Adaptive Video Streaming

Adapting video quality to radio links with different characteristics

WILLIAM EKLÖF

KTH Information and Communication Technology

Master of Science Thesis
Stockholm, Sweden 2008

COS/CCS 2008-28

Adaptive Video Streaming

Adapting video quality to radio links with different characteristics

William Eklöf
<williame[at]kth.se>

Master thesis

2008-12-09

Examiner and academic supervisor, KTH: *Gerald Q. Maguire Jr.*
Industrial supervisor, Ericsson Research: *Kristofer Dovstam*

Abstract

During the last decade, the data rates provided by mobile networks have improved to the point that it is now feasible to provide richer services, such as streaming multimedia, to mobile users. However, due to factors such as radio interference and cell load, the throughput available to a client varies over time. If the throughput available to a client decreases below the media's bit rate, the client's buffer will eventually become empty. This causes the client to enter a period of rebuffering, which degrades user experience. In order to avoid this, a streaming server may provide the media at different bit rates, thereby allowing the media's bit rate (and quality) to be modified to fit the client's bandwidth. This is referred to as *adaptive streaming*.

The aim of this thesis is to devise an algorithm to find the media quality most suitable for a specific client, focusing on how to detect that the user is able to receive content at a higher rate. The goal for such an algorithm is to avoid depleting the client buffer, while utilizing as much of the bandwidth available as possible without overflowing the buffers in the network. In particular, this thesis looks into the difficult problem of how to do adaptation for live content and how to switch to a content version with higher bitrate and quality in an optimal way.

This thesis examines if existing adaptation mechanisms can be improved by considering the characteristics of different mobile networks. In order to achieve this, a study of mobile networks currently in use has been conducted, as well as experiments with streaming over live networks. The experiments and study indicate that the increased available throughput can **not** be detected by passive monitoring of client feedback. Furthermore, a higher data rate carrier will **not** be allocated to a client in 3G networks, unless the client is sufficiently utilizing the current carrier. This means that a streaming server must modify its sending rate in order to find its maximum throughput and to force allocation of a higher data rate carrier. Different methods for achieving this are examined and discussed and an algorithm based upon these ideas was implemented and evaluated. It is shown that increasing the transmission rate by introducing stuffed packets in the media stream allows the server to find the optimal bit rate for live video streams without switching up to a bit rate which the network can not support.

This thesis was carried out during the summer and autumn of 2008 at Ericsson Research, Multimedia Technologies in Kista, Sweden.

Sammanfattning

Under det senaste decenniet har överföringshastigheterna i mobilnätet ökat så pass mycket att det nu är möjligt att erbjuda användarna mer avancerade tjänster, som till exempel strömmande multimedia. I mobilnäten varierar dock klientens bandbredd med avseende på tiden på grund av faktorer som störningar på radiolänken och lasten i cellen. Om en klients överföringshastighet sjunker till mindre än mediets bithastighet, kommer klientens buffert till slut att bli tom. Detta leder till att klienten inleder en period av ombuffring, vilket försämrar användarupplevelsen. För att undvika detta kan en strömmande server erbjuda mediet i flera olika bithastigheter, vilket gör det möjligt för servern att anpassa bithastigheten (och därmed kvalitén) till klientens bandbredd. Denna metod kallas för adaptive strömning.

Syftet för detta examensarbete är att utveckla en algoritm, som hittar den bithastighet som är bäst lämpad för en specifik användare med fokus på att upptäcka att en klient kan ta emot media av högre kvalité. Målet för en sådan algoritm är att undvika att klientens buffert blir tom och samtidigt utnyttja så mycket av bandbredden som möjligt utan att fylla nätverksbuffertarna. Mer specifikt undersöker denna rapport det svåra problemet med hur adaptering för direktsänd media kan utföras.

Examensarbetet undersöker om existerande adapteringsmekanismer kan förbättras genom att beakta de olika radioteknologiers egenskaper. I detta arbete ingår både en studie av radioteknologier, som för tillfället används kommersiellt, samt experiment med strömmande media över dessa. Resultaten från studien och experimenten tyder på att ökad bandbredd **inte** kan upptäckas genom att passivt övervaka ”feedback” från klienten. Vidare kommer **inte** användaren att allokeras en radiobärare med högre överföringshastighet i 3G-nätverk, om inte den nuvarande bäraren utnyttjas maximalt. Detta innebär att en strömmande server måste variera sin sändningshastighet både för att upptäcka om mer bandbredd är tillgänglig och för att framtvinga allokering av en bärare med högre hastighet. Olika metoder för att utföra detta undersöks och diskuteras och en algoritm baserad på dessa idéer utvecklas.

Detta examensarbete utfördes under sommaren och hösten 2008 vid Ericsson Research, Multimedia Technologies i Kista, Sverige.

Table of Contents

List of figures

List of Tables

Acronyms and Abbreviations

CODEC	Coder/Decoder
CPCH	Common Packet Channel
CSRC	Contributing source
CR	Carriage Return
CS	Coding Scheme
CTCH	Common Traffic Channel
DCH	Dedicated Channel
DLSR	Delay since Last Sender Report
DSCH	Downlink Shared Channel
DTCH	Dedicated Traffic Channel
DVD	Digital Versatile Disc
EDGE	Enhanced Data Rates for the GSM Evolution
FACH	Forward Access Channel
FDD	Frequency Division Duplex
FDMA	Frequency Division Multiple Access
GGSM	Gateway GPRS Support Node
GPRS	Global Packer Radio Service
GSM	Global System for Mobile communications
GTP	GPRS Tunneling Protocol
HLR	Home Location Registry
HSCSD	High-Speed Circuit-Switched Data
HSDPA	High-Speed Downlink Packet Access
HS-DSCH	High-Speed Downlink Shared Channel
HTTP	HyperText Markup Language
IETF	Internet Engineering Task Force
I-Frame	Intraframe
IMSI	International Mobile Subscriber Identity
IP	Internet Protocol
LF	Line Feed
LSR	Last Sender Report
MCS	Modulation and Coding Scheme
MPEG	Moving Picture Experts Group
MS	Mobile Station
MSC	Mobile Switching Center
NACK	Not Acknowledged
NADU	Next Application Data Unit
NAT	Network Address Translator
NTP	Network Time Protocol
P-Frame	Predicted frame
P-TMSI	Packet Temporary Subscriber Identity
PACCH	Packet Associated Control Channel
PAGCH	Packet Access Grant Channel
PBCCH	Packet Broadcast Control Channel
PC	Personal Computer
PDCH	Packet Data Channel

