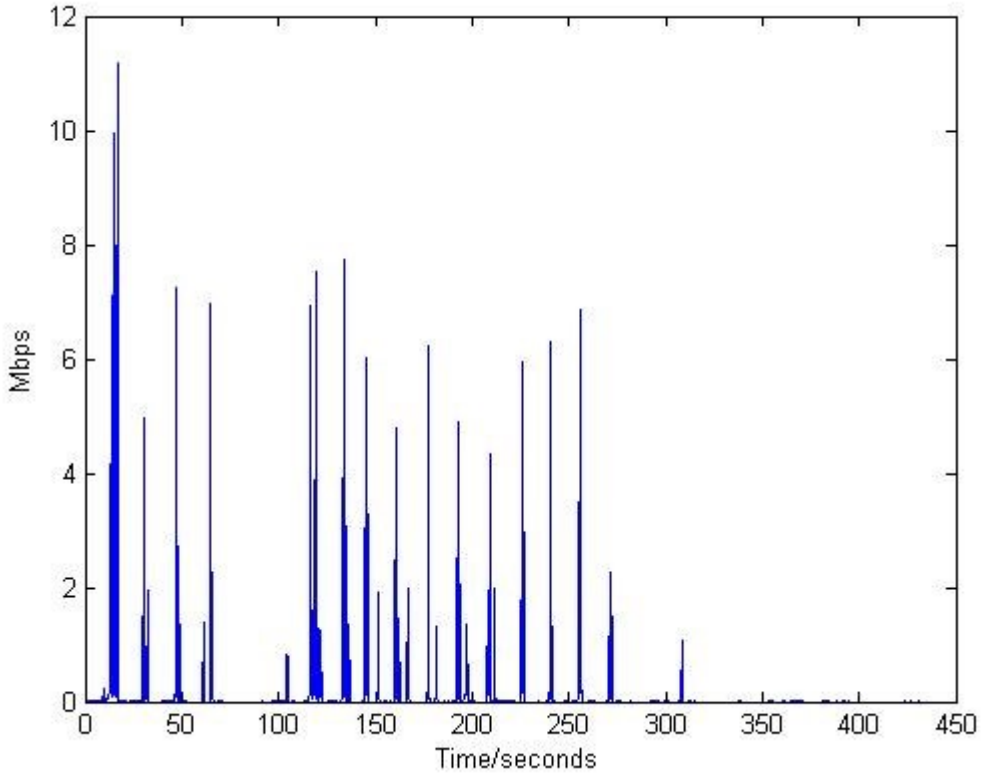

Figure 34: Measured PDCP throughput versus time for clip 7

In this figure, it can be noticed that in the speedup phase throughputs in the PDCP level were much higher than later. For example, here the maximum value was 11.72 Mbps but later all the values were clearly under 8 Mbps. Several reasons can influence on this

result; e.g. *Nemo* sampling period average was only 0.47 seconds and median was exactly 0.5 seconds. Because after the speedup phase data appears in short bursts, this can affect the recorded throughputs heavily and downgrade them. So, there is a reason to believe that the recorded maximum values give a more accurate result for the throughput capacity estimation of the network. This was also verified in *Wireshark* logs, where DL throughput values around 10 Mbps were seen constantly. An estimation of the maximum UL throughput is more difficult, because video transmission is DL biased and there are no long periods of constant UL transmission. The average value of the maximum DL throughput was 12.2 Mbps and for UL it was 0.45 Mbps in *Nemo* PDCP logs.

5.2.2 Major TCP streams

The rest of the data was filtered out except the two most dominant TCP streams. This was followed by calculating the proportion of all data in these two major streams from the whole of the data transferred. The average value for this proportion was 99 % (for LAN measurement it was 97 %). Ratio of DL IP packets and UL IP packets was 1.73, which basically means that for every 17 DL TCP/IP packets there are 10 UL TCP/IP packets.

Like with LAN, TCP/IP packet time differences were calculated inside the chunks. Values from all 10 clips were used and results include both UL and DL packets. Total of 291850 values were used in the calculations and Figure 35 presents the cumulative distribution of the results. In this figure the LAN differences are in red colour and the LTE differences in blue.

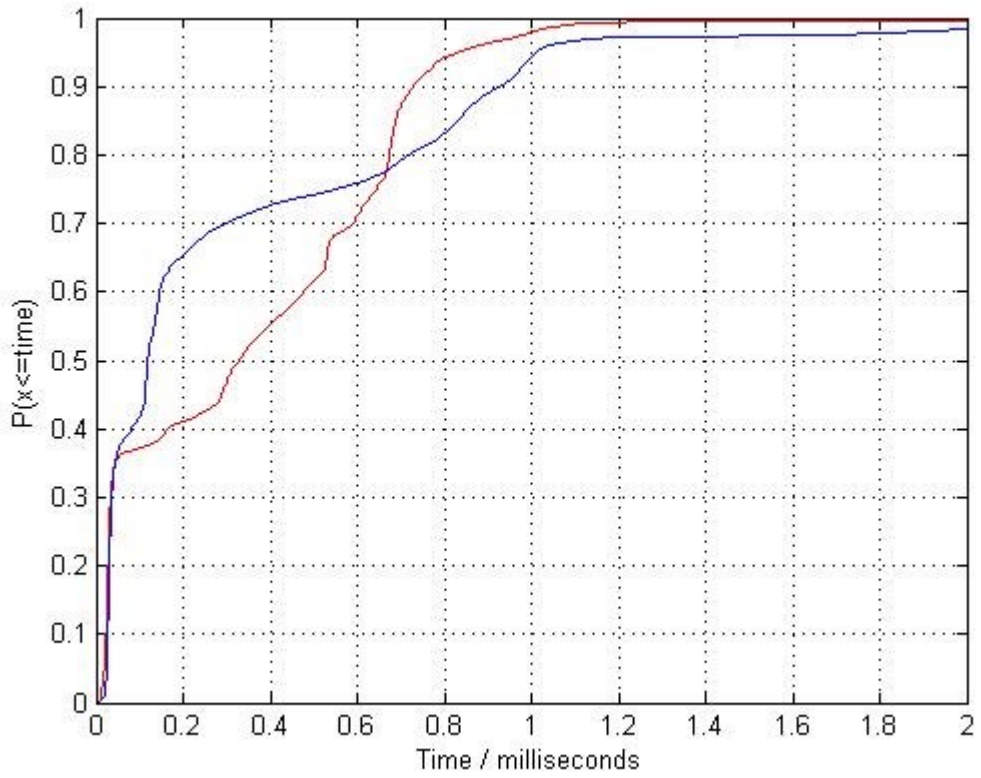

Figure 35: Cumulative distribution of LTE TCP/IP packet time differences inside the chunks. LTE is in blue colour and LAN in red colour.

Like in LAN, the packets inside the chunks arrived in very short intervals in LTE. Average value was 0.5 ms and median 0.12 ms (corresponding LAN values were 0.42 ms and 0.33 ms). The maximum value was 199 ms, because a greater difference means that the packets belong to the other chunks. The median difference with LTE was much lower than with LAN and ECDF also showed that the probability for smaller values was larger than with LAN. This could be explained with LTE's radio network features, IP packets are buffered in 3GPP layers until radio is ready to transmit or to receive and packets are then transmitted with minimum intervals. Table 18 shows detailed cumulative probability values for LTE. Furthermore, it is noticeable that in LTE there were also some larger values and approximately 3 % of the packets had a greater time difference than 1.2 ms. This was not discoverable in LAN. In LTE a low level hybrid-ARQ retransmission takes 8 ms as a minimum value and, for 0.6 % share of the packets, the time difference was over 8 ms.

Table 18: ECDF of packet time differences inside chunks in LTE

Time equal or below (ms)	0.03	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.2
Probability (%)	18	42	65	70	73	74	76	79	83	89	97
Time equal or below (ms)	1.7	1.8	1.9	2.0	2.1	2.8	3.7	5.9	10.9	28.5	40
Probability (%)	97.5	97.8	98.1	98.5	98.8	99.0	99.2	99.4	99.6	99.8	99.9

The time differences between the chunks varied in the same way as with LAN. The median value of all the median differences of the clips was 5.17 seconds, whereas the average value was 6.56 and the median of standard deviations was 5.67 seconds. These were in the same level as with LAN.

Since there were clearly two chunk streams (higher and lower), values were calculated separately for both streams. Chunk sizes over 500000 bytes (488 kBytes) were considered belonging to the upper stream and other chunks to the lower stream. This is the same as was done with LAN measurements. Figure 36 reveals the median value for LTE in blue, the average value for LTE in red, the median value for LAN in green and the average value for LAN in violet. The LTE values are very similar with the LAN ones. In addition, here it is evident that the clips 1 and 8 had lower values than the rest of the clips.

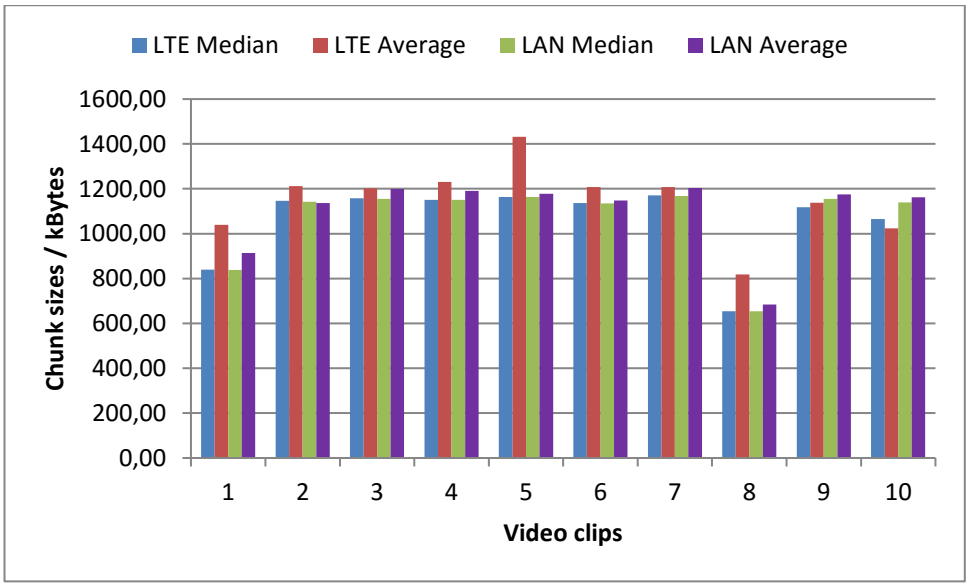

Figure 36: Median and average values of high stream DL chunk sizes. LTE median in blue, LTE average in red, LAN median in green and LAN average in violet.

Like with LAN, median of DL high stream chunk sizes were compared to total DL transferred data, this is presented in Figure 37, in which LTE median is in blue and LAN in red. Again, we can see that the high stream chunk size is about 4 % of the total DL data amount. When comparing to the same results with LAN, a great correlation between the video clips becomes apparent. The average value was 0.041 with LAN and with LTE it is now 0.043. N.B. in LTE clips 9 and 10 had different videos than in LAN.

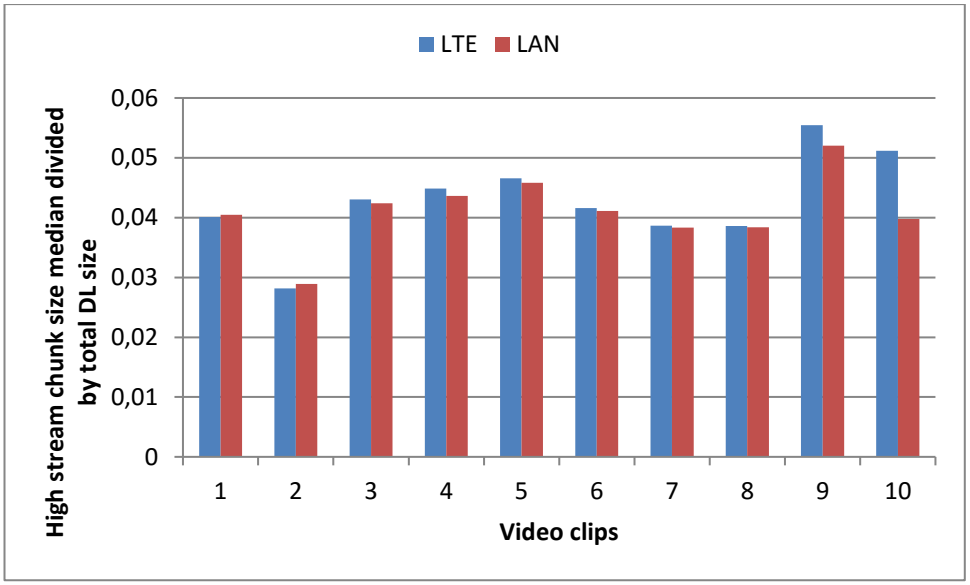

Figure 37: High stream DL chunk size median divided by total DL video clip size. LTE is in blue colour and LAN in red colour.

The time differences between the high stream chunks were calculated next. Like with LAN the median values were analyzed more closely. Figure 38 displays the results, LTE results in red and LAN in blue.

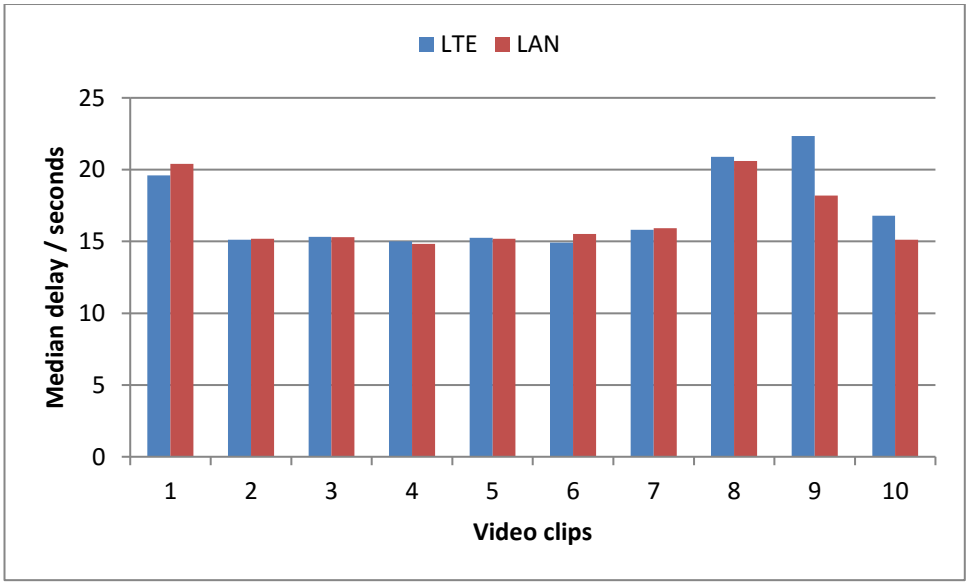

Figure 38: Median delay between high stream chunks in LTE. LTE is in blue colour and LAN in red colour.

There can be seen that in seven video clips the median value of the time difference was very close to 15 seconds and three clips had median values close to or a little higher than 20 seconds. The median of medians was 15.57 seconds for LTE (15.41 seconds in

LAN). Like with LAN, clips 1 and 8 transmitted least bytes per second during the viewing period. Notice again that clips 9 and 10 were different in LAN and LTE.

Next calculations point out the proportion of the last full chunk time to total transmission time of the clip. These proportions are in Figure 39, again LTE in blue and LAN in red colour. This proportion varies from 0.71 – 0.82 and the average value is 0.74, which is exactly the same value as with LAN. We can see here that, although clips 1 and 8 had longer median differences between the chunks, they come to end conveniently after 70 % of the video has elapsed. Additionally, video clip 2 has exactly the same value 0.82 as it had with LAN.