PDTCH	Packet Data Traffic Channel
PDN	Packet Data Network
PDP	Packet Data Protocol
PRACH	Packet Random Access Channel
PSS	Packet-Switched Streaming
PSTN	Public Switched Telephony Network
RACH	Random Access Channel
RAB	Radio Access Bearer
RAN	Radio Access Network
RFC	Request for Comments
RNC	Radio Network Controller
RTCP	Real-Time Control Protocol
RTD	Round-Trip Delay
RTP	Real-time Transport Protocol
RTSP	Real-Time Streaming Protocol
RR	Receiver Report
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SR	Sender Report
SDES	Source Description
SGSN	Serving GPRS Support Node
SSRC	Synchronization Source
TCP	Transmission Control Protocol
TDD	Time Division Duplex
TDMA	Time Division Multiple Access
TMSI	Temporary Mobile Subscriber Identity
TTI	Transmission Time Interval
TTL	Time To Live
UDP	User Datagram Protocol
UE	User Equipment
UMTS	Universal Mobile Telecommunications System
URL	Uniform Resource Locator
UTRAN	UMTS Terrestrial Radio Access Network
VLR	Visitors Location Registry
VoIP	Voice over IP
WCDMA	Wideband Code Division Multiple Access
QoE	Quality of Experience
QoS	Quality of Service
XR	Extended Report

Chapter 1: Introduction

1.1 Background

The introduction of the third generation (3G) of cellular networks and improvements to the second generation (such as *Enhanced Data Rates for the GSM Evolution* (EDGE)) have led to a rapid increase of the capacity in cellular networks. These changes have also introduced increased data rates that enable implementation of services previously only available to users accessing the Internet via wired connections, such as streaming multimedia.

In order to provide the end-user with an acceptable quality of service, a streaming server must provide its clients with data at a steady pace, to ensure that they always have media content available. This is often harder to achieve in cellular networks than in classical wired networks, since the wireless links have a higher bit error rate and the throughput may vary depending on parameters such as distance to the base station and the number of active users in the cell. Today significant variations in performance are also due to techniques which explicitly try to exploit good link quality (for example by allocating higher rate channels to specific users when the link quality is high) while deferring transmission over links which have bad link quality - thus significantly *increasing the variance in the link data rate and increasing jitter*.

The term *adaptive streaming* refers to techniques that dynamically adjust the encoded bit rate of a media stream depending on the current network conditions. To achieve this, the streaming server must continuously estimate the state of the network. RFC 3550 [1] defines the *Real-time Transport Protocol* (RTP) for carrying real-time traffic, and its companion protocol, the *Real-time Control Protocol* (RTCP), which provides the sender(s) with feedback regarding the quality experienced by the receiver(s) and provides the receiver(s) with information about what was sent.

1.2 Aim of this thesis

The purpose of this thesis is to investigate how a streaming server may detect that the bandwidth available to a client, who is connected via a cellular network, has increased and to adjust the video quality accordingly. Figure 1.1 depicts a possible scenario. A video encoder supplies the server with a video stream encoded at different bit rates (In the figure, four bit rates (32, 64, 128, and 384 kbps) are shown). The server streams the encoded video to its clients using RTP and receives feedback via RTCP. For each client, the server examines the RTCP feedback; in order to try to determine which video bit rate is the most appropriate for each client. An assumption made is that the clients are standard mobile handsets, without any additional software. Furthermore, the radio link is assumed to be the bottleneck link.

In a live streaming session, it is generally more challenging to discover when more bandwidth is available than when less is available. If the *available* bandwidth drops below the current transmission rate, then the cellular network will be forced to buffer packets since it is no longer capable of servicing all of the traffic. If the link bandwidth is kept below the transmission rate for a sufficiently long period, the link layer buffers in the mobile network will overflow and cause packet loss. Preferably, the server should detect this before the buffers overflow and lower its transmission rate (by selecting a lower (video) bit rate). Intuitively, the transfer time of each packet should increase when the bandwidth drops, since each packet spends more time in buffers, hence arrives at the receiver after a longer cumulative delay. As described in chapter 2, RFC

3550 [1] defines an algorithm for estimating the round-trip delay using RTCP information. If the transmission rate is constant and the effective bandwidth falls below that rate, this should be reflected in an increased round-trip delay[1]. By constantly estimating the round-trip delay and other parameters, a decrease in bandwidth can, typically, be detected and accommodated for by the streaming server *before* the network buffers fill up and significant packet loss occurs.

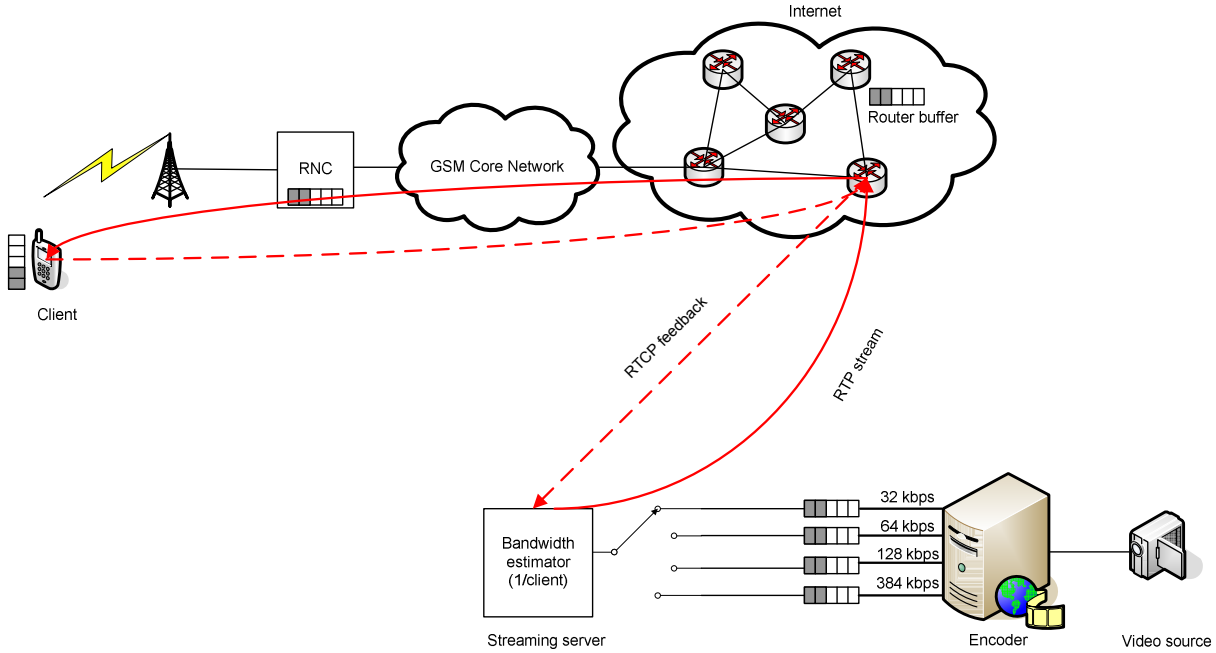

Figure 1.1: The network model assumed in this thesis

Conversely if the client's available link bandwidth increases, it might be possible for the sender to take advantage of this increased bandwidth by increasing the quality of the media. However, detecting an increase in available bandwidth is not as straightforward as detecting a decrease, because the network buffers will not be affected (at least not as much). The purpose of this thesis is to investigate if it is possible for a streaming server to *reliably* detect an increase in bandwidth by monitoring the feedback received via RTCP. Preferably, this should be done without explicitly probing the network. In order to achieve this, the unique properties of different cellular networks must be accounted for.

[1] Although it is the delay from the server to the client that is interesting, the round-trip delay can be used as an estimate of the one-way delay. It is (mainly) *changes* in the delay that are of interest, these are reflected in the round-trip delay as well as the one-way delay.

1.3 Organization of this report

This thesis is divided into eight chapters. Chapters two and three provide the theoretical background needed for reading this thesis. Chapter 2 examines techniques and protocols involved in streaming media as well as some basic media CODEC information. In chapter three some of the more common cellular network techniques in use today are investigated in order to find out how their unique properties might affect video streaming. The fourth chapter describes the algorithm developed by Ericsson Research, which this thesis builds upon. The fifth chapter contains measurement data captured during video streaming sessions both in live and emulated cellular networks. In chapter six, an algorithm for detecting which radio access technology the client is connecting through is described and tested. The seventh chapter describes and evaluates an algorithm for switching up the bit rate. The last chapter contains a project summary and conclusions, as well as suggestions for future work.

1.4 Previous work

Due to the fluctuating nature of the effective bandwidth in mobile networks, several schemes to adapt to these conditions have been proposed and implemented. There are basically two approaches to content rate adaption; either the client decides when to change the content rate or the server does. The advantage of having the client decide the content rate is that it has better knowledge of its current available rate, thus it has better knowledge about which rate it may best receive. However, this limits the service to clients that support such features. Moreover, the server might be heavily loaded, which might be a reason not to not switch up, even though the client is not optimally utilizing its link. Most of the techniques currently in use today are based upon modeling the client's receive buffer and base their adaptation mechanism upon the fullness of the receive buffer. The model of the receive buffer, if sufficiently accurate, will enable the server to discover when the client is consuming content faster than it is receiving (which will eventually cause a buffer underrun) and react accordingly.

The *Third Generation Partnership* (3GPP) [2] is a collaboration between several telecommunications standards bodies. 3GPP has standardized an extension to RTCP called the NADU APP-packet (see section 2.7.2). The purpose of the NADU-packet is to explicitly signal information regarding the client's buffer.

Previous work done at Ericsson [3][4] have investigated bit rate modulation to control the transmission rate of the media. In [3] a proxy is inserted in the network path which varies the transmission rate of the media. The proxy then decides whether to switch or not based on the difference between the sending of a RTP packet and the receiver report acknowledging reception of that packet. The algorithm in [4] also modulates the transmission rate of the media and measures the transmission time. Upswitching is based upon fixed thresholds.

A thesis by Xiaokun Yi [5] proposes a scheme to adapt to varying network conditions by switching the media CODEC (thus enabling both rate adaptation and robustness adaption). His technique mainly focuses on monitoring the packet loss along the network path. However, he also attempts to increase the rate, then observes if this is successful - if so then a higher bit rate CODEC will be used otherwise the current CODEC is used. A small amount of hysteresis is used to stabilize the selection process.

Alexander Tarnowski presents an algorithm for content switching [6], which utilizes modeling of the client's buffer. This algorithm uses an RTP extension called the RTP Retransmission Payload Format [7], to extract the information necessary for his content rate switching algorithm.

Another approach to streaming, as opposed to RTP, is *HTTP streaming* [8] or *progressive download*. As the name implies, the *HyperText Transfer Protocol* (HTTP) [9], which is the transfer protocol of the World Wide Web, is used to transfer the media. In HTTP streaming, the media file is downloaded as ordinary web pages, but playout begins just as soon as the first bytes are received (excluding client side buffering) instead of waiting for the entire file. This approach is widely used by video sharing sites on the Internet, such as YouTube [10]. However; progressive download is best suited for short pre-encoded clips, since TCP's congestion control mechanism makes HTTP streaming cumbersome for live contents and longer clips.

A third approach is to not use the traditional client-server solution and instead use a peer-to-peer architecture. Commercial applications using this approach include SOPCast and Joost. Athanasios Markis and Andreas Strikos have developed a peer-to-peer distribution system for live IPTV [11].