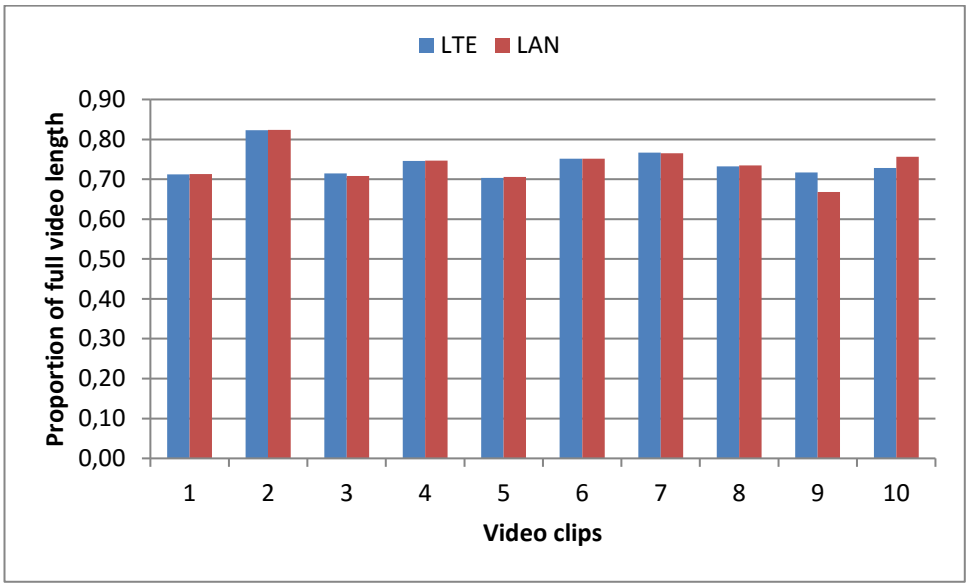

Figure 39: Proportion of the last high stream chunk of total video length. LTE is in blue colour and LAN in red colour.

The chunk sizes for the low stream were calculated, too. These are presented in Figure 40, where the LTE median is in blue, the LTE average in red, the LAN median in green and the LAN average in purple. Unlike with LAN, here can be noticed variation in the median values. The median value is 239 kBytes except for clips 4 and 9. Moreover, clip 9 has exactly the same average as the median value. Additionally, the LTE average values are clearly lower than in the LAN measurements.

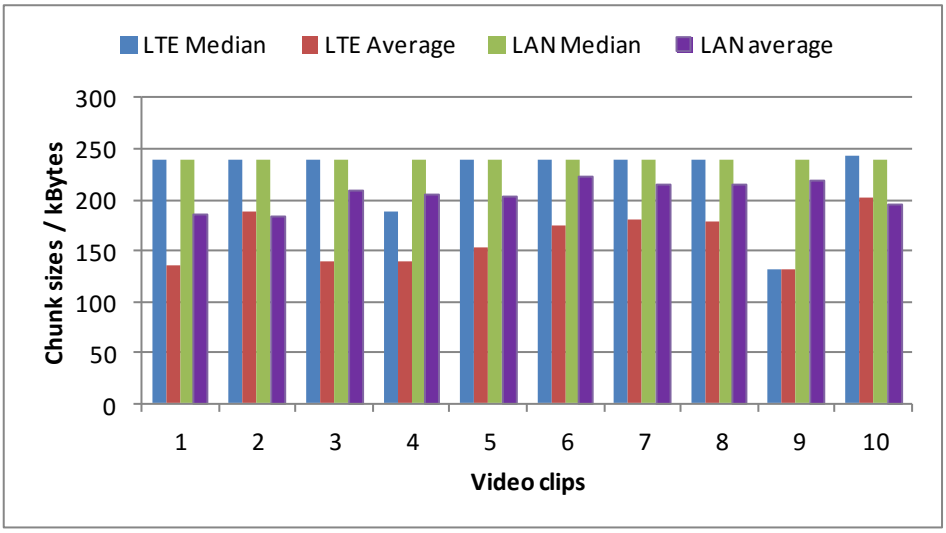

Figure 40: Median and average values of low stream chunk sizes with LTE. LTE median in blue, LTE average in red, LAN median in green and LAN average in purple.

We can notice the reason for this in Figure 41, which presents the two major streams in clip 9. The low stream is in red and the high stream in blue.

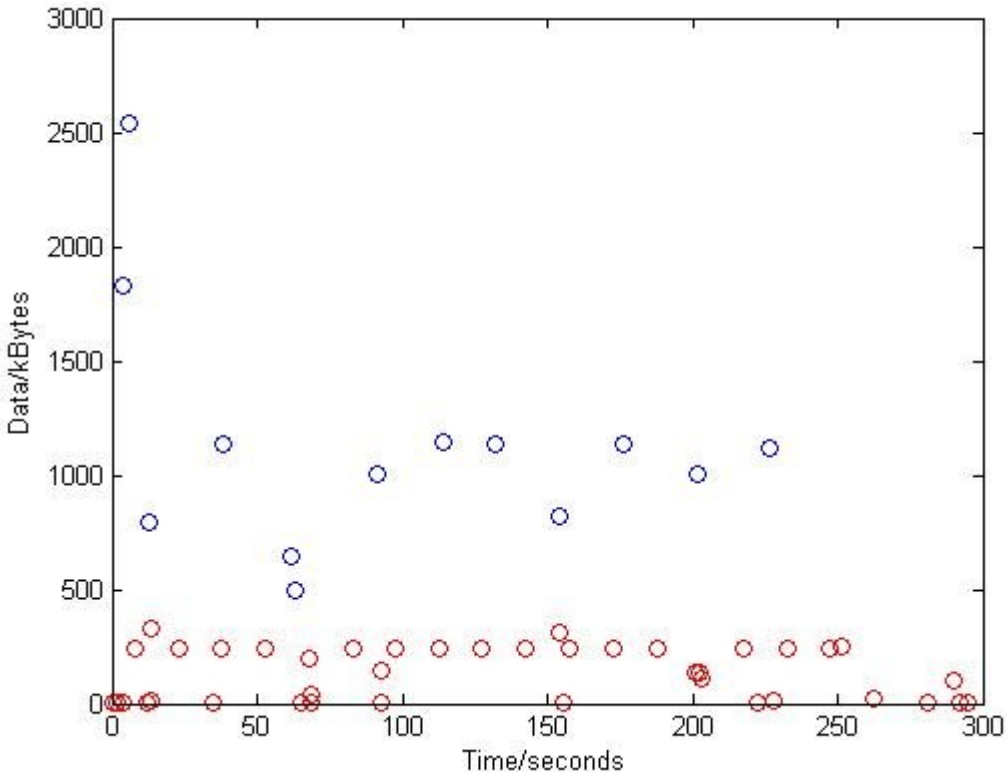

Figure 41: Two major TCP streams in LTE clip 9. High stream is in blue colour and low stream in red colour.

There was unusually much variation in the high stream and low stream and there were several very small chunks to be spotted. It is possible that during the measurements the chunks were broken into small pieces due to the LTE network delays. If all the chunks below 100 bytes were discarded, the median value for the low stream was again exactly 239 kBytes. Generally, in LTE there were more variations noticed than in LAN. Figure

42 shows standard deviations of both the high and low streams. LTE high stream STD is in red, LAN high stream STD in blue, LTE low stream STD in green and LAN low stream STD in purple.

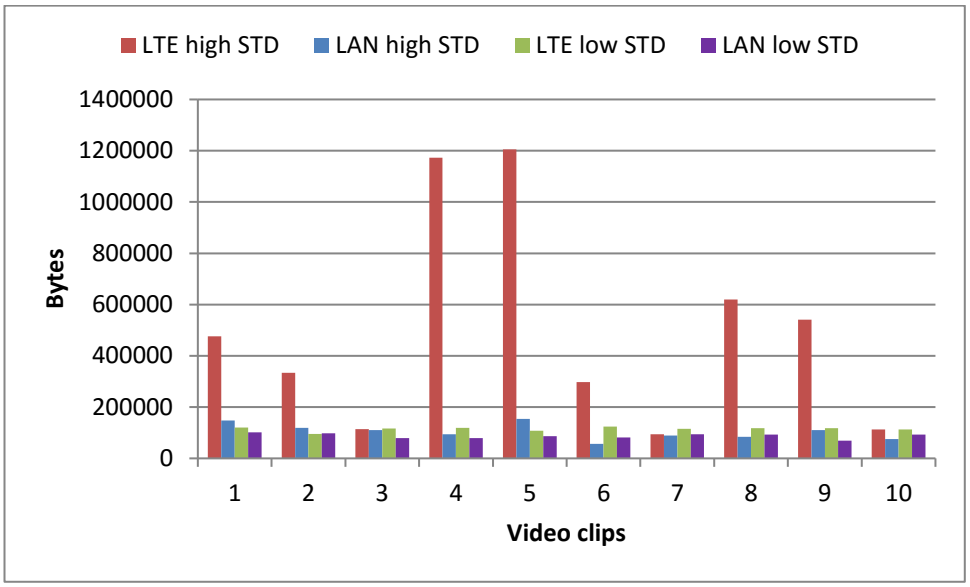

Figure 42: Standard deviations of chunk sizes in major streams. LTE high stream STD in red, LAN high stream STD in blue, LTE low stream STD in green and LAN low stream STD in purple.

Very high LTE STDs of the clips 4 and 5 can be explained by one very big chunk during the speedup phase. It was more probable that chunks were combined and large chunks were formed during the speedup phase in LTE than LAN, because the throughput in LTE was smaller. With all the clips both the low and high stream chunk sizes had more variation in LTE than in LAN.

Next calculation dealt with delays between the low stream chunks. These are pointed out in Figure 43, in which blue stands for LTE results and red stands for LAN results. In LAN these results were much more consistent and the median of median values was 14.91 seconds, but with LTE it was only 10.82 seconds. Clips 1, 2, 3 and 7 are quite close to each other, but clips 4, 5, 6 and 8 are considerably different and clip 9 had the median value of only 4.89 seconds. N.B. in LTE clips 9 and 10 had different videos than in LAN. Once again, if the chunk sizes below 100 bytes were discarded, the median value of clip 9 rose up to 10.74 seconds. All in all, there were more variations in the time differences between the major stream chunks in LTE than there were with LAN. The reason could be the way how LTE buffers and then transmits packets. The radio channel is not always open and there can be delays between TCP/IP packets.

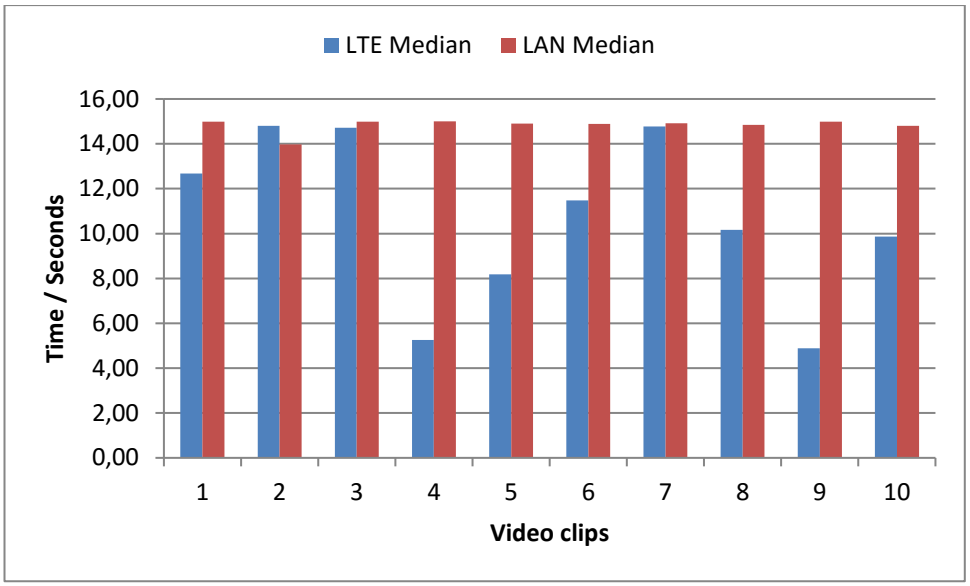

Figure 43: Median time differences between lower stream DL chunks. LTE is in blue colour and LAN in red colour.

5.2.3 Background noise streams

Like with LAN, background noise streams were analyzed for LTE. Apart from the two major TCP streams, the rest of the data is regarded as "noise". All the data from all 10 video clips were added together before analysing and the analysis data contained both UL and DL packets. There were a total of 2247 noise chunks in the LTE logs, which was substantially more than the LAN value of 1139 chunks. Like with LAN, most of the chunks were under 200 kBytes. The largest single chunk was 1365 kBytes. The median value of the chunk sizes was 182 bytes, which was less than LAN median 1280 bytes. 3615735 bytes were transferred in DL and 1215707 bytes in UL, which made UL/DL proportion to be 33.6 %. It is interesting to notice that although there were more chunks with LTE than with LAN, there was less data transferred (in LAN DL 7340542 bytes) and UL/DL proportion was higher (in LAN 20.4 %). This means that the chunk sizes in LTE must be considerably smaller than with LAN, and this becomes apparent in cumulative distribution too. The reason for this could be different background processes in the used computer, but possibly that is not the only explanation. A detailed clarification is left for the future studies. The empirical cumulative distribution of the chunk sizes is presented in Figure 44. Additionally the figure includes calculation of an exponential distribution with the mean value of 493 bytes and the corresponding LAN ECDF is also presented. The mean value for the exponential distribution was calculated for the chunk sizes lower than 3000 bytes and RMSE between empirical and cumulative distribution was 0.109. The exponential distribution only gives a very rough estimate of the cumulative distribution. Over 50 % of the chunk sizes were less than 174 bytes. When with LAN the probability of the chunk size being under 2000 bytes was 63.7 %, with LTE the same probability was only 400 bytes. Detailed probabilities for the chunk sizes 0-2000 bytes can be noticed in Table 19.

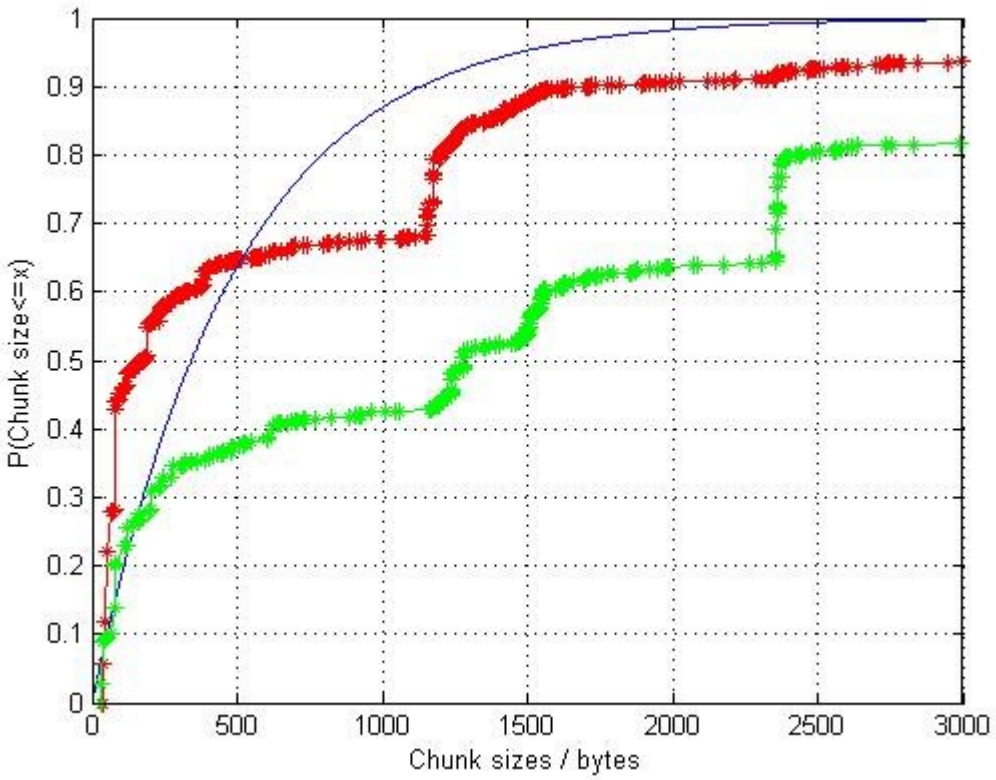

Figure 44: Empirical CDF of noise chunk sizes between 0 - 3000 bytes in LTE in red colour and exponential CDF with mean value 493 bytes in blue. LAN ECDF is in green colour.