1.4.1 Commercial streaming products

There are several streaming servers currently on the market. This section mentions some of them and how they provide bit rate adaptation. Documentation for RealNetworks's Helix Mobile Server is publicly available and QuickTime's Darwin Server is open-source. These are described in the following sections. Other solutions include Vidiator's Xenon Streaming Server [12] and Mobixell [13]. Ericsson also has a commercial bit rate adaptation solution, based on the Ericsson Research algorithm described in chapter 4.

1.4.1.1 RealNetwork's Helix Mobile Server

RealNetworks's streaming solution, the Helix Mobile Server, only provides adaptation for pre-encoded content [14]. The adaptation technique is based upon controlling both the transmission rate and the media bit rate. A modified version of *Transport Friendly Rate Control* (TFRC) [15] is used to control the transmission rate. The server uses 3GPP PSS [16] (see section 2.7) to model the client buffer and decides when to do bit rate switching depending on the buffer fullness [14].

1.4.1.2 Apple's and QuickTime's Darwin Streaming Servers

The Darwin Streaming Server [17] is an open-source version of QuickTime Streaming Server. The Darwin server also bases its bit rate switching decisions upon a model of the client's buffer. By examining the source code, it appears as if Darwin uses the 3GPP NADU APP-packet to receive information from the client regarding the occupancy of the buffer. Upswitching is performed at regular intervals until the client's buffer is more than 50% full.

Chapter 2: Streaming Media and CODECs

Streaming media differs from ordinary media by the fact that streaming media is played out (almost) as soon as it is received; instead of waiting for the entire file to be transferred over the network as traditionally done. The main advantage of this is that a user can begin the playout of large files immediately (thus not needing to wait for the entire file to be downloaded). This enables the sender to transmit live[2] content.

Due to the large increase in network capacity and the popularization of the personal computer and the Internet in the mid-1990s, streaming audio and video to viewers over the Internet became both feasible and economical. The *Internet Engineering Task Force* (IETF) has standardized a set of protocols for carrying real-time media over computer networks. This section deals with the details of these protocols. For this thesis, RTP (Section 2.3) and RTCP (2.4) are the most important protocols, but for completeness RTSP (Section 2.1) and SDP (Section 2.2) are also described.

RTP provides the functionality needed to transport the media between the end points and provides the information necessary to reconstruct the original stream at the receiver. However, how the media should be handled after it has arrived at the destination is left up to the (local) implementation. Commonly, the receiver buffers data for a period of time before displaying/outputting it to the user. This reduces the effect of *jitter*, i.e. variations in the delay, introduced by the network. Because of this, the buffer is sometimes referred to as the *dejitter buffer*. The most common cause of jitter is due to the fact that different packets spend different amounts of time in network queues [18]. Jitter caused by the cellular networks will be discussed in chapter 3. The dejitter buffer provides additional benefits, such as the ability to conceal out-of-order delivery by the network and the ability to support techniques that try to hide packet loss. The client usually buffers packets for a few seconds before beginning play out.

2.1 Real-Time Streaming Protocol (RTSP)

The *Real-time Streaming Protocol* (RTSP), standardized in RFC 2326 [19], resides at the application layer in the TCP/IP-model and allows the user to control media playback with functions similar to those of DVD-players, such as PLAY, PAUSE, and TEARDOWN (stop). It should be noted that RTSP does not specify how the media should be buffered, compressed, or transported across the network [18].

RTSP is an out-of-band protocol, meaning that it is not part of the stream itself. It is usually carried over TCP, using a default port of 554. An RTSP *presentation* denotes a set of streams belonging together, for instance an audio and video stream [6]. The presentation-concept makes it possible for a client to manipulate several streams with a single request (All streams are assumed to share a common timeline, thus making this behavior desirable). Each presentation has its own URL (rtsp://host.domain/path).

[2] Strictly speaking the media stream is not completely live, since the receiver usually buffers the data for a few seconds before starting the play out. Furthermore, there are generally delays purposely introduced at the sender side to allow editing, censoring, etc.

RTSP has been designed to mimic the structure of the *HyperText Transfer Protocol* (HTTP), which implies that it is a text-based request-reply protocol. Just as HTTP, RTSP-requests also consist of a request line (all lines terminated with <CR><LF>) and then a variable number of header lines, followed by an empty line and an optional message body. The request line has a similar structure to its HTTP counterpart as well, consisting of a method, followed by a resource, and ending with the version of the protocol.

RTSP-responses have the same structure as the requests, the only difference being that they start with a status line. The status line consists of a version string, a status code, and a corresponding reason phrase. Figure 2.1 shows a possible interaction between the client and the server.

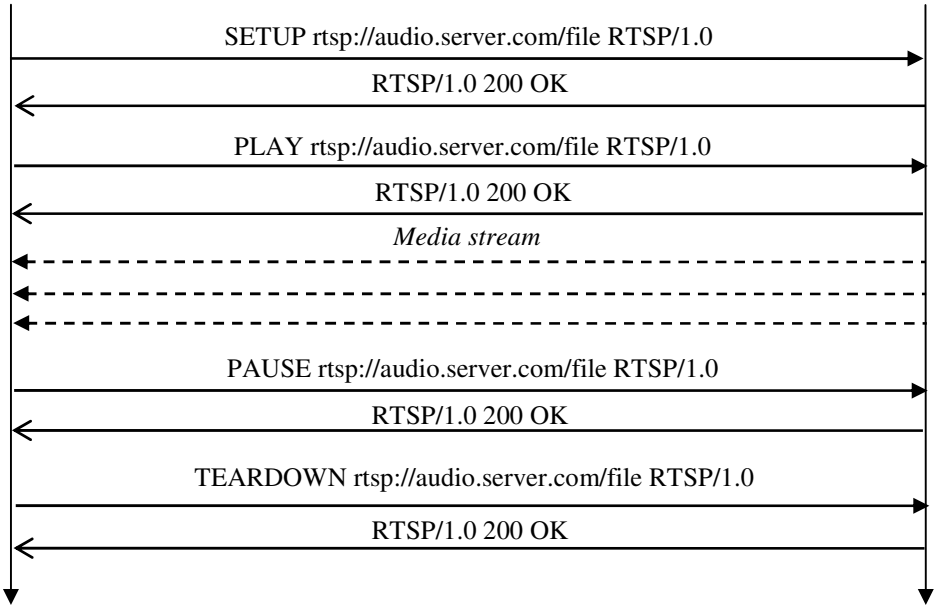

Figure 2.1: An example RTSP session. Only the details of the request/reply lines are shown.

However, there is one important difference between the structures of these two protocols; RTSP is **stateful** while HTTP is stateless. In a stateful protocol, the server must keep state information for each client as long as the connection is open. A stateless protocol does not force the server to save any information between requests. When a client initiates a session, the server assigns it a unique session ID. Each request has a sequence number, which is incremented by one by the client at each request. The next expected sequence number is an example of state that the server must keep.

2.2 Session Description Protocol (SDP)

The *Session Description Protocol* (SDP) is a protocol for exchanging metadata (such as the audio and video CODECs to be used as well as the IP addresses and ports on which to receive and send media) between two parties in a media session. SDP is a proposed IETF standard and is defined in RFC 4566 [20]. Worth noting is that SDP describes *sessions*, not *streams*. A session may contain several media streams.

SDP is a textual protocol, consisting of a tuple of the form <type>=<value>. The type is always a one character string, while the value can be longer and its format depends upon the type. While this certainly limits the number of possible types, they are not intended to be extensible [20]. Should SDP needed be tailored to a certain type of application or media the attribute type (a=) should be used [20], instead of extending the protocol with more types.

In order to simplify parsing and enhance error detection, the RFC 4566 states that type fields must be in a specific order. The type fields can be divided into three categories: the first category describes the session; the second provides information on when and for how long the session is active; and the last describes the media that is carried in the session. Figure 2.2 provides an example of a minimal SDP description.

```
v=0
o=user 3430039441 3430042401 IN IP4 10.20.30.40
s=SDP Example
i=This is an example SDP description
u=http://www.somehost.com/example/sdp.pdf
c=IN IP4 224.20.30.40/127

t=0 0

m=audio 50000 RTP/AVP 0
m=video 60000 RTP/AVP 99
a=rtpmap:99 h263-1998/90000
```

Figure 2.2: Example SDP description

The example describes a session named “SDP Example” originated by “user” at IP-address 10.20.30.40 and sent out on multicast address 224.20.30.40. The session consists of an audio RTP-stream on port 50000 and a video RTP-stream on port 60000. The rtpmap is an RTP-specific attribute that is used to map between RTP-payload types and different media formats. This allows the mapping to be dynamic, which avoids the problem of depletion of payload types [21]. For a more thorough discussion on RTP-payload types, see section 2.3.

SDP only specifies the format of a session description; it does not mandate how this description should be transferred to the receiver. Commonly, a session description is exchanged during session setup with the *Session Initiation Protocol* (SIP) or with RTSP (a client can request a presentation using the method DESCRIBE). For more details regarding SIP and RTSP see [22].

2.3 Real-Time Protocol (RTP)

RFC 3550 [1] defines the *Real-time Transport Protocol* (RTP), which is a protocol designed to carry real-time media content across a network. It is difficult to place RTP in the TCP/IP-layered model, since it performs services associated with the transport layer (such as the use of sequence numbers to enable orderly delivery of packets to higher level protocols), yet it does not provide a complete transport mechanism. Due to this fact, it is useful to think of RTP as a sublayer between the transport and application layer. Formally, RTP is a transport protocol that uses another transport protocol (e.g. UDP).

RTP was designed to only have limited functionality, specifically the minimum common functionality needed for modern multimedia applications [6]. The designers intended the protocol to be extensible by using different *RTP profiles* and *payload formats.* This protocol builds upon two major Internet design philosophies: *application-level framing* and the *end-to-end principle* [21].

The application-level framing philosophy states that only the application itself has sufficient information about its data in order to make a knowledgeable decision about its transportation. This implies that the transport layer should receive data in application-specific chunks called *application data units* (ADUs) and provide feedback regarding delivery of these chunks [22]. Thus, the application can decide how to cope with errors introduced by the lower layers. This is a quite different approach than TCP, which tries to hide the lossy nature of the underlying network by the means of retransmissions. The reason for this is that only the application knows which data it really needs and which data it can operate without (perhaps with some degradation).

There are two ways to build a system that provides reliable communication across a network. One way is to require reliable hop-by-hop delivery (with each intermediate node being responsible for re-transmissions as necessary to ensure correct delivery). The other approach is to accept the fact that the network is unreliable and leave it up to the communicating parties to handle any network-introduced errors.

The latter approach is used on the Internet (either at the transport layer with TCP or at the application layer). The systems along the network path never take responsibility for the data; hence they can be simple and are not required to be robust. They may even throw away data they are unable to deliver, because the endpoints are expected to recover without their help. This is referred to as the *end-to-end principal* and implies that the intelligence is pushed out to the end nodes and the core network is kept relatively dumb. The traditional telephony network uses another model, where the network is intelligent and the end nodes dumb [22].

RTP usually runs on top of UDP, but the specification does not mandate any particular transportation mechanisms. Indeed there exist implementations of RTP over TCP and RTP has even been used on non-IP networks, such as *Asynchronous Transfer Mode* (ATM) networks [22].

RTP uses the term *session* to denote the media streams a group of users are exchanging via RTP. Each participant identifies the session based upon a network address and two pairs of ports, one pair on which data should be sent and one pair on which data is received. The first port of each pair denotes the port on which the real-time media is transported and the second port of each pair is used to convey feedback information regarding the quality of the session. The feedback information is carried by RTP's companion protocol, the *Real-Time Control Protocol* (RTCP), which is described in section 2.4.

RTP does not utilize a dedicated port number, instead the only requirement[3] is that RTP data packets should use an even port number, n, and that the corresponding RTCP packets should use port n+1. In this thesis, the audio and video are not part of the same RTP session (this implies that audio and video are separate streams, i.e. they use separate ports).