Table 19: Probabilities for chunks sizes 0-2000 bytes in LTE

Chunk size (bytes)	0-100	100-200	200-300	300-400	400-500	500-600	600-700	700-800
Probability %	45.0	10.4	4.0	3.9	1.4	0.8	1.3	0.3
Chunk size (bytes)	800-900	900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600
Probability %	0.4	0.3	0.3	11.4	5.0	1.1	2.6	1.5
Chunk size (bytes)	1600-1700	1700-1800	1800-1900	1900-2000				
Probability %	0.3	0.3	0.2	0.3				

Table 20 presents probabilities for chunk sizes between 0 – 1400000 bytes. The probability of the chunk size being less than 10000 bytes is 98.2 %. Consequently, there can be some single chunks with a large amount of data but the probability for that is very small. Thus, generally speaking, noise stream chunks were very small compared to major stream chunks, e.g. the median of median values in the lower stream was 244498 bytes whereas in the noise stream the median was only 182 bytes.

Table 20: Probabilities for chunk sizes 0-200000

Chunk size (10000 bytes)	0-1	1-2	2-3	3-4	4-5	5-6	6-10	10-12
Probability %	98.2	0.5	0.3	0.2	0.4	0.1	0	0.1
Chunk size (10000 bytes)	12-14	14-15	15-20	20-140				
Probability %	0	0.1	0	0.1				

The empirical cumulative distribution of the time differences between the noise stream chunks is plotted in Figure 45 in red, a calculated exponential distribution in blue colour and the LAN results in green.

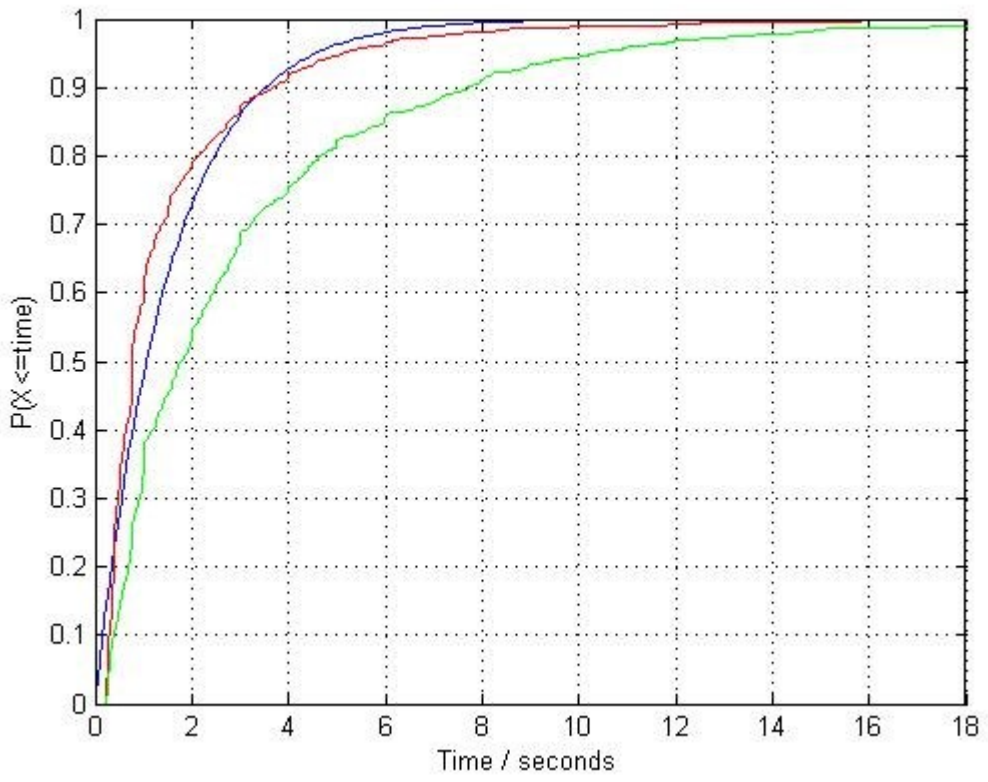

Figure 45: Empirical cumulative distribution of time differences in red colour and calculated exponential distribution in blue colour with the average value of 1.5246 seconds. LAN ECDF is in green colour.

The average value of data was 1.5246 seconds. We can see that the empirical and the exponential distributions are very close to each other, as was with LAN. RMSE was 0.075. The maximum difference was 28.5 seconds and the median value was 0.75 seconds. 94.6 % of the differences are under 5 seconds which is much more than 82 % with

LAN. This is only natural because there were many more chunks with LTE than with LAN, but the video clips were of the same length.

5.3 Differences between LAN and LTE

This chapter summarizes and highlights the most important differences which were noticed when the LAN and LTE results were compared in a TCP/IP level.

Generally, it can be seen that video transfer over LTE followed the same pattern as was noticed with LAN. This was, of course, the expected result, because it would be unlikely that YouTube servers or web page implementations checked packet transfer methods. The situation can be different with embedded web applications found e.g. in Android platforms. Future work could include the study of these applications.

In LTE network throughput capacity can be estimated best by using *Nemo SW* during the speedup phase. In DL the throughput was estimated to be 12.2 Mbps and in UL 0.45 Mbps. This is roughly half of the measured LAN capacity i.e. 20 Mbps in DL and 2 Mbps in UL.

The speedup phase length in LTE was 9.10 seconds whereas in LAN it was 9.97 seconds, but both in LAN and in LTE some 20 % of the packets were transferred during the speedup phase.

99 % of the data in LTE was transferred in the two major TCP/IP streams whereas the corresponding value in LAN was 97 %. The reason for this could be differences in background processes. In the two major TCP/IP streams the DL/UL TCP packets ratio was 1.73 in LTE and in LAN it was 1.59. In LTE the UL byte amount was 1.9 % of DL amount while in LAN it was 2.1 %. All in all, it seems that both in LAN and LTE, beside acknowledgements, something else is transmitted in UL, and in LAN the amount is little higher. The time differences between TCP/IP packets inside the chunks in LTE were lower than in LAN. One reason for this might be the fact that LTE radio network is buffering data while waiting for the radio resources and finally the packets are sent during the established radio resources as fast as possible. On the other hand, 3 % of the time differences were greater than 2 ms. This differs from LAN and it could be due to hybrid-ARQ retransmissions in LTE. Generally, the chunk sizes and delays of the major streams in LAN and LTE were close to each other. But there were more variations both in the high and low stream chunk sizes in LTE than in LAN. With low stream in LTE there were noticed more small chunks. This could result from some delays in the LTE network, which causes a chunk to split into several smaller chunks. This happens when there is a delay of over 200 ms, according to our definition; the packet belongs into the different chunk in that case. In some cases very large chunks were seen during the speedup phase in the high stream. This could happen because LTE radio link was much slower than LAN and chunks were combined. The same phenomenon also affects the lower and higher stream chunk time differences which have more variation than with LAN.

There were many more noise chunks in LTE than in LAN but the total data amount of the noise chunks was less than with LAN. This naturally means that single chunks themselves were smaller in size. The reason for this is not clear and the clarification is left for the future work. Because there were more chunks in LTE, it means that the time differences between them were smaller than with LAN, but it can still be seen that the time differences follow the exponential distribution. From the eNB scheduler point of view this means that there are no long pauses in transition, which could give possibility for long DRX or sleeping periods.

Since the differences between LTE and LAN were not large and seemed to be mainly because of eNB parameters and LTE radio network functionality, the LAN measurement results are used in the YouTube model in Chapter 6. LTE measurements were needed to verify that basic YouTube behaviour is the same regardless of the used network.

6 EMPIRICAL YOUTUBE TRAFFIC MODEL

This chapter presents a summary of how YouTube works and it introduces a simple YouTube model based on the findings with LAN dealt with in the previous chapters. This model can be used either for simulating YouTube traffic or as a background for understanding consequences of YouTube traffic for systems like e.g. DRX in LTE. The model should be more realistic than previous Poisson based models [14], [15] used e.g. in standardisation.

6.1 Summary of findings

This model is based on the measurements done in LAN, because LAN gives a good approximation of the network without major disturbances or delays. YouTube transmits audio and video over the Internet using the two main TCP/IP streams. Normally, 97 % of the whole data is transmitted in these two streams. In addition, there are several smaller TCP/IP streams during the session, where the rest 3 % of data is transmitted. This model applies for when a web browser based YouTube is used. When examining the data patterns more closely, it can be seen that data is sent periodically – not continuously. The actual transmission time in fast networks (around 10 Mbps) is very short compared to the actual video viewing time. The transmission time can be just around 3 % of the viewing time. Naturally, YouTube transmission is very DL biased.

Figure 46 gives a view of how YouTube data transfer looks like. There are highlighted different main parts in the video reception:

1. Speedup phase in red at the beginning of transfer

   On the average 20 % of the total data is transferred during the speedup phase in order to fill video codec buffers for non-interrupted viewing. On time scale, the speedup phase usually lasts around 10 seconds and also during the speedup phase data is delivered in chunks, but the chunks can be much larger than later in the other phases. The chunk sizes during the speedup phase followed the chunk sizes of the low stream or the high stream or both of them (239 kBytes, 1130-1170 kBytes or 1380 kBytes).

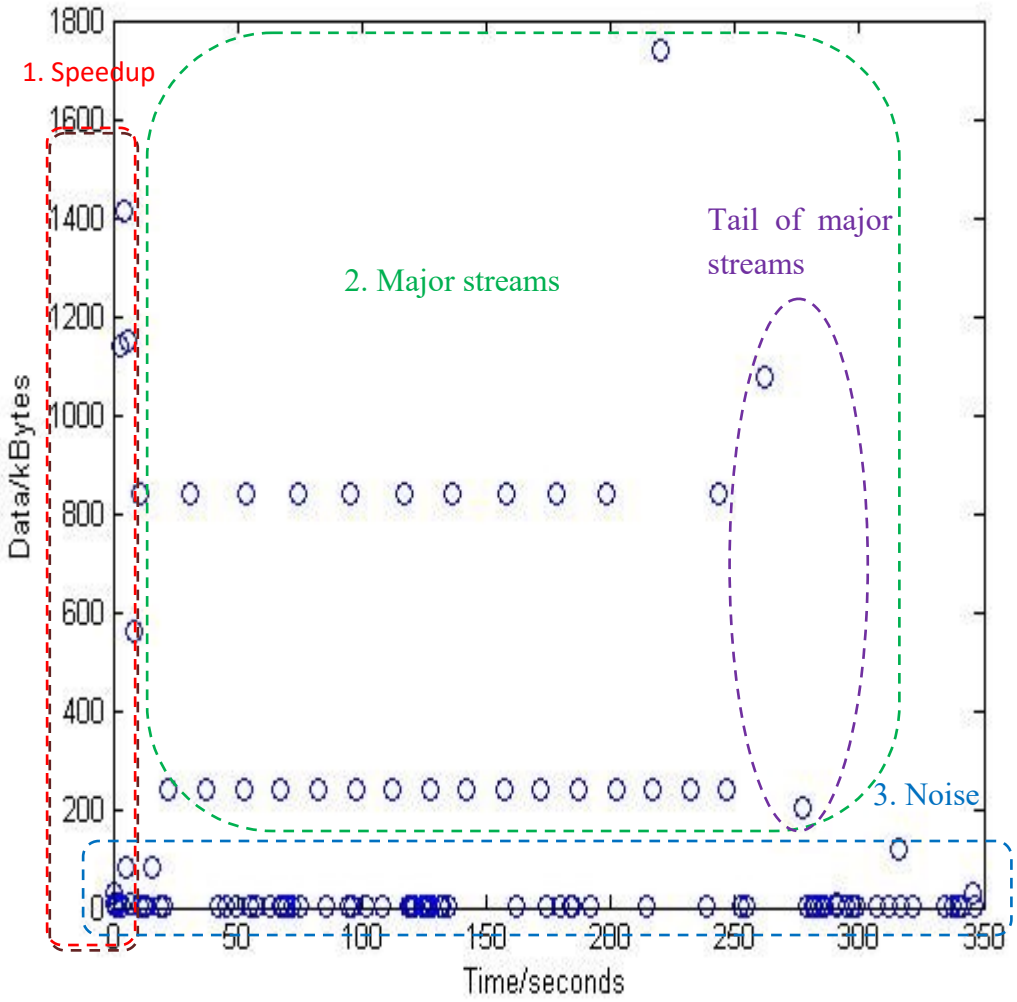

Figure 46: Typical view of YouTube data transfer over LAN (video clip 1). Speedup phase is in red colour, major streams in green colour, noise in blue colour and tail of the major stream in violet.

2. Two major TCP/IP streams (high and low) in green

The two major TCP/IP streams deliver over 97 % of the data during reception. These two TCP/IP streams can be further divided into high stream and low stream chunks. One must bear in mind that these TCP/IP streams do not necessarily match one to one for the low stream and high stream (e.g. see Figure 10 and Figure 11). This distribution of the high and low streams can be modelled as a superposition of two packet streams, both having their own statistical distribution for the packet sizes and time differences.

There can be seen one TCP UL acknowledgement for every 1.59 TCP DL packets. UL byte count is 2.1 % of the DL data count. Inside the chunks the time differences between the TCP/IP packets are very small, over 96 % of the packets are less than 0.9 ms of the next packet. Over 80 % of the timer differences between the chunks during the speedup phase were between 1 and 4.1 seconds.

One high stream chunk usually contains 4 % of the total received data and the maximum size of the chunk is less than 1200 kBytes. Time difference between the chunks is very often 15 seconds and the last full size regular high stream chunk appears when 74 % of the viewing time has elapsed. YouTube can expand this time difference in case the file size is less than the average. In this case 4 % and 74 % rule is applied, too. Only the time difference between the chunks is increased relatively.

During the steady phase YouTube tries to keep the high stream chunk sizes constant. Over 90 % of the chunks were between 1130 – 1170 kBytes. The same applies for the time differences between the high stream chunks in the steady phase. Depending on the video size, values around 15 and 20 seconds could be observed.

In general, one low stream chunk contains 244498 bytes (239 kBytes), so it seems that the size is quite fixed and it does not depend on the total video size. Over 70 % of the time differences between the low stream chunks were very close to 14.9 seconds. The studies indicated no such variation in time differences that was noticed in high stream chunks based on the video size. Additionally, the last low stream chunk appears when approximately 74 % of the video has been viewed. It is possible that this low stream contains audio data of the file.

The tail of the major streams is formed of a few chunks of the higher and the lower streams after 74 % of the file has been transferred. These chunks are irregular in size (again see e.g. Figure 10 and Figure 11).

3. Background noise, (the rest 3 % data), in blue

Here the chunks are very small compared to the major TCP streams, and 64 % of the chunks are under 2 kBytes. UL byte count is 20.4 % of the received DL bytes, so obviously something else but 40 bytes TCP/IP acknowledgements are sent in UL, too. For every UL packet 1.12 DL packets were received. The time differences between the chunks follow the exponential distribution, which can be used to give probabilities with the mean value of 3.08067 seconds. So during a transfer there are a lot of this kind of extra activity although data amounts are quite small compared to the two major TCP/IP streams. It is possible that some of these streams are also used for YouTube activities, e.g. to control data sending, advertisements and side bars of the view.

6.2 Simple YouTube model

A simple YouTube model was made by using *Matlab* scripting. This model generates YouTube-like traffic for a given video size and video length. The algorithm is simple but easily expandable if more accurate features are needed. It can be used to generate TCP/IP packets without headers or at simplest, only Data Link layer's Service Data

Units (SDU) without real TCP/IP headers or real TCP/IP functionality. Also, the algorithm can help to understand the summary presented earlier because the model is more detailed. The *Matlab* code is given in the Appendix. Here is a brief introduction to the main points in the model:

- First, a user selects the video DL size in bytes and the length in seconds. After this no more parameters are needed and he can run the code.
- In the code, the chunks for the major streams in the speedup phase are calculated first. Parameters for the chunk sizes and the time differences are from Table 12 and Table 13. The speedup phase takes 19.3 % of the major streams DL data.
- The basic high steam chunk size – 4 % of the given video size – is calculated next. Using this value is calculated the average time difference between the high stream chunks. As can be remembered, the regular sized high stream chunks end when 74 % of the video is viewed. An average time difference calculation is not needed for low stream chunks because it is not depend on the file size and the low stream ends at the same time as the high stream.
- Next both high stream and low stream chunks are generated for the steady phase. The high stream chunk probabilities are from Table 8 and Table 9. The low stream chunk probabilities are from Table 10 and Table 11.
- After that, the noise chunks are generated for the whole duration of the video clip. The noise chunk size probabilities are from Table 14 and Table 15. The time differences for the noise chunks are generated using the exponential cumulative distribution function with the mean value of 3.08067 seconds.
- Then, IP packets are generated from the major stream chunks. Fixed sizes of 49 bytes in UL and 1495 bytes in DL are used. DL/UL packet ration is kept in 1.59.
- Next, IP packets are generated from the noise chunks. Fixed sizes of 179 bytes in UL and 781 bytes in DL are used. DL/UL packet ration is kept in 1.1225.
- Finally, both the major stream and the noise stream chunks and the IP packets are combined and sorted in time order.

So, the model produces the chunks first, and later IP packets are based on them bearing in mind that this model presents the conditions in one setup using a very good LAN channel. The setup was chosen to provide a simple model of the transport layer behaviour. By applying this model in top of a system level simulator instead of an infinite constant rate buffer, this model can provide a valuable insight for mobile broadband system designers about the effect of network parameterization in terms of QoS and the energy consumption of mobile devices.

The model could be changed to a more theoretical one by changing the tables used for the high and low stream chunk sizes and time differences to contain less variation or even fixed values. Additionally, the IP packet generation could be replaced with a more accurate model of the TCP/IP. Alternatively, the table generating IP packet time differences could be changed to give more disturbances in the channel. This model uses simply fixed size IP packets and there can also be IP packets which are unrealistically low in

size. If a user wants to simulate the effects of interrupted viewing, he can run the model for the whole video and later simply cut the results in the wanted time point. Or if there is a need to study the effect of several simultaneous video transmissions, the model is first used to create each stream separately and then different transmissions streams can be combined using the timestamps. The noise stream is handled separately in this model, and it would be straightforward to change the table for noise chunk sizes or change the time difference distribution. This way different background noise situations can be simulated, e.g. there could be used background noise of LTE measured in this study.

The used tables for the time differences sometimes make the chunks be too close to each other, which causes the chunks to merge after an IP packet creation. This occurs especially when the earlier chunk is large in size and the time difference is very small. To prevent this from occurring, the model enlarges the time differences if needed. It would require further study to see if the time difference is actually dependent on the size of the previous chunk.

Moreover, in this model the tail of the major streams is assumed to contain only the bytes, which are not sent during the steady phase. It would require more study to see if this is also the case in reality.

7 YOUTUBE AND DRX

In this chapter YouTube traffic is studied with DRX functionality. First, theoretical values for possible DRX functionality are presented. Then, as a special case, YouTube with LTE DRX is examined more closely and the effect of different promotion timer values for energy consumption is simulated.

7.1 YouTube transmission and RF activity

DRX can be used only in cases when there are breaks in data transmission. In order to know how long a time TX and RX must be on, software was developed which calculates transmission and reception times based on the measured TCP/IP packet sizes and time stamps. The model assumes full-duplex operation and no delays in packet delivery. UL and DL throughput values can be given separately to the SW. Figure 47 shows an example how the total transmission time was calculated. DL and UL packet transmission times were calculated based on the measurements and given throughput values, and finally, they both were combined to get the total RF time.

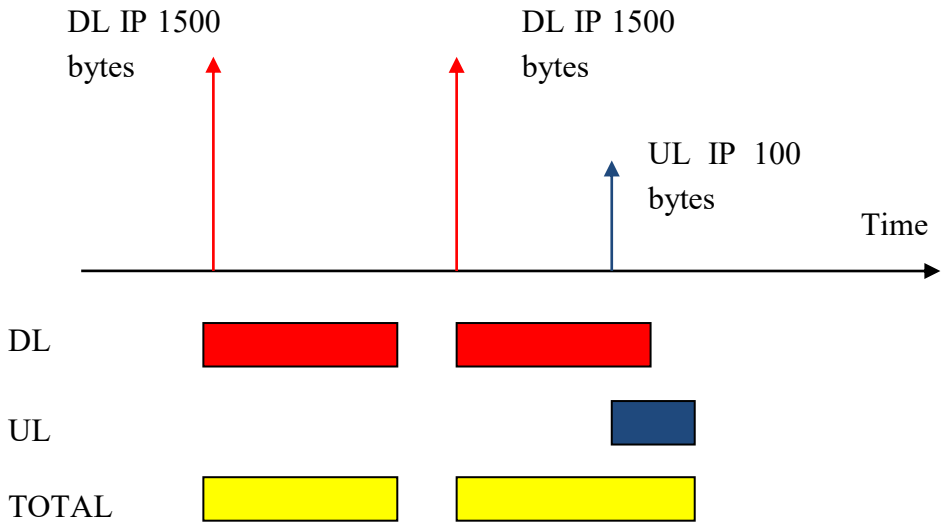

Figure 47: Example of transmission/reception time calculation

These values were calculated for all the measured LAN clips. For the clips 20 Mbps DL speed and 2 Mbps UL speed was used. The results are presented in Table 21, which shows calculated possible RF activity times. It can be noticed that theoretical RF time was very short compared to the total transmission time and non-activity time was clearly the most dominant figure. Median RF time lasted only 3.6 % of the total viewing

time and median of UL activity time versus DL activity time was 25.9 %. So with an optimum DRX implementation the device could sleep over 96 % of the video viewing time in FDD. Because UL transmission speed was quite low, TX time was high compared to transmitted byte amounts. It must also be taken into account, that since UL and DL activities overlap in FDD, the total RF activity time is not directly the sum of UL and DL activity. Total RF activity time average was 84 % of the sum of UL and DL activity.

Table 21: RF activity times for LAN FDD, times are in seconds

Clip	DL activity time	UL activity time	RF activity time	RF non-activity time	Non-activity %
1	8.49	2.10	8.98	337.63	97,4
2	16.19	3.98	16.88	458.15	96,4
3	11.16	2.85	11.80	287.83	96,1
4	10.81	2.67	11.29	286.88	96,2
5	10.40	2.72	11.11	267.27	96,0
6	11.30	3.08	12.01	318.77	96,4
7	12.47	3.34	13.19	384.28	96,7
8	6.98	1.82	7.43	349.86	97,9
9	9.09	2.43	9.57	352.59	97,4
10	11.73	3.02	12.30	329.25	96,4

The sum of DL and UL activity presents RF activity in Time Division Duplexing (TDD) system, where transmission and reception cannot overlap. The average RF non-activity time in FDD was 96.6 % whereas in TDD it was 96.1 %.

Next the same calculations were done for the LTE measurements. Here DL throughput 12.2 Mbps and UL throughput 0.45 Mbps were used. These values were earlier derived in Chapter 5.2.1. The results for LTE are presented in Table 22.

Table 22: Activity times for LTE FDD, DL 12 Mbps and UL 0.45 Mbps, times are seconds

Clip	DL activity time	UL activity time	RF activity time	RF non-activity time	Non-activity %
1	14.27	8.28	15.47	336.57	95,6
2	27.54	18.54	29.26	443.92	93,8
3	18.18	11.27	19.07	279.76	93,6
4	17.35	10.16	18.13	276.59	93,8
5	16.89	10.05	18.15	260.10	93,5
6	18.49	12.40	19.64	312.31	94,1
7	20.44	13.39	21.66	377.29	94,6
8	11.67	8.58	13.94	344.45	96,1
9	13.59	7.64	14.84	308.04	95,4
10	14.18	10.64	15.37	287.80	94,9

RF non-activity average was 94.5 % in FDD and 91.7 % in TDD. The difference was clearly larger than in LAN, because very slow UL caused transmission to spread in time. When compared to the LAN results it can be seen that both DL and UL activity times were much longer than with LAN. With LTE the median RF time lasted 5.7 % of the total file time and the median of UL activity versus DL activity was 64 %. Both of the figures are much higher than with LAN, because the radio link was slower than in the fixed LAN. Although UL was now very slow and UL transmitting took clearly longer than with LAN, the total RF activity time in FDD did not increase in the same proportion. Total RF activity time average was only 65 % of the sum of UL and DL activity, which was less than LAN figure of 84 %. This can be explained by the fact that DL was now slower, too. Because both UL and DL were slow, there were more opportunities for TX and RX to overlap, which is not a problem in FDD. Besides, YouTube video viewing in UL consisted mainly of TCP acknowledgements which appeared inside the chunks at the same time as downlink receptions were taking place. TCP acknowledgements are very short compared to DL packets and the device has time to transmit them during DL activity in FDD. In TDD the situation is quite different and slow UL caused long RF activity times because UL and DL cannot overlap.

The same calculations were done in LTE for UL throughput 50 Mbps and DL throughput 100 Mbps which are the maximum throughputs for category 3 in LTE. These results are to be seen in Table 23.

Table 23: Activity times for LTE FDD, DL 100 Mbps and UL 50 Mbps, times are seconds

Clip	DL activity time	UL activity time	RF activity time	RF non-activity time	Non-activity %
1	1,74	0,07	1,76	350,28	99,5
2	3,36	0,17	3,40	469,78	99,3
3	2,21	0,10	2,24	296,59	99,3
4	2,11	0,09	2,14	292,58	99,3
5	2,06	0,09	2,08	276,17	99,3
6	2,26	0,11	2,29	329,67	99,3
7	2,49	0,12	2,52	396,42	99,4
8	1,42	0,08	1,45	356,93	99,6
9	1,66	0,07	1,67	321,19	99,5
10	1,73	0,10	1,76	301,41	99,4

E.g. DL activity time for clip 2 decreased to 3.36 seconds in FDD and on the average, the non-activity percent share increased from 94.5 % to 99.4 %. It is logical that the RF activity time decreased in the same proportion as the throughput increased. In the same way for clip 2, the UL activity time decreased to only 0.17 seconds, so the time used for UL is very small with high throughputs. The average TDD RF non-activity increased up to 99.4 %, which is exactly the same as FDD average. This leads to the conclusion that when the throughput increases, the proportion of non-activity time of a TDD system closes the non-activity proportion of a FDD system.

Because YouTube transmission happens mostly in two major TCP/IP streams, it is good to see what happens if only those two streams are transmitted. For LAN traffic, RF activity for two major TCP/IP was calculated and presented in Table 24. The last column in this table shows how much longer RF can sleep more when compared to a full file with noise, presented in Table 21.

Table 24: RF activity times for LAN for only two major TCP/IP streams in FDD, times are seconds

Clip	DL activity time	UL activity time	RF activity time	RF non-activity time	RF Non-activity difference to full file
1	7.77	1.55	7.92	338.69	1.06
2	15.96	3.36	16.22	458.81	0.66
3	10.87	2.22	11.04	288.59	0.76
4	10.45	2.17	10.63	287.54	0.66
5	10.17	2.02	10.34	268.04	0.77
6	11.03	2.39	11.21	319.57	0.80
7	12.31	2.70	12.53	384.94	0.66
8	6.78	1.35	6.96	350.33	0.47
9	8.88	1.74	9.03	353.13	0.54
10	11.53	2.49	11.72	329.83	0.58

It can be seen on the last column that removing everything else but the two major TCP/IP streams causes only minor increase in RF non-activity. The increase is on the average only 0.70 seconds, i.e. 0.02 % of the viewing time for a single clip. This was an expected result because in the measurements the two major streams carried 97 % of the data. In TDD system the increase is a little greater, on the average 0.89 seconds per clip, i.e. 0.03 %.

Next the calculations dealt with the distribution of RF non-activity lengths. To get these lengths the same method was used as in the previous chapter and in Figure 47. The measurements from LAN were used and the case, which included all the existing data including major streams and noise streams. The data from all the 10 clips were combined and a cumulative distribution of the lengths is shown in Figure 48 and with a different time scale in Figure 49. The empirical results of all the streams are plotted in red colour. Median value was 0.0086 seconds, average 0.26 seconds and maximum value was 25.15 seconds. In the empirical distribution 91 % of the pauses in the transmission were below 200 ms, but there also existed greater values. Still, 89.8 % of the lengths remained under 100 ms and 86 % of the lengths under 50 ms. Most of the small pauses take place because RF is able to transmit a TCP packet in a chunk before the next TCP packet arrives. This means that there are not very many opportunities for long sleeping periods in RF circuitry, and very small but frequent pauses in RF activity dominate the distribution. To study the effect of frequent noise stream packets to pause lengths, it was decided to delay all the noise IP packets so that they appeared only when there was also activity with the two major TCP streams. The total time of RF activity caused by the noise TCP streams was only 6.96 seconds. The noise TCP streams were filtered out and there were seen 5461 pauses in the remaining two major TCP streams.

The assumption was that every RF non-activity period in the two major TCP streams was decreased by the amount of RF activity in the noise streams using the formula:

$$time\ decrease = \frac{Noise\ RF\ activity\ time}{Number\ of\ pauses\ with\ two\ major\ TCP\ streams} = \frac{6.96\ \text{seconds}}{5461} = 0.0013\ \text{seconds} = 1.3\ \text{ms} \quad (9)$$

This gives a good approximation of the effect of delaying the noise streams.

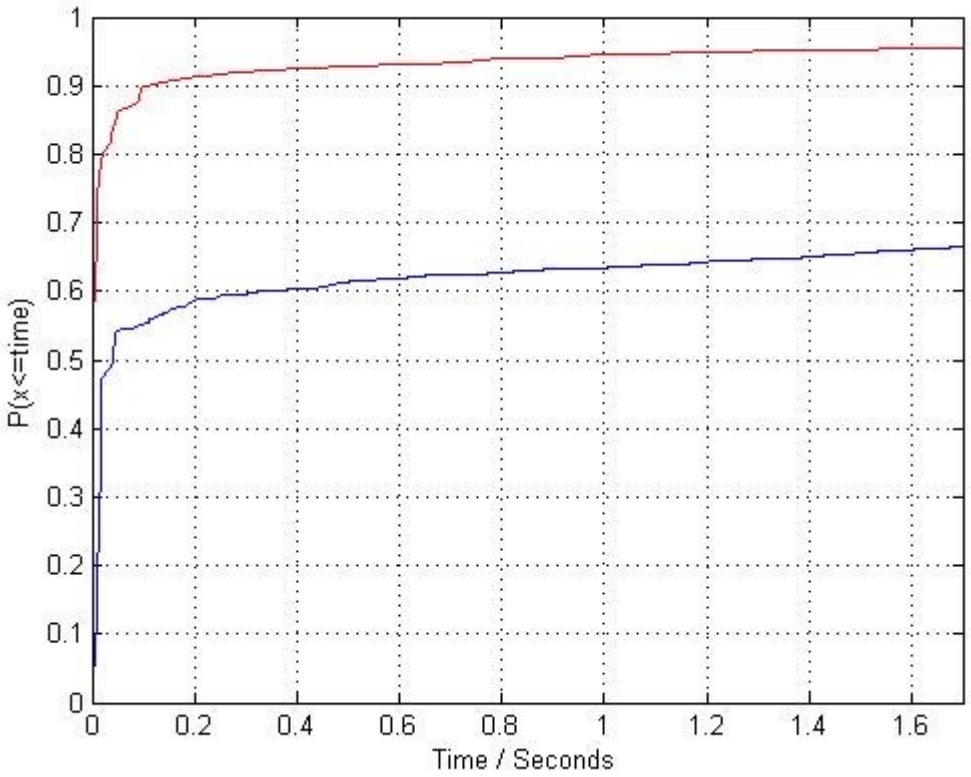

Figure 48: Cumulative distributions of RF non-activity lengths with all the streams in red colour and the distribution when noise TCP/IP packets are delayed in blue colour.