[3] In a recent revision of the RTP specification, the requirements on port numbers were relaxed in order to enable RTP packets to more easily traverse *Network Address Translators* (NATs) [21].

2.3.1 RTP packet format

RTP was designed to avoid making any assumptions about the number of participants or which transport mechanism is used. Because of this design decision, the RTP header contains header fields that can be used for synchronizing between several senders and receivers. However, this thesis focuses on unicast content distribution; hence fields related to multicast will only be briefly mentioned.

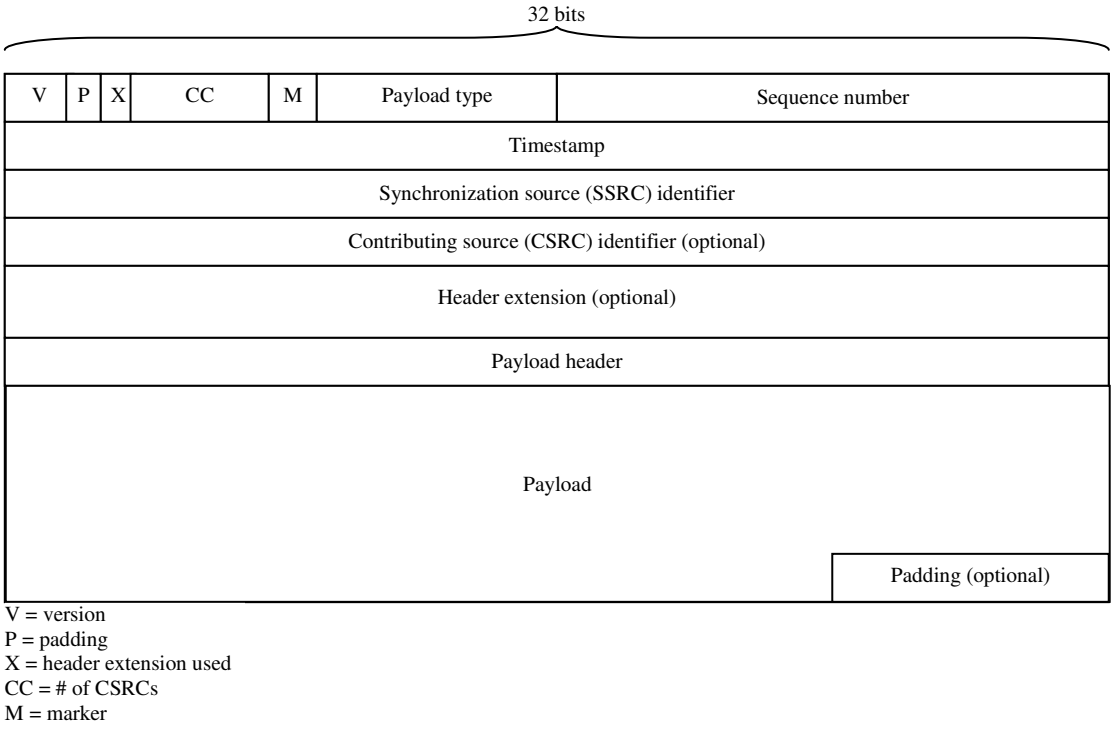

Figure 2.3: RTP packet format

2.3.1.1 Details of the header fields

Version (V) (2 bits). This field is used to indicate which version of RTP is used. The only version currently in use is 2.

Padding (P) (1 bit). This bit is used to flag that the packet has been padded to reach a certain size. The main use of padding is in conjunction with encryption, since some encryption algorithms requires fixed-sized blocks.

Extension (X) (1 bit). Indicates that a header extension field is present.

CSRC count (CC) (4 bits). This field indicates how many CSRC identifier fields the packet contains. For unicast sessions, this number will always be zero (unless mixers are used, see the CSRC identifier field).

Marker (M) (1 bit). This bit is used to mark that this packet has a special meaning related to the profile and media format. The RTP/AVP profile, for instance, uses the marker to indicate the first packet after a period of silence in an audio stream or to mark that this is the last packet containing a certain video frame (as a video frame might be too big to fit in a single RTP-packet).

Payload type (7 bits). This field tells the receiving application the media type that is transported in this packet. The mapping between payload types and media formats can either be statically defined by the RTP profile or it can be assigned dynamically via a signaling mechanism, such as SDP (see section 2.2 for more information on SDP). Section 2.3.2 discusses the most common RTP profile (RTP/AVP) in more detail.

Sequence number (16 bits). The sequence numbers are used to uniquely identify each RTP-packet in a stream. Their main purpose is to detect packet loss and out-of-order delivery introduced by the underlying network. In order to avoid a receiver confusing a new session with an earlier one, which happens to use the same port number; the sequence number should start at a random value. This is also useful to aid encryption algorithms, if RTP encryption is used[4]. Unfortunately, 16 bits is not enough to identify every packet in a long session. Because of this, the participants must keep a wrap-around counter, which is incremented by 1 each time the sequence number wraps around. With the help of this counter, an *extended sequence number* can be calculated:

$$\text{extended sequence number} = \text{sequence number} + 2^{16} * \text{wrap around counter}$$

Formula 2.1: Extended sequence number [1]

Timestamp (32 bits). This field is used so that the receiver can reconstruct the payload's position in the session timeline (i.e., its relative temporal base). The first media sample is assigned a random timestamp and all subsequent packets add a payload-dependent offset to this value. This is needed since RTP is not required to send the packets in "playout order". MPEG video is an example of a media format that utilizes this fact and sends packets out-of-order [21]. Since the sequence number denotes the order in which the packets were sent and **not** the order in which they were sampled, timestamps are needed to reconstruct the stream in correct playout order.

[4] This is needed to avoid a type of a attack known as *known plaintext attack*, when an attacker compares an unencrypted packet with an encrypted one containing the same message. Using a random initial sequence number make this more difficult [21][23].

SSRC identifier (32 bits). This field is used to identify the source of the transmission, referred to as the *synchronization source* (SSRC). The sender chooses this value randomly at the start of each RTP session. Since there can be several senders in an RTP session, the RFC defines an algorithm for resolving collisions.