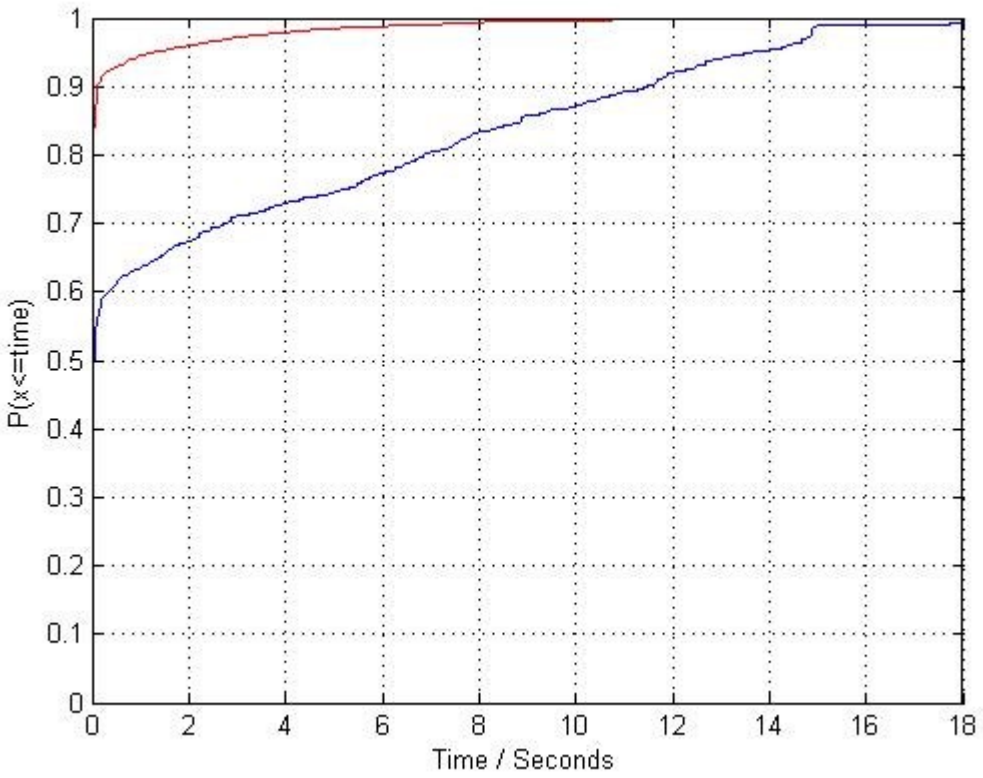

Figure 49: Distributions of RF non-activity lengths in transmission with all the streams in red colour and the distribution when noise TCP/IP packets are delayed in blue colour.

Again, the distribution of RF non-activity is plotted and can be seen in Figure 48 and Figure 49 in blue colour. Average was 3.01 seconds, median 0.041 seconds and maximum was 30.03 seconds. 60 % of the RF non-activity lengths were under 0.32 seconds and below 11.48 seconds stayed 90 % of the values. Over 200 ms were 41 % of the values. This shows that delaying the noise packets causes the probability for larger pauses in RF activity to increase heavily. 9 % of the pauses were over 200 ms when not delaying noise packets. Here can be noticed clearly again that 98.9 % of the RF pauses were under 15 seconds in both cases. This is not a surprise, because 15 seconds periods were noticed with the two major TCP streams. So it becomes quite evident that in a normal situation noise packets cause short RF non-activity lengths between the transmissions. This means few opportunities for RC circuitry sleeping and benefits of delaying those noise streams are evident. There are still many pauses below 100 ms, because RF is able to transmit a TCP packet in a chunk before a next TCP packet arrives. If also major TCP streams were delayed little and buffered, it should be possible to get rid of also most of the small pauses, which are below 100 ms. Generally, this kind of delaying is a form of traffic shaping or coalescing technique and similar kind of systems are explained in [28],[29] and [32].

As a side effect, there will be delays in the noise packets transmission. The longest RF non-activity length was 30.03 seconds that is also the longest delay which can occur for the delayed noise chunk. As can be remembered from Chapter 4.3.3, the noise chunk appearance followed the exponential distribution with the average value of 3.08 sec-

onds. To see the distribution of the delays, *Matlab* simulation was done where the noise chunks were generated using this exponential distribution, and the delays of the noise chunks were then calculated using real data from the two major TCP streams. A cumulative distribution of the delays is visible in Figure 50.

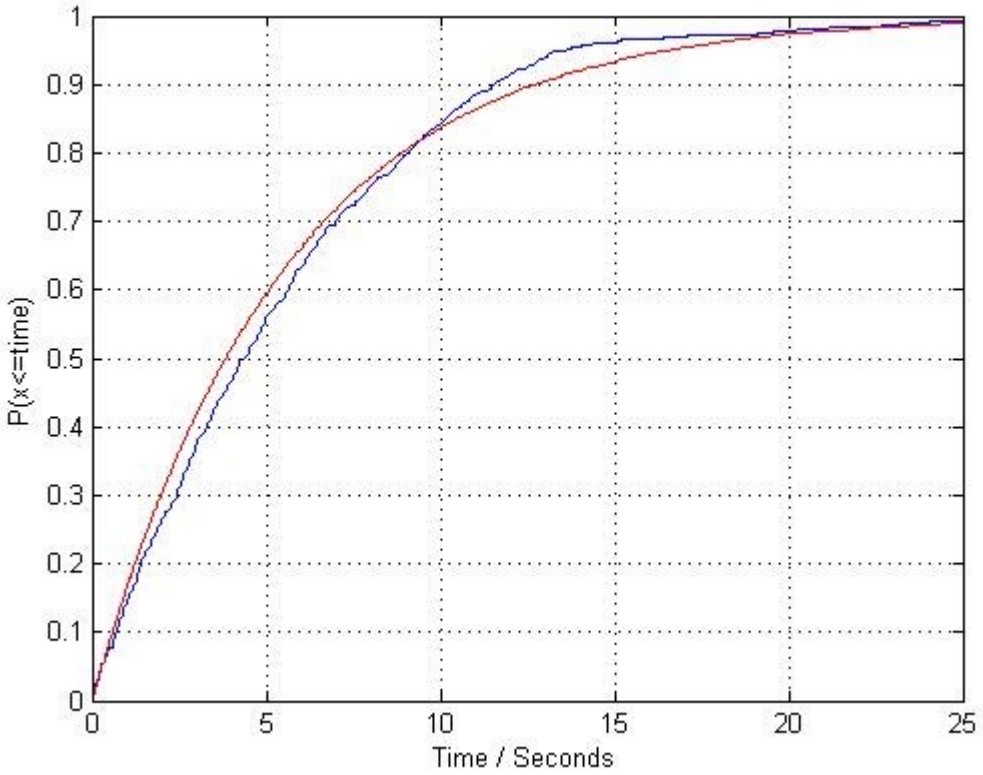

Figure 50: CDF of delay lengths of noise chunks. Empirical distribution is in blue colour and exponential distribution with the average value of 5.7 seconds in red colour.

An exponential distribution was calculated with the average value of 5.7 seconds and it is in the picture in red colour. RMSE average was 0.038, which tells us that the exponential distribution characterizes the delay quite well. In this distribution over 58 % of the noise delays are less than 5 seconds and over 90 % less than 13.2 seconds. We can expect TCP retransmission timer to be a few seconds at the beginning, so even with the delay of 5 seconds, there will most probably be retransmissions of noise TCP packets. The TCP retransmission timer value is doubled with every retransmission, so the connection failure should not happen with delay tolerant applications. A longer delay also means that the packets must be buffered both in the network and in the mobile. The retransmissions cause increase in buffer requirements, but luckily the noise packets are small in size. The buffer requirements could possibly be alleviated with an intelligent duplicate detection, which could remove the retransmitted TCP packets. Delaying the packets will not work with delay sensitive applications like voice. So some intelligent detection would be needed in the system to find such traffic.

7.2 LTE DRX and promotion timer

This chapter presents measured YouTube traffic patterns combined with LTE DRX and promotion timer in the network. The first thing was to create a *Matlab* model to calculate energy, because, as far as the MS is concerned, saved energy describes DRX usefulness. For simplicity reasons it was assumed that the short DRX is not used in the system and only long DRX is in use. For energy calculations values in Table 25 were used.

Table 25: Values used in calculations, from [18]

State	Power (mW)	Timer Duration (ms)	Timer	Explanation
LTE promotion	1210.7	Varied	Promotion timer controlled by network	When MS moves from RRC_IDLE to RRC_CONNECTED it takes 260.1 ms. Movement from RRC_CONNECTED to RRC_IDLE happens instantly in model
BIDI data transmission RRC_CONNECTED	3204			Simultaneous data transmission/reception
DL data reception RRC_CONNECTED	2327			Data reception only 20 Mbps (DL)
UL data transmission RRC_CONNECTED	2165			Data transmission only 2 Mbps (UL)
LTE tail base RRC__CONNECTED	1060			No data transmission but UE is ready and listening to channel, DRX is possible.
LTE DRX On RRC_IDLE	594.3	43.2	onDurationTimer	UE listens to paging during DRX in RRC_IDLE
RRC_CONNECTED		40	LongDRX-Cycle	Cycle period for DRX
RRC_IDLE		1280	DRX-Cycle	Cycle period for DRX (How often MS listens to paging)
RRC_CONNECTED		100	drx-inactivity Timer	How long UE waits until DRX is started

These values as well as timer values came from previous studies [18]. Values for data transmission during the RRC_CONNECTED state were calculated using the equation (3) for instant power presented in Chapter 3.3. Additionally, using values from [18] meant that in the RRC_CONNECTED state, it was not possible to differentiate when the device listened to paging messages during DRX cycle. Consequently, the same power level defined in the LTE tail base state was used both when DRX-inactivity Timer and longDRX-Cycle timer were running. This means that this study concentrates on the effect of the promotion timer which is controlled by a network. When this timer expires a network will order an MS to move from the RRC_CONNECTED state to the RRC_IDLE state. When data transfer starts again, it requires extra energy to move the MS back to the RRC_CONNECTED state. This is called LTE promotion energy. The model also assumed that this transfer from the RRC_IDLE state back to the RRC_CONNECTED state takes 260.1 ms but it could happen only when DRX-Cycle had expired. The model did not take into account possible MS started UL activities during DRX periods. If some DRX period were to overlap with the next packet transmission or reception, this overlapping period was reduced from the next RF inactivity time. In this model different power levels were used during paging and idling while the device was in the RRC_IDLE state.

The minimum transmission energy for data was calculated first. Like previously, the data was from the LAN measurements and included all of the 10 video clips. The minimum transmission energy means necessary activity of radio as presented in Table 21. When RF is not active, energy consumption is assumed to be 0 J. RF was active 28.01 seconds in UL, 108.62 seconds in DL and total RF time was 114.56 seconds. This makes as *RF active energy*:

$$\begin{aligned} &RF\ active\ energy \\ &= UL\ only\ time \times 2165\ mW + DL\ only\ time \\ &\times 2327\ mW + BIDI\ time \times 3204\ mW \\ &= (114.56\ \text{s} - 108.62\ \text{s}) \times 2165\ \text{mW} \\ &+ (114.56\ \text{s} - 28.01\ \text{s}) \times 2327\ \text{mW} \\ &+ (114.56\ \text{s} - 5.94\ \text{s} - 86.55\ \text{s}) \times 3204\ \text{mW} \\ &= 5.94\ \text{s} \times 2165\ \text{mW} + 86.55\ \text{s} \times 2327\ \text{mW} \\ &+ 22.07\ \text{s} \times 3204\ \text{mW} = 284.97\ \text{J} \end{aligned} \tag{10}$$

RF active energy stayed constant in the model, because there was always the same amount of data to be transmitted and received. *Total energy* can be calculated using the equation:

$$Total\ energy = RF\ active\ energy + RF\ nonactive\ energy \tag{11}$$

Next calculations dealt with the situation in which all of the RF non-activity time appeared with LTE tail base power in the RRC_CONNECTED state using the equation (11):

$$\begin{aligned} Base\ energy &= RF\ active\ energy + sleeping\ time \\ &\times LTE\ tail\ base\ power \\ &= 284.97\,\mathrm{J} + 3372.51\,\mathrm{s} \times 1060.0\,\mathrm{mW} = 3860\,\mathrm{J} \end{aligned} \quad (12)$$

This *Base energy* is used as a reference value because it presents the situation where an MS always stays in the RRC_CONNECTED state and the long DRX cycle is 40 ms. This very same *Base energy* could be calculated using the model with very long promotion timer value and adding the result with *RF active energy*. The timer must be long enough to prevent promotions to the RRC_IDLE from occurring. So the used *Matlab* model actually calculated *RF non-active energy* using the given promotion timer value.

Next the promotion timer was set to a value of 11.576 seconds and the *Matlab* model was used for calculations. The promotion timer value is used here as an example, because it is directly from [18] where it was calculated using reverse engineering in a real network. With this value, the state transition from the RRC_CONNECTED state to the RRC_IDLE state occurred 29 times and using the equation (11), total energy was:

$$\begin{aligned} Total\ energy &= RF\ active\ energy + RF\ nonactive\ energy \\ &= 284.97\,J + 3461.30\,J = 3746.27\,J \end{aligned} \quad (13)$$

Finally, *RF non-active energies* and *Total energies* were calculated for all the promotion timer values from 1 to 30 seconds. The results were compared to *Base Energy* and they are plotted in blue in Figure 51. The smaller the promotion timer value was, the larger the energy savings were. With the 15 seconds timer the gain was only 0.7 %, and with the 26 seconds timer there was not a single promotion and the gain was thus 0 %. Clearly, using the state transitions from the RRC_CONNECTED to the RRC_IDLE saves energy. After the energy model and promotion timer study in this thesis was done but not yet published, Foddis et al. [36] published their study about promotion timer values. In their study data was mainly something other than video but their findings were similar to the ones in this thesis: small promotion timer values can save energy and with too large values the MS never goes to RRC_IDLE state.

Next was studied the situation where all the noise IP packets were delayed so much that they appeared only when there was activity with the two major TCP streams. Again, *Base energy* for the reference was calculated using equation (12) after applying equation (9) to the two major TCP streams. This makes *Base energy* 3475.77 J. This value is less than previously because the data with only two major TCP streams comes to an end earlier than the clips with all the data. To compensate this, approximately 364 seconds of noise IP streams was added at the end of the calculations. After this addition the *Base energy* was the same 3860 J as before. Again, Matlab model was used to calculate energies with the different promotion timer values. The results were compared to the calculated *Base energy* and are presented in Figure 51 in red. Still with 29-second promotion timer a minor energy saving of 0.02 % was received. To compare how much more energy was saved with the delayed noise packets, there is a line in magenta in

Figure 51 that shows the percentage of energy savings with delayed noise packets (red line) when compared to the situation without delaying (blue line). E.g. delaying noise packets gave over 10 % savings with the promotion timer value of 10 seconds. The savings melted down when the promotion timer value increased. Around the measured promotion timer value of 11.576 seconds in real network [18], the energy savings with delaying the noise packets were 7.6 %.

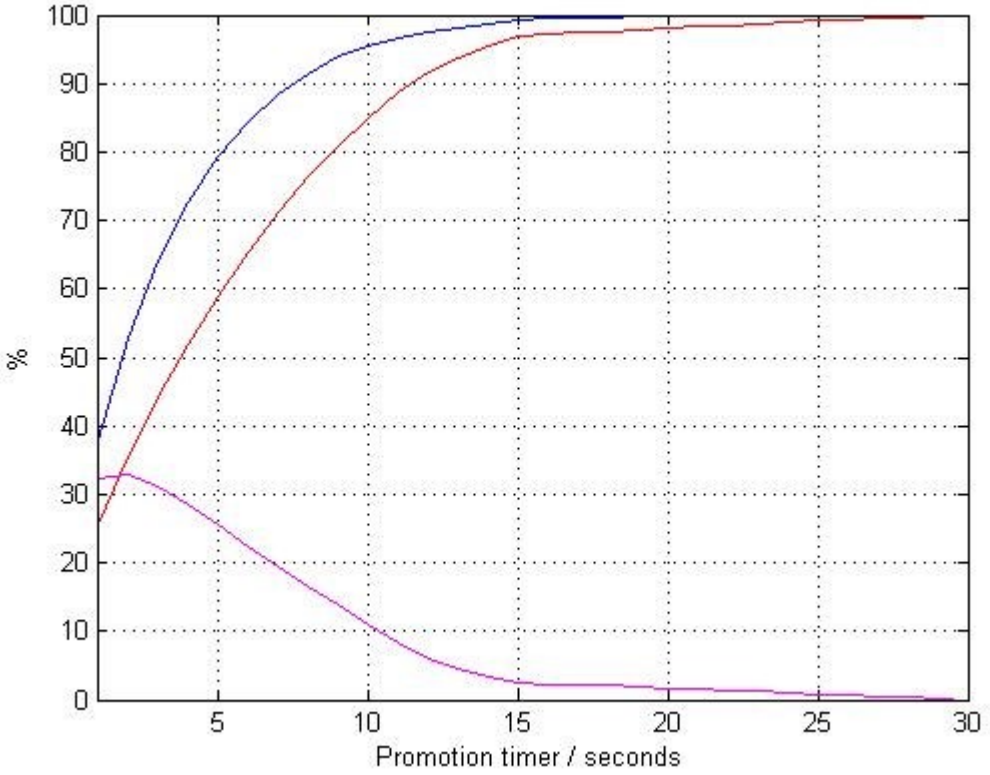

Figure 51: Energy usage comparison to *Base Energy* (always in RRC_CONNECTED) with different promotion timer values (MS moves to RRC_IDLE). Energies without delaying noise packets in blue, with delaying noise packets in red, and the energy saving percentage if noise packets are delayed in magenta.

It is difficult to compare these results to the results of other studies [32], [33], [35], where different kind of traffic shaping had been used. This was because non-video data had been used, shaping had been more aggressive, network parameters had been dynamic and power values had been different. Those studies show maximum energy savings between 35 – 70 %. It is also difficult to say what would be the normal energy level to be used as a reference. For example, if it is assumed that the normal level in this thesis was the promotion timer value of 11.576 seconds, then it could be said that delaying noise packets with smaller promotion timer values can give over 50 % energy savings.

This Figure 51 shows that in this model it is nearly always beneficial to drop into the RRC_IDLE state than to stay in the RRC_CONNECTED state. Even without delaying the noise packets this is clear. The reason for this is the used power values in Table 25. When an MS is in the RRC_CONNECTED state and not transmitting anything it is in the *LTE tail base* state according to the table and uses 1060 mW. In the RRC_IDLE state an MS uses only 594.3 mW when checking paging messages and 1210.7 mW

when moving back to the RRC_CONNECTED state. Paging messages are checked every 1.28 seconds and this model expected that there is always at least one paging reception in the RRC_IDLE state. So it can be calculated the time t_{con}, when it is still better to stay in the RRC_CONNECTED state:

$$Energy\ in\ LTE\ tail\ base < Energy\ in\ idle \Rightarrow$$

$$Energy\ in\ LTE\ tail\ base < Energy\ in\ promotion + Energy\ in\ paging \Rightarrow \quad (14)$$

$$1060\ mW \times t_{con} < 1210.7\ mW \times 260.1\ ms + 594.3\ mW \times 43.2\ ms$$
$$\Rightarrow$$
$$t_{con} < 321.3\ ms$$

In conclusion, it is better to stay in the RRC_CONNECTED state only when the period without any transmission or reception lasts less than 321.3 ms. The model assumed that the transition from the RRC_CONNECTED state to the RRC_IDLE state occurred with zero energy usage. Even if some energy is used for that transition and e.g. *Energy in idle* in equation (14) is doubled, t_{con} must be less than only 642.6 ms. LTE uses more energy in the *LTE tail base* than during the actual transmission or reception in a long term transmission according to [18].

On the other hand, moving from the RRC_IDLE state to the RRC_CONNECTED state causes delay before data transfer can continue. Huang et.al. [18] measured this delay to be 260.1 ms. Figure 52 presents the number of promotions with different promotion timer values for both the system without delaying noise packets and with delaying noise packets.

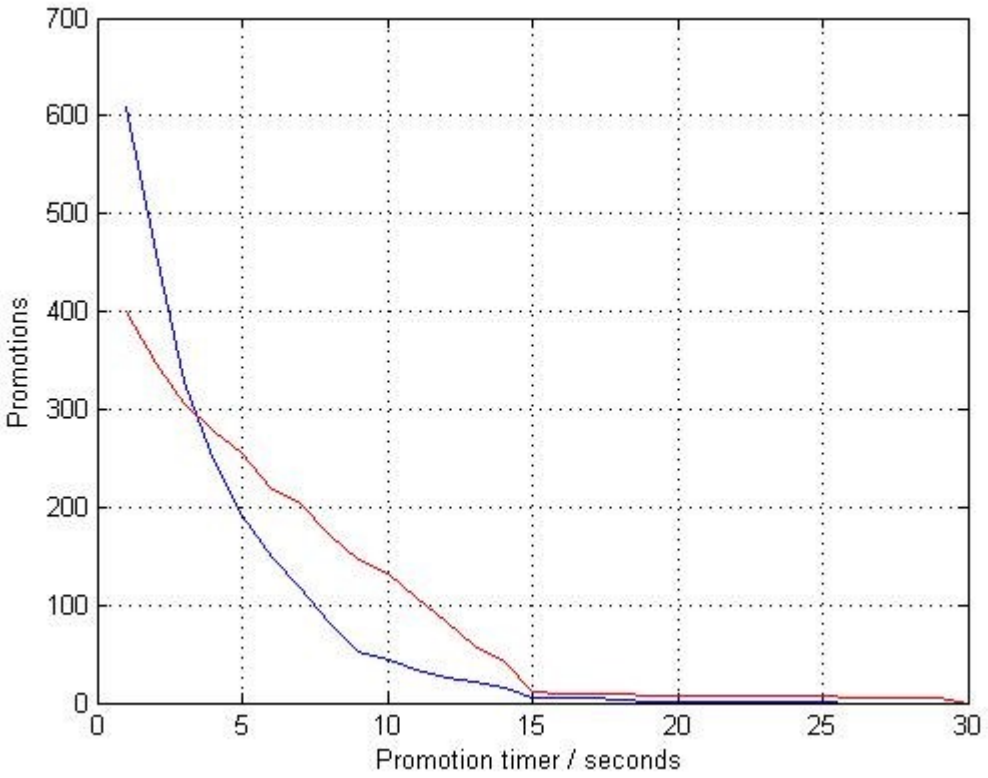

Figure 52: Number of promotions. In blue promotions without delaying noise packets and in red promotions with noise packet delaying

With low timer values there were hundreds of promotions. 609 promotions mean that the sum of the delay is over 158 seconds but here must be remembered that the total length of the clips was over 3487 seconds. Therefore, in the worst case scenario, there could be promotion timer expiration every 5.7 seconds. With low promotion timer values the promotion occurs often but the duration in the RRC_IDLE is usually very short because a new packet transmission or reception is on its way. We must bear in mind that after an MS has moved into the RRC_IDLE state, it moves back to the RRC_CONNECTED state only when there is data to deliver. Accordingly, in the delayed noise packets system there were less promotions but they lasted longer or the same amount of time with low promotion timer values. And without the delayed noise packets there were very many promotions but soon the transmission requirements forced an MS back to the RRC_CONNECTED state with low promotion timer values. This is the reason why there were more promotions without the delayed noise packets than with the delayed noise packets with small promotion timer values. This, of course, does not mean that the system without the delayed noise packets was more energy efficient. There were just more often transitions between the RRC_IDLE state and the RRC_CONNECTED state. On the other hand, the number of promotions decreased heavily as the promotion timer value increased. This is because there were not long enough pauses in packet transmission where promotion time expiration could occur in the system without the delayed noise packets. For example, with the 10-second promo-

tion timer there were only 45 promotions in the system without the delayed noise packets and 132 promotions with the delayed noise packets.

Moreover, moving back and forth from the RRC_IDLE state to the RRC_CONNECTED state increases signalling load between an MS and a LTE network. This added signalling may cause congestion in the control channels and reduce the overall performance of the network, because control channel resources are very limited [51]. Signalling includes first a command from the network to the MS to move to the RRC_IDLE state. When the data transmission occurs again, signalling includes possible paging from the network and the MS to use random access procedure. Using the three-way-handshake procedure the network orders the MS to the signalling radio channel. After that user data delivery becomes possible. [21]

7.3 Summary of DRX

The results of the studies showed that the actual YouTube transmission only takes a minor portion of the total RF time and, during 96 % of the video viewing time, RF could be switched off in FDD. Noise packets caused an increase of only 0.02 % for the RF time in FDD. As was seen in Figure 32, the appearance of noise packets followed an exponential distribution with the average value of 3.08 seconds, so this means that a device has to wake up for RF activity very often. The distribution of RF non-activity lengths were studied and in normal case 9 % of the lengths were over 200 ms, but when the noise packets were delayed, 41 % of the lengths were over 200 ms. Thus, delaying the noise packets is beneficial for RF sleeping opportunities. The delays of the noise packets followed the exponential distribution of the average value of 5.7 seconds, which means that over 58 % of the delays were less than 5 seconds. Delaying the packets means that there must be buffers both in the MS and in the network, and with these delay values TCP retransmissions are expected to take place, which increases requirement for buffering. On the other hand, the noise packets were small in size, which eases the buffering requirement. The delaying should not harm this kind of video transmission, but should there be delay sensitive transmissions at the same time, those would certainly suffer. This could be alleviated by an intelligent buffering control, which could use the delaying only in suitable cases. The difficulty in such control is detecting the cases from different TCP streams. Even with noise packets delaying there were still many RF non-activity lengths below 100 ms because RF was able to transmit a TCP packet in a major stream chunk before the arrival of the next TCP packet. The major TCP streams being delayed a little and buffered, it should also be possible to get rid of most of the small RF non-activity periods that were below 100 ms. This would increase the spectral efficiency of the DL transmission even more since the network would transmit larger continuous chunks instead of multiple separate packets.

When LTE was studied, it was noticed that with slow throughput values TDD kept RF on more than FDD, but when throughput was increased, the difference became smaller. In general, RF activity time decreased in the same proportion as throughput

increased. This means that with higher throughputs there appear longer RF non-activity periods, which give better opportunities for RF sleeping. But higher throughput means higher power consumption during reception and transmission according to equation (3). But then, consumed energy is power multiplied by time, so some energy is saved by faster transition to DRX and savings can be increased using noise packet delaying, which gives long idle periods.

A promotion timer in an LTE network decides when an MS can be moved from the RRC_CONNECTED state to the RRC_IDLE state. This timer is network dependent and it has not been standardized. Using the promotion timer, the value of 11.576 seconds gave approximately 3 % energy savings and 5-second timer gave over 20 % energy savings if the noise packets were not delayed. These values are in line with the studies [18], [36] where the importance of promotion timer for energy consumption was noticed with normal non-video data transmissions. While the noise packets were delayed, the corresponding values were 7.6 % and over 40 %. The smaller the promotion timer value, the larger energy savings were observed. Delaying the noise packets is therefore good for energy consumption because a network has more opportunities for moving an MS to the RRC_IDLE state as there are larger idle periods between the transmissions. As a side effect, every promotion causes an extra delay in transmission as an MS must be moved back from the RRC_IDLE state to the RRC_CONNECTED state. Also, this moving causes extra signalling in the signalling channels. Since the promotion timer is not standardized, a network vendor could use a different timer value for each MS, depending on e.g. applications used or network traffic conditions.

In this study, there was used constant LTE tail base power so the DRX functionality during the RRC_CONNECTED mode could not be separated from the paging reception power. Accordingly, the energy study part concentrated on the effect of the promotion timer. Additionally, used power values were quite high and for the future studies more accurate values are needed. New power values could be measured in laboratory environment using radio testers where timers and conditions can be controlled accurately. Regardless of the actual power levels, if the power ratios are similar the fraction of energy saved when using noise packet delaying or different promotion timer values should remain the same.

In the future networks where data speeds can be around gigabits per second, one could expect RF circuitry to be very power hungry. In that case it is very important to design DRX and sleeping periods in a way that RF can be switched off as often as possible. With such data speeds there can be more pauses even inside the YouTube chunks if the video server cannot send the data fast enough and fill the radio channel. Buffering and delaying latency tolerant data like a YouTube video would mean that data is transmitted as efficiently as possible in uniform chunks without RF pauses inside the chunks. After a chunk transmission RF can be switched off for several seconds to save power and bandwidth. There in may lie a problem how to detect such traffic from many choices. A video server might use some standardised method to inform networks about the nature of the traffic. This thesis concentrated on YouTube, but e.g. NetFlix may use a

totally different profile in video transmission. There could be standardised video transmission profiles, which could be optimised for wireless transmission. All different video providers could use those profiles in their implementations. This should benefit video providers, standardisation, users, network vendors and operators, since all would be aware of what the most efficient way of transmitting video over the wireless network was. Additionally, LTE uses quite a lot power also in the RRC_CONNECTED during DRX [18]. The future network standardisation should pay attention to ways of improving this.

8 CONCLUSIONS

Users watch and transmit more and more videos over the Internet. Optimizing either video transmission methods or doing network adaptation for video streaming can save scarce radio resources. When planning and standardizing future networks it is important to know how video streaming behaves. YouTube owned by Google is one of the dominant video sources nowadays and this thesis concentrates on studying YouTube transmission behaviour on the Internet.

After the first measurements of YouTube over LAN, it soon became evident that the earlier studies in [7], [9], [11], [32] about YouTube data profile did not match with the newly received measurement results. The reason for this is unknown and it is possible that YouTube had changed their algorithms or something in the test setup caused patterns to be different. Earlier studies have included different profiles for different setups, too. The measurements in this thesis were done using a standard YouTube web page so the change of the algorithm could be done without interference to special YouTube application users. Dedicated YouTube application behaviour was not measured or studied in this thesis. During testing it was developed several *Matlab* scripts and *Python* scripts for the data analysis. These scripts and testing methods could also be used to study other kinds of data formats which are transmitted over the Internet.

It was shown that 97 % of the data transmitted during YouTube video streaming was formed by two TCP/IP streams. Additionally, several other smaller TCP/IP streams were noticed during viewing. They were called background noise in this thesis. YouTube servers transmit data periodically and in time level three different main phases can be recognized: *speedup* at the beginning of the video viewing, *steady phase* where data is sent periodically and finally the *video tail*, where a video is still viewed but all the data has already been sent. The two major TCP streams sent data in bursts which were called chunks in this thesis. Two different streams could be separated based on the chunk sizes, i.e. low stream and high stream. The high stream chunk size was 4 % of the total received video size, and the low stream chunk size was 244498 bytes. The time difference between the high stream chunks was 15 seconds whereas between the low stream chunks it was 14.9 seconds. The whole of the data of the video was transmitted when 74 % of the viewing time had elapsed. Other TCP streams caused several small chunks, which seemed to follow the exponential distribution with the mean value of 3.08067 seconds. These other TCP streams are called as noise in this thesis. Thus, high regularity in YouTube video transmission profile was recognized and it was used to create a simple model for YouTube videos. This model can be used on top of a system level simulator instead of an infinite constant rate buffer model to provide valuable in-

sight for mobile broadband system designers about the effect of network parameterization in terms of QoS and energy consumption of mobile devices.