CSRC identifier (32 bits). The architecture of RTP allows a node called a *mixer*. The role of a mixer is to take several different RTP streams and merge them into one. The mixer will put itself as the SSRC and the source of each RTP stream as a *contributing source* (CSRC). The CC field provides information about how many CSRC fields are present in the header. The topic of mixers is out of the scope of this thesis and will not be discussed further.

Header extension ($\geq$ 32 bits). If the X bit is set to one, this indicates that a header extension is present. The first 16 bits defines the type of extension and the following 16 bits defines the length of the extension in octets. Header extensions are rarely used [21]; hence they will not be discussed further in this thesis.

Payload header (variable). A header specific to the payload type used.

Padding (variable). If the P bit is set, this indicates that the packet has been padded to reach a certain length. This could for instance be used if the RTP packet is being encrypted and the encryption algorithm requires blocks of certain size. The last octet of padding indicates the total number of padding octets.

2.3.2 RTP profiles

The information carried in the basic RTP header is often insufficient for the client to interpret the contents of the packet correctly. This is a deliberate design, since including all the data necessary for all possible media formats would make the header cluttered and waste a lot of bandwidth. Instead, RTP can be extended to include media-dependent information via profiles and a payload format description. The payload format description contains information such as: how to packetize the given media and the organization of the payload data.

Among the information provided in the RTP profile are how profile-dependent fields (such as the marker bit) in the RTP header should be interpreted and guidelines for RTCP usage. By far, the most common RTP profile in use today [6] is the "*RTP profile for Audio and Video Conferences with Minimal Control*" (abbreviated as AVP or RTP/AVP) [24]. RTP/AVP was until recently the only profile in use, but during the last three to four years several new profiles have been proposed, such as the *Audio Visual Profile with Feedback* (RTP/AVPF) [25]. The RTP/AVP profile is described next, while RTP/AVPF is described in section 2.7.1.

RTP Profile for Audio and Video Conferences with Minimal Control (RTP/AVP) is the most common RTP profile and lives up to its name by providing only minimal extensions to ordinary RTP [6]. The profile does little more than provide guidelines regarding audio sampling, slightly

relaxing RTCP timing constraints, and defining a set of default payload type/media format mappings. As described in [21], most payload formats in use require signaling anyway, thus the advantage of using static payload type mappings is lost. As mentioned earlier, using dynamic mapping also avoids the problem of depleting the payload types. Because of this, the IETF Audio/Video Transport working group has now adopted the policy that no additional static assignments will be defined and that mappings should be signaled out-of-band.

2.4 Real-time Transport Control Protocol (RTCP)

The *Real-time Transport Control Protocol* (RTCP) is a companion protocol to RTP and is defined in the same RFC as RTP [1]. The main purpose of RTCP is to provide feedback to the participants in an ongoing session regarding the quality of the session[5]. It is important to note that an RTCP packet is actually not transmitted by itself; instead the RFC mandates that a single UDP datagram carrying RTCP must contain *at least* two RTCP-packets and that the RTCP packets contained in the UDP datagram must appear in a specific order. As a result the UDP datagram contains:

- A 32-bit *encryption prefix*, if and only if encryption is used.
- A mandatory RTCP *Receiver Report* (RR) or *Sender Report* (SR) (if the sender is an active source)
- Additional RTCP Receiver Reports
- A Source Description (SDES) packet containing a CNAME must be included.
- Any additional packet types.

The group of RTCP-packets carried in a UDP datagram is refereed to as a *compound RTCP-packet*. Somewhat confusingly, compound RTCP-packets are sometimes referred to simply as RTCP-packets. A recent IETF Internet draft [26] proposes a relaxation of rules for compound packets, even allowing the transmission of a single RTCP-packet.

The rate at which RTCP-packets are sent varies depending on the number of participants and the media format used. Since there can be many participants in a RTP-session, the specification restricts the RTCP bandwidth to 5% of the total session bandwidth in order to avoid the network being flooded with RTCP-packets. Since Receiver Reports are crucial to the operation of the protocol, they are allocated 3.75% of the session bandwidth (thus only 1.25% of the session bandwidth is available to other RTCP-packets). In order to restrict transmissions further, a minimum interval, which by default is 5 seconds [21], is used.

[5] Additionally, RTCP was designed to provide additional information to participants, such as who the other participants are and how they might be contacted (outside of this RTP session). Many of these other types of information are today provided by other protocols, such as SIP; but they were originally defined in RTP because RTP was designed to be usable by itself (i.e., without a session management protocol), for example in a multicast multimedia session established by other means.

2.4.1 Basic RTCP packet format

RFC 3550 defines five standard types of RTCP-packets:

- *Receiver Reports* (RRs) are used to give the source of the media feedback (such as packet loss and interarrival jitter). At least one RR must be present in each compound packet. Receiver Reports are discussed more in depth in section 2.4.1.1.

- *Sender Reports* (SRs) must be transmitted by participants who recently transmitted RTP data. The main purpose of these reports is to aid the receiver in synchronizing multiple media streams, for instance audio and video. Sender Reports are discussed in section 2.4.1.2.

- A *Source Description* (SDES) is used to convey information about the user to other participants in the session. A SDES-packet consists of a number of SDES items, which provides some information about the sender. The canonical name (CNAME) item is mandatory in each SDES and is used to identify a participant across sessions. The CNAME is often generated from the user name and the network address of the client (e.g. alice@12.23.45.67).

- *RTCP BYE* is sent to notify other participants that the user is leaving the session.

- *RTCP APP* is an application-dependent extension. It consists of a 4 byte field, which should contain a four-character ASCII string that uniquely identifies this extension and is then followed by application-defined data. The main idea behind APP-packets is that they should be used to test new features before a new packet type is registered. An example of an RTCP APP-packet is the PSS NADU APP defined by the *Third Generation Partnership Project* (3GPP). The PSS NADU APP is examined more thoroughly in section 2.7.2.