Theoretical RF activity times were calculated for the data with the estimated constant throughput values of 20 Mbps in DL and 2 Mbps in UL. The results show that the minimum RF time is only 3.6 % of the total video viewing time. So there is a huge possibility for DRX operation. It was also observed that with slow throughput values TDD kept RF on more than FDD but when throughput was increased the difference became smaller. In general, RF activity time decreased in the same proportion as throughput increased. Unfortunately the noise packets make RF activity pauses very short, which makes RF circuitry sleeping difficult. Luckily, YouTube video watching does not require real time transmission with strict delay constraints. Therefore, sending the noise packets could be delayed to occur only when two major TCP streams are transmitting. This makes RF non-activity periods longer between transmissions and receptions, which means that longer DRX periods are possible. As a side effect, there will be delays of several seconds in the noise packets and there must be extra buffer capacity in the MS and in the network. These delays follow an exponential distribution with the mean value of 5.7 seconds.

LTE network uses a timer to move an MS from the RRC_CONNECTED state to the RRC_IDLE state in case there are longer periods without data transmission or reception. The effect of this promotion timer was studied and the timer value of 11.576 seconds gave approximately 3 % energy savings and a 5-second timer gave over 20 % energy savings if the noise packets were not delayed. While the noise packets were delayed, the corresponding values were 7.6 % and over 40 %. It was observed that delaying the noise packets can give over 30 % energy savings with small timer values when compared to the situation without delaying while using the same promotion timer value. But moving between the RRC_CONNECTED state and the RRC_IDLE state causes extra signalling and an extra delay of a few hundreds of milliseconds and, in addition, with small promotion timer values the timer expirations increase heavily. There can still be small RF non-activity periods between the TCP packets in the two major TCP streams. So, besides delaying noise packets, delaying packets belonging to the same chunk in the two major TCP streams is beneficial from the spectral efficiency point of view and can be expected to have only a minor effect on performance. For example, by using 200 ms buffering window to convert RX activity from a packet based activity to continuous chunk based activity could increase the spectral efficiency of the DL transmission significantly because there would be no RF non-activity between the separate packets inside the chunk.

In further studies it could be examined more closely what the particular noise TCP streams were that occurred during video viewing. Furthermore, it could be studied if those streams appear the same way before and after the actual video viewing in order to see if they really are related to video transmission. The power consumption model used in this thesis could not differentiate whether there appeared DRX or not during the RRC_CONNECTED state in the LTE. This power consumption could be measured us-

ing e.g. a LTE radio tester in laboratory environment. This way a more accurate model could be derived to estimate MS energy consumption with LTE DRX timer values.

The future network standardisation should pay attention to energy efficiency during a connected mode DRX. Also, using buffering and delaying the packets when traffic is delay insensitive - like YouTube - could provide benefits. The most energy efficient way would be transmitting all the data inside the chunks without any transmission pauses. Because video data amounts are increasing all the time, standardisation should carefully study the effects of video profiles and the first step would be to use real life models to simulate the performance. Furthermore, service providers, network operators, network vendors and standard organisations could standardize video transmission algorithms as well as try to optimize DRX and DTX opportunities in the future mobile networks.

REFERENCES

[1] Cisco Systems Inc, "Cisco Visual Networking Index: Forecast and Methodology, 2012-2017," [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html. [Accessed 29 May 2014].

[2] International Telecommunication Union, “Report ITU-R M.2370-0; IMT traffic estimates for the years 2020 to 2030,” Geneva, 2015.

[3] International Telecommunication Union, ”Report ITU-R M.2243; Assessment of the global mobile broadband deployments and forecasts for International Mobile Telecommunications,” Geneva, 2011.

[4] International Telecommunication Union, ”Report ITU-R M.2290-0; Future spectrum requirements estimate for terrestrial IMT,” Geneva, 2014.

[5] Sandvine Inc, “Global Internet Phenomena Report,” 2015. [Online]. Available: https://www.sandvine.com/trends/global-internet-phenomena/.

[6] 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects, *3GPP TS 23.203 V10.10.0; Policy and charging control architecture,* 2014.

[7] P. Ameigeiras, J. J. Ramos-Munoz, J. Navarro-Ortiz and J. M. Lopez-Soler, "Analysis and modelling of YouTube traffic," in *Transactions on emerging telecommunication technologies*, 2012.

[8] Wireshark organisation, "Wireshark," 2014. [Online]. Available: http://www.wireshark.org/.

[9] J. J. Ramos-Munoz, J. Prados-Garzon, P. Ameigeras, P. Navarro-Ortiz and J. M. Lopez-Soler, "Characteristics of Mobile YouTube Traffic," *IEEE Wireless Communications,* no. February, pp. 18-25, 2014.

[10] A. Finamore, M. Mellia, M. M. Munafò, R. Torres and S. G. Rao, “YouTube everywhere: impact of device and infrastructure synergies on user experience,” in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, Berlin, 2011.

[11] A. Rao, L. Yeon-sup, C. Barakat, A. Legout, D. Towsley and W. Dabbous, "Network Characteristics of Video Streaming Traffic," in *CoNEXT'11 Proceedings of the Seventh Conference on emerging Networking Experiments and Technologies, Article No. 25*, 2011.

[12] J. Prados-Garzon, P. Ameigeiras, J. Navarro-Ortiz and J. M. Lopez-Soler,

"Simulation-Based Performance Study of YouTube Service in 3G LTE," in *2013 IEEE 14th International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2013, ; Madrid; Spain; 4 June 2013 through 7 June 2013*, 2013.

[13] R. Li, Z. Zhao, X. Zhou, J. Palicot and H. Zhang, "The Prediction Analysis of Cellular Radio Access Network Traffic: From Entropy Theory to Networking Practice," *IEEE Communications Magazine,* no. June 2014, pp. 234-240.

[14] S. Ghandali and S. M. Safavi, “Modeling Multimedia Traffic in IMS Network Using MMPP,” in *2011 3rd International Conference on Electronics Computer Technology*, India, 2011.

[15] C. R. Baugh and J. Huang, *Traffic Model for 802.16 TG3 MAC/PHY Simulations,* IEEE 802.16 Broadband Wireless Access Working Group, 2001.

[16] S. Tanwir and H. Perros, “A Survey of VBR Video Traffic Models,” *IEEE Communications Surveys & Tutorials,* vol. 15, no. 4, 2013.

[17] Samsung, ”Samsung,” 2014. [Online]. Available: http://www.samsung.com/sg/consumer/mobile-devices/smartphone/android-os/GT-I9506ZWAXSP.

[18] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen and O. Spatscheck, “A Close Examination of Performance and Power Characteristics of 4G LTE Networks,” in *MobiSys'12*, Low Wood Bay, 2012.

[19] 3rd Generation Partnership Project; Technical Specification Group Radio Access Network, *3GPP TS 36.300: "Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall Description; Stage 2 (Release 10)",* V10.11.0 ed., 2013.

[20] 3rd Generation Partnership Project; Technical Specification Group Radio Access Network, *3GPP TS 36.321: "Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification (Release 10)".*

[21] 3rd Generation Partnership Project, *3GPP TS 36.331 V10.14.0; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification (Release 10),* 2014.

[22] C. S. Bontu ja E. Illidge, ”DRX Mechanism for Power Saving in LTE,” *IEEE Communications Magazine,* June 2009.

[23] E. Dahlman, S. Parkvall, J. Sköld ja P. Beming, 3G Evolution: HSPA and LTE for Mobile Broadband, Elsevier Ltd., 2008.

[24] T. Kolding, J. Wigard and L. Dalsgaard, “Balancing Power Saving and Single User Experience with Discontinuos Reception in LTE,” in *Wireless Communication Systems. 2008. ISWCS '08. IEEE International Symposium on*, Reykjavik, 2008.

[25] 3rd Generation Partnership Project, *3GPP TR 25.982 v6.0.0; Technical*

Specification Group Radio Access Network; Feasibility Study for Orthogonal Frequency Division Multiplexing (OFDM) for UTRAN enhancement (Release 6), 2004.

[26] M. Polignano, D. Vinella, D. Laselva, J. Wigard and T. B. Sörensens, "Power Savings and QoS Impact for VoIP Application with DRX / DTX Feature in LTE," in *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*, Budapest, 2011.

[27] K. Aho, T. Henttonen, J. Puttonen, L. Dalsgaard and T. Ristaniemi, "User Equipment Energy Efficiency versus LTE Network Performance," *International Journal on Advances in Telecommunications,* no. 3, 2010.

[28] S. Herrería-Alonso, M. Rodríguez-Pérez, M. Fernández-Veiga and C. López-García, "Adaptive DRX Scheme to Improve Energy Efficiency in LTE Networks with Bounded Delay," *IEEE Journal on Selected Areas in Communications,* vol. 33, no. 12, December 2015.

[29] K. Christensen, P. Reviriego, B. Nordman, M. Bennett, M. Mostowfi and J. A. Maestro, "IEEE 802.3az: The Road to Energy Efficient Ethernet," *IEEE Communications Magazine,* November 2010.

[30] M. A. Hoque, M. Siekkinen, J. K. Nurminen, M. Aalto and S. Tarkoma, "Mobile multimedia streaming techniques: QoE and energy saving perspective," *Pervasive and Mobile Computing,* no. 16, pp. 96-114, 2015.

[31] Y.-W. Chen, M.-H. Lin and C.-C. Huang, "Study of Buffer Aware Scheduling for Video Streaming in LTE network," in *17th International Symposium on Wireless Personal Multimedia Communications (WPMC2014)*, 2014.

[32] M. Siekkinen, M. A. Hoque, J. K. Nurminen and M. Aalto, "Streaming over 3G and LTE: how to save smartphone energy in radio access network-friendly way," in *MoVid '13 Proceedings of the 5th Workshop on Mobile Video*, Oslo, 2013.

[33] S. G. Lee, J. Park and H. Kim, "A User-Side Energy-Saving Video Streaming Scheme for LTE Devices," *IEEE Communications Letters,* vol. 19, no. 6, 2015.

[34] M. A. Hoque, M. Siekkinen and J. K. Nurminen, "Energy Efficient Multimedia Streaming to Mobile Devices - A Survey," *IEEE Communications Surveys & Tutorials,* vol. 16, no. 1, pp. 579-597, 2014.

[35] S. Deng and H. Balakrishnan, "Traffic-aware techniques to reduce 3G/LTE wireless energy consumption," in *Proceedings of the 8th international conference on emerging networking experiments and technologies*, 2012.

[36] G. Foddis, R. G. Garroppo, S. Giordano, G. Procissi, S. Roma and S. Topazzi, "LTE Traffic Analysis for Signalling Load and Energy Consumption Trade-Off in Mobile Networks," in *2015 IEEE International Conference on Communications (ICC)*, London, 2015.

[37] A. Aqil, A. O. F. Atya, S. V. Krishnamurthy and G. Papageorgiou, "Streaming

Lower Quality Video over LTE : How Much Energy Can You Save ?," in *2015 IEEE 23rd International Conference on Network Protocols (ICNP)*, San Francisco.

[38] Y. Xiao, R. S. Kalyanaraman and A. Yla-Jaaski, "Energy Consumption of Mobile YouTube:Quantitative Measurement and Analysis," in *The Second International Conference on Next Generation Mobile Applications, Services, and Technologies*, 2008.

[39] W. Lee, J. Koo, S. Jin and S. Choi, "EQ-Video: Energy and Quota-Aware Video Playback Time Maximation for Smartphones," *IEEE Communication Letters,* vol. 19, no. 6, pp. 1045-1048, 2015.

[40] P. J. Huber and E. M. Ronchetti, ROBUST STATISTICS, John Wiley & Sons, Inc, 2009.

[41] J. G. Proakis, DIGITAL COMMUNICATIONS, Third Edition, McGraw-Hill Book Co., 1995.

[42] MathWorks Inc., "normcdf," [Online]. Available: http://se.mathworks.com/help/stats/normcdf.html. [Accessed 23 November 2015].

[43] F. Zheng and S. Zhong, "Time series forecasting using a hybrid RBF neural network and AR model based on binomial smoothing," *International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering,* vol. 5, no. 3, 2011.

[44] N. Vogt, "YouTube audio quality bitrate used for 240p, 360p, 480p, 720p, and 1080p," July 2012. [Online]. Available: http://www.h3xed.com/web-and-internet/youtube-audio-quality-bitrate-240p-360p-480p-720p-1080p.

[45] Google, "Upload instructions and settings," 2014. [Online]. Available: https://support.google.com/youtube/answer/1722171?hl=en.

[46] MathWorks Inc., "expcdf," [Online]. Available: http://se.mathworks.com/help/stats/expcdf.html. [Accessed 23 November 2015].

[47] M. H. Faber, Statistics and Probability Theory, Springer, 2012.

[48] Microsoft, "Microsoft Network Monitor," 2014. [Online]. Available: http://www.microsoft.com/en-us/download/details.aspx?id=4865.

[49] Anite plc, "Nemo Outdoor - the ultimate drive test tool for wireless networks," Anite, 2014. [Online]. Available: http://www.anite.com/businesses/network-testing/products/nemo-outdoor-ultimate-drive-test-tool-wireless-networks#.VAgUsGNAVaQ.

[50] Huawei, "Huawei Enterprise Support Community," 16 March 2014. [Online]. Available: http://support.huawei.com/ecommunity/bbs/10182143.html. [Accessed 10 January 2016].

[51] G. Gorbil, O. H. Abdelrahman and E. Gelenbe, "Storms in mobile networks," in *Proceedings of the 10th ACM symposium on QoS and security for wireless and mobile networks (Q2SWinet'14)*, Montreal, 2014.

APPENDIX

```
%{
Models YouTube Transmission
After the run:
Major stream DL chunks are in: chunkMajorTotal
Corresponding major stream time stamps are in: chunkTimeTotal
Noise stream UL+DL chunks are in: chunkNoiseTotal
Corresponding noise time stamps are in: noiseTimeTotal

All IP packets (UL+DL) are in time order in matrix: ipData
where the first column is time in seconds
the second column is byte amount
and the third column is direction, where 1=UL, 0=DL

All IP packets are in time order in cell array: ipCell
where the first column is time in seconds
the second column is byte amount
and the third column is direction, either "UL" or "DL"

Finally the cell array is written in file.

Plotted picture of the chunks contains:
- Major stream DL chunks in speedup in magenta
- Major stream DL chunks in steady phase in blue
- Noise stream UL+DL chunks in red

%}


clear
% Parameters:

% Change this parameter for different file sizes
% This does not include noise, this is
% major Streams total byte amount in DL.
% Major streams made 97% of the total data
% in LAN these were between 16939494 - 39891626

fileSize=27564396;

% Video length in seconds
% In LAN these were between 272 - 468
% Change this parameter for video length
videoLength=324;

% In normal run  - no changes are needed in parameters after this line
%-----------------------------------------------------%
% Change this if you want to have lower majorStream size than fileSize
majorStream=fileSize;
```

```
% DL Throughput. Affects how close high and low stream can be
% IP packet table works best for 20Mbits/s because values are from
that
% connection
throughput=20e6;

% CDF for noise size chunk probabilities (bytes)
nCDF=[1 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400
1500 ...
    1600 1700 1800 1900 2000 2100 2200 2300 2400 2500 2600 2700 2800
4700 ...
    4800 6000 9000 10000 20000 30000 40000 50000 60000 70000 80000
90000 100000 ...
    110000 120000 130000 140000 150000 160000 170000 180000 190000
200000 ...
    250000 300000 400000; ...
    0 20.3 27.9 34.6 35.4 37.4 38.2 41 41.4 41.8 42.4 42.7 43.6 51.7
52.3 ...
    54.1 60.5 61.9 62.7 63.2 63.7 63.9 64 64.2 79.7 80.4 80.9 81.3
81.5 ...
    82.7 90.2 91.4 92.9 93.9 94.8 95.2 96.3 96.8 96.9 97.3 97.5 97.7
97.9 ...
    98 98.1 98.4 98.6 98.6 99 99 99.3 99.4 99.4 99.6 99.8 100];

% CDF for major streams sizes speedup size (kBytes)
spmCDF=[0.001 5 238 239 653 654 838 839 892 893 1070 1080 1130 1140
1150 1160 ...
    1170 1310 1320 1370 1380 1390 1400 1410;
    0 4.5 4.5 37.9 37.9 40.9 40.9 42.4 42.4 45.5 45.5 48.5 48.5 57.6
59.1 ...
    65.2 68.2 68.2 69.7 69.7 72.7 80.3 90.9 100];

% CDF for major streams time differences in speedup phase, seconds
sptCDF=[0.37 0.5 0.75 1 1.25 1.5 1.75 2 2.25 2.5 2.75 3 3.25 3.75 4
4.19;
    0 5.4 21.4 37.5 46.4 51.8 71.4 78.6 83.9 91.1 92.9 92.9 96.4 96.4
...
    96.4 100];

% CDF for high stream sizes in steady phase (kBytes)
hszCDF=[699 700 800 900 1000 1100 1130 1135 1140 1145 1150 1155 1160
1165 1170 ...
    1200 1300 1400;
    0 0.8 0.8 2.4 3.3 5.7 8.9 21.1 33.3 46.3 46.3 56.9 75.6 85.4 99.2
99.2 99.2 100];

% CDF for high stream time differences in steady phase, seconds
hstCDF=[11.55 12 12.5 13 13.5 14 14.5 15 15.5 16 16.5 17 17.5 18 18.5
19 ...
19.5 20 20.5 21 21.5 22 22.5 23 23.5 24 24.5 25 25.5 26.5 27 28;
0 1.0 1.0 1.0 1.0 2.9 4.8 16.2 50.5 65.7 75.2 78.1 81.9 87.6 92.4 92.4
...
93.3 94.3 94.3 94.3 95.2 95.2 95.2 96.2 96.2 96.2 97.1 97.1 98.1 98.1
...
99.0 100];

% CDF for low stream sizes in steady phase (kBytes)
lszCDF=[0.001 50 100 150 200 238.7675 238.7676 240 300 400 450;
    0 13.7 15.1 17.8 18.7 19.6 88.1 98.2 98.6 98.6 100];
```

```
% CDF for low stream time differences in steady phase, seconds
lstCDF=[0.325 2.5 5 10 14 14.7 14.8 14.9 15 15.1 15.2 15.3 16 23 24 26
27 28 ...
    29 30 30.2;
    0 5.3 9.1 12.9 18.2 22.0 26.3 40.7 67.5 86.1 91.9 93.3 94.7 94.7
...
    95.7 95.7 96.2 96.7 96.7 97.6 100];

% CDF for IP packet time differences for major streams, ms
tcpMajorCDF=[0 0.03 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.2 1.3;
    0 25 37 41 47 55 61 71 87 94 96 99 100];

% These parameters give minimum distance between the high/low chunks
in seconds
% This is for probability that high and low stream chunks are so close
after IP
% calculation that they will merge.
% minimum from high to low (1200 kBytes/20e6 bits/s)+0.33
minDistHL=1200*1024*8/throughput+0.33;
% minimum from low to high from measurements
minDistLH=244498*8/throughput+0.33;

% Speedupsize is 19.3%
% Major streams portion of speedup in DL
majorSpeedupSize = majorStream * 0.193;

chunkMajorTotal=[];
chunkTimeTotal=[];
chunkNoiseTotal=[];
noiseTimeTotal=[];
highTimeMajor=0;
prevTimeNoise=0;

% for plotting
clf
hold
xlabel('Time / Seconds');
ylabel('Chunk Sizes / kBytes');
grid;

%First chunks for speedup in major streams

while majorSpeedupSize>0
    % random is in kBytes
    chunkMajor=round(FindRandom(spmCDF)*1024);
    majorSpeedupSize=majorSpeedupSize-chunkMajor;
    chunkMajorTotal=[chunkMajorTotal chunkMajor];
    % time (starts from 0)
    chunkTimeTotal=[chunkTimeTotal highTimeMajor];
    % time for next chunk
    highTimeMajor=FindRandom(sptCDF)+highTimeMajor;
    % if too close
    lastChunk=length(chunkTimeTotal);
    minDistSpeed=chunkMajorTotal(lastChunk)*8/throughput+0.2;
    % to prevent overlapping of IP packets
    while (highTimeMajor-chunkTimeTotal(lastChunk))<minDistSpeed
        highTimeMajor=highTimeMajor+0.1;
    end
end
plot(chunkTimeTotal,chunkMajorTotal/1024,'*m')
```

```
% How many bytes left
remainMajorStream=majorStream - sum(chunkMajorTotal);

% Ending points in speedup
% If the last is over 1300 KB then low and high stream chunks were
combined

% high stream ending
oldHighPoint=max(find(chunkMajorTotal>5e5));
highTimeMajor=chunkTimeTotal(oldHighPoint);

% low stream ending point
% it is either under 5e5 or over 1300 kB
oldLowPoint=max(find(chunkMajorTotal<5e5 |
chunkMajorTotal>(1300*1024)));
lowTimeMajor=chunkTimeTotal(oldLowPoint);

% Next must be calculated what is time difference between high stream
chunks
% High stream ends when 74% of the video has been viewed
% And one high steam chunk contains 4% of the DL data
highChunkSize=majorStream*0.042;

% Maximum should be under 1200 kBytes
if (highChunkSize/1024)>1200
    highChunkSize=1200*1024;
end

% How much playing time left with low stream
endTime=videoLength*.74-lowTimeMajor;

% Low stream size approximation
lowTotalSize=endTime/14.1*244498;

% How much playing time left with high stream
endTime=videoLength*.74-highTimeMajor;

% Finally, time difference between high stream chunks
% Only full sized packets are taken

highDiff=endTime/floor((remainMajorStream-
lowTotalSize)/highChunkSize);

% Table gives correct values if average is 1138 kBytes
% Thats why it is balanced to give the same average as
% highChunkSize

if (hszCDF(1,1)+highChunkSize/1024-1138)<0
    % if the lowest would be negative
    hszCDF(1,:)=hszCDF(1,:)-hszCDF(1,1);
    disp('High stream chunk size is too low');
else
    hszCDF(1,:)=hszCDF(1,:)+highChunkSize/1024-1138;
end

% And time table gives correct values if average is 16.2945 seconds
% Thats why it is balanced
```

```
if (hstCDF(1,1)+highDiff-16.2945)<0
    % if the lowest would be negative
    hstCDF(1,:)=hstCDF(1,:)-hstCDF(1,1);
    disp('High stream chunk time difference is too low');
else
    hstCDF(1,:)=hstCDF(1,:)+highDiff-16.2945;
end

% this tells in the next plot where the previous ended
nextPlot=length(chunkMajorTotal)+1;

% Then major stream chunks are calculated for steady phase
% Values for next time points

highTimeMajor=FindRandom(hstCDF)+highTimeMajor;
lowTimeMajor=FindRandom(lstCDF)+lowTimeMajor;
% First if too close to the previoius ones
minDistSpeed=chunkMajorTotal(oldLowPoint)*8/throughput+0.2;
% to prevent overlapping of IP packets low to low
while (lowTimeMajor-chunkTimeTotal(oldLowPoint))<minDistSpeed
    lowTimeMajor=lowTimeMajor+0.1;
end
minDistSpeed=chunkMajorTotal(oldHighPoint)*8/throughput+0.2;
% to prevent overlapping of IP packets high to low
while (lowTimeMajor-chunkTimeTotal(oldHighPoint))<minDistSpeed
    lowTimeMajor=lowTimeMajor+0.1;
end
% to prevent overlapping of IP packets high to high
while (highTimeMajor-chunkTimeTotal(oldHighPoint))<minDistSpeed
    highTimeMajor=highTimeMajor+0.1;
end
% to prevent overlapping of IP packets high to low
minDistSpeed=chunkMajorTotal(oldLowPoint)*8/throughput+0.2;
while (highTimeMajor-chunkTimeTotal(oldLowPoint))<minDistSpeed
    highTimeMajor=highTimeMajor+0.1;
end

% If too close then the newest one is moved
if (lowTimeMajor<=highTimeMajor)
    if (abs(lowTimeMajor-highTimeMajor)<minDistLH)
        highTimeMajor=lowTimeMajor+minDistLH;
    end
else
    if (abs(lowTimeMajor-highTimeMajor)<minDistHL)
        lowTimeMajor=highTimeMajor+minDistHL;
    end
end

% Major stream chunks are calculated

while remainMajorStream>0
    if highTimeMajor<=lowTimeMajor
        % random is in kBytes
        chunkMajor=round(FindRandom(hszCDF)*1024);
        if chunkMajor>remainMajorStream
            chunkMajor=remainMajorStream;
        end
        remainMajorStream=remainMajorStream-chunkMajor;
        chunkMajorTotal=[chunkMajorTotal chunkMajor];
```

```
            chunkTimeTotal=[chunkTimeTotal highTimeMajor];
            % time for next chunk
            highTimeMajor=FindRandom(hstCDF)+highTimeMajor;
            % if too close
            % then the newest one is moved
                if (lowTimeMajor<=highTimeMajor)
                    if (abs(lowTimeMajor-highTimeMajor)<minDistLH)
                        highTimeMajor=lowTimeMajor+minDistLH;
                    end
                else
                    if (abs(lowTimeMajor-highTimeMajor)<minDistHL)
                        lowTimeMajor=highTimeMajor+minDistHL;
                    end
                end
        % low stream values are calculated when time for low stream is
        % less than high steam
        else
            chunkMajor=round(FindRandom(lszCDF)*1024);
            if chunkMajor>remainMajorStream
                chunkMajor=remainMajorStream;
            end
            remainMajorStream=remainMajorStream-chunkMajor;
            chunkMajorTotal=[chunkMajorTotal chunkMajor];
            chunkTimeTotal=[chunkTimeTotal lowTimeMajor];
            % time for next chunk
            lowTimeMajor=FindRandom(lstCDF)+lowTimeMajor;
            % In case high and low stream chunks would be too close
            if (lowTimeMajor<=highTimeMajor)
                if (abs(lowTimeMajor-highTimeMajor)<minDistLH)
                    highTimeMajor=lowTimeMajor+minDistLH;
                end
            else
                if (abs(lowTimeMajor-highTimeMajor)<minDistHL)
                    lowTimeMajor=highTimeMajor+minDistHL;
                end
            end
        end
    end
    plot(chunkTimeTotal(nextPlot:length(chunkTimeTotal)),chunkMajorTotal(n
    extPlot:length(chunkTimeTotal))/1024,'*b')

    % Then noise chunks for the whole clip
    % Loop until end of major stream is found
    while prevTimeNoise<videoLength
        noiseTimeTotal=[noiseTimeTotal prevTimeNoise];
        % Noise stream chunk size bytes
        chunkNoise=round(FindRandom(nCDF));
        chunkNoiseTotal=[chunkNoiseTotal chunkNoise];
        % Noise stream time difference mean=3.08067 seconds
        u=rand(1,1);
        prevTimeNoise=expinv(u,3.08067)+prevTimeNoise;
        % if too close
        lastChunk=length(chunkNoiseTotal);
        minDistSpeed=chunkNoiseTotal(lastChunk)*8/throughput+0.2;
        % to prevent overlapping of IP packets
        while (prevTimeNoise-noiseTimeTotal(lastChunk))<minDistSpeed
            prevTimeNoise=prevTimeNoise+0.1;
        end
    end
    plot(noiseTimeTotal,chunkNoiseTotal/1024,'*r')
```

```
% Next IP packets for major streams
% Constant size 1495 bytes in DL and 49 bytes in UL
ipData=[];
sizeIPUL=49;
sizeIPDL=1495;
% Data will be in matrix in format: time amount direction
% direction=1 is UL, 0 is DL
% Example: 1.2 1495 0

dlPacketAmount=0;
ulPacketAmount=0;
% ratio should be dlPacketAmount=ulPacketAmount*1.59
for i= 1:length(chunkMajorTotal)
    % chunk size is only for DL bytes in major streams
    chunkSize=chunkMajorTotal(i);
    ipTime=chunkTimeTotal(i);
    startTime=ipTime;
    while(chunkSize>0)
        if (ulPacketAmount*1.58885<=dlPacketAmount)
            ipData=[ipData;ipTime sizeIPUL 1];
            ulPacketAmount=ulPacketAmount+1;
        else
            if(chunkSize-sizeIPDL)<0
                % it is possible to get unrealistic low sizes
                ipData=[ipData;ipTime chunkSize 0];
            else
                ipData=[ipData;ipTime sizeIPDL 0];
            end
            chunkSize=chunkSize-sizeIPDL;
            dlPacketAmount=dlPacketAmount+1;
        end
        % time is ms in table and must be converted to seconds
        ipTime=FindRandom(tcpMajorCDF)/1000+ipTime;
    end
end


% IP packets for noise streams
dlPacketAmount=0;
ulPacketAmount=0;
sizeIPUL=179;
sizeIPDL=781;

for i= 1:length(chunkNoiseTotal)
    % chunk size is for UL and DL bytes in noise streams
    chunkSize=chunkNoiseTotal(i);
    ipTime=noiseTimeTotal(i);
    while(chunkSize>0)
        if (ulPacketAmount*1.1225<=dlPacketAmount)
            if(chunkSize-sizeIPUL)<0
                % it is possible to get unrealistic low sizes
                ipData=[ipData;ipTime chunkSize 1];
            else
                ipData=[ipData;ipTime sizeIPUL 1];
            end
            chunkSize=chunkSize-sizeIPUL;
            ulPacketAmount=ulPacketAmount+1;
        else
            if(chunkSize-sizeIPDL)<0
                ipData=[ipData;ipTime chunkSize 0];
            else
```

```
                ipData=[ipData;ipTime sizeIPDL 0];
            end
            chunkSize=chunkSize-sizeIPDL;
            dlPacketAmount=dlPacketAmount+1;
        end
        % time is ms in table, the same table is used as for major
stream
        ipTime=FindRandom(tcpMajorCDF)/1000+ipTime;
    end
end

% order is sorted according to time
ipData=sortrows(ipData);

% the data can be written to a file

%first ipData is converted to cell array with text
%this array will have format "time size direction"
%where direction is either 'UL' or 'DL'
%example: 0.001045 1495 DL

ipCell=cell(length(ipData),3);
for i=1:length(ipData)
    if (ipData(i,3)==1)
        ipCell(i,1:3)={ipData(i,1),ipData(i,2),'UL'};
    else
        ipCell(i,1:3)={ipData(i,1),ipData(i,2),'DL'};
    end
end

%{

% output directory
cd('C:\Users\Administrator.WINDOWS-
KQQRRGM\Documents\Koulu\Mittaukset');

% then write to file
fileID = fopen('model_ip_data.txt','w');

formatSpec = '%f %d %s\r\n';
nrows= length(ipCell);
for row = 1:nrows
    fprintf(fileID,formatSpec,ipCell{row,:});
end
fclose(fileID);

%}
```

```
function [ randomResult ] = FindRandom( cdfTable )
% This function returns a random value based on cdfTable
% In cdfTable the first row gives values and the second row the
% cumulative propability in percents for that value e.g. if the first
row has a value
% 100 and the second row value 45, it means that the propability for
value to be 0-100
% is 0.45. Inside the levels uniform distribution is assumed.
% the first values in cdfTable must be 0 and the last value 100
```

```
% size of the matrix
n=size(cdfTable,2);
% First random variable
u=rand(1,1);
% Find the place in the table and the limits
for i=1:n
    if u<=(cdfTable(2,i)/100)
        upper_limit=cdfTable(1,i);
        lower_limit=cdfTable(1,i-1);
        break;
    end
end
% Second random
u=rand(1,1);
randomResult=(upper_limit-lower_limit)*u+lower_limit;
```