All RTCP-packets are required to be aligned to a multiple of 32 bits in order to easily manipulate them when building compound packets. The first 32 bits of the header has the same format for all packet types and is depicted in figure 2.3:

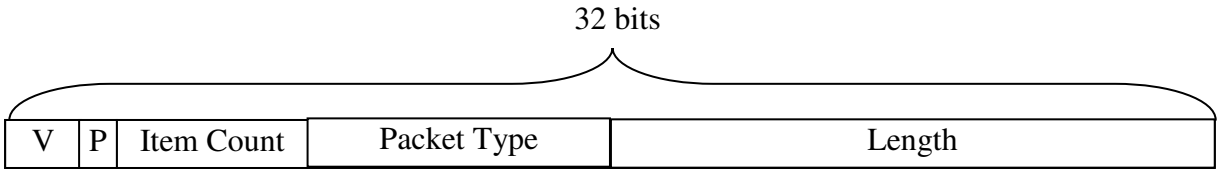

Figure 2.4: Basic RTCP header

Version (V) (2 bits). Just as in RTP, this field is always set to 2.

Padding (P) (1 bit). This field indicates that the packet has been padded.

Item count (IC) (5 bits). RTCP-packets often contain a list of items and this field is used to indicate how many items there are. The different RTCP-packets often renames this field to something more

specific and packets that do not need an item count may use this field for other purposes. The size of five bits limits the number of items to 31, if more items need to be sent they must be split up into two or more RTCP-packets.

Packet type (PT) (8 bits). This field is used to indicate the packet type.

Length (16 bits). This field contains the length of the rest of the packet counted in 32 bits words. Since this field is 16 bits, the maximum length of an RTCP-packet is 65,536 words (2,097,152 bits or 256 kB).

2.4.1.1 RTCP Receiver Reports

If the packet type field is set to the decimal value 201, this means that the rest of the packet should be interpreted as a receiver report. This packet is sent by all participants in the session who receive data and it contains information such as packet loss and delay.

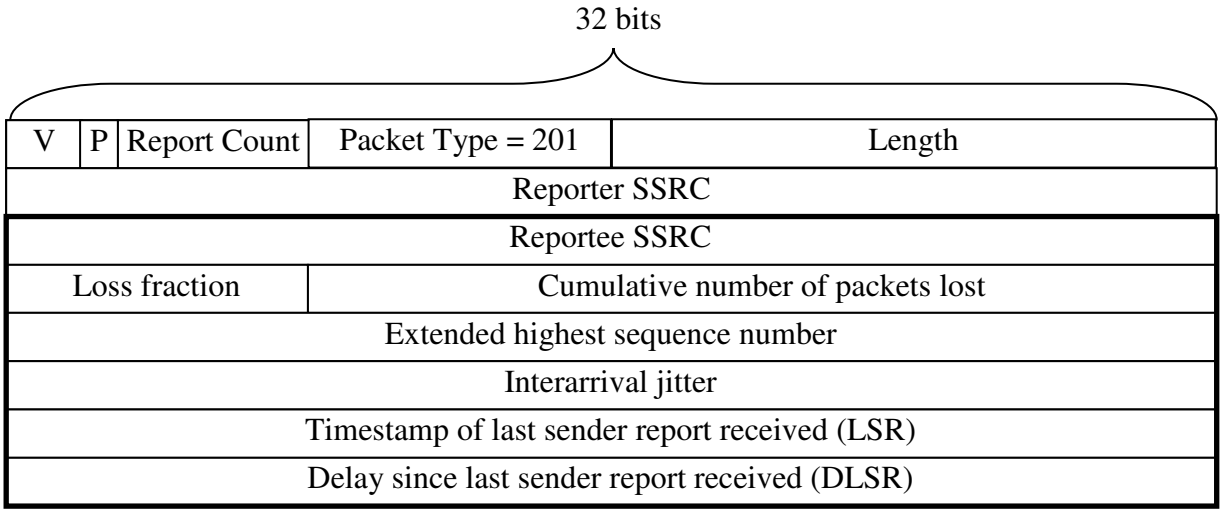

Figure 2.5: RTCP Receiver Report. The bold square indicates the scope of a single report block

Report Count (RC) (5 bits). This field enumerates the number of report blocks contained within this packet. One report block is needed for every source in the current session, but in this report we only deal with the case of a single source[6].

Reporter SSRC (32 bits). This field contains the SSRC of the participant who is transmitting this receiver report.

Reportee SSRC (32 bits). Denotes the source to which the information in this report block refers.

[6] As mentioned earlier, even though both audio and video are sent, the streams are not part of the same *RTP session* and are therefore not carried within the same RTCP-packets.

Loss fraction (8 bits). This field contains the fraction of the packets lost in this interval. Since one can never be completely certain that a packet has been lost and not just delayed, RTCP takes a rather simple approach to calculating the packet loss:

```
packets lost = expected number of packets –
               packets received
```

Formula 2.2: RTCP packet loss [1]

The expected number of packets is the difference between the current value of the highest extended sequence number received and the value at the end of the previous interval. The loss fraction is then calculated by dividing the number of packets lost (i.e. not yet received) with the number of expected packets. Since the underlying network may introduce duplicates of packets the loss fraction may be negative. In that case, the loss fraction field should be set to zero. The loss fraction is bit shifted left eight bits (i.e. multiplied by 256); so the field contains the eight most significant non-integer bits (the implied integer part should always be 0 – since there should be fewer packets lost than sent!).

Cumulative number of packet lost (24 bits).

This field contains the number of packets lost during the *entire session* and is calculated using Formula 2.2. For the cumulative loss, the expected number of packets is defined as the highest extended sequence number received minus the sequence number of the initial RTP-packet. Since the underlying network may introduce duplicate packets, the number of packets received may exceed the number of packets expected [21]. Because of this, the field is signed.

Extended highest sequence number (32 bits).

The extended highest sequence number is calculated using formula 2.1. This field contains the lower 32 bits of highest extended sequence number received during the entire session.

Interarrival jitter (32 bits). This field contains the variance of the estimated transit times of the RTP-packets. The true value of the transit time can not always be calculated, since it requires the sender and receiver to have perfectly synchronized clocks. Due to this fact, the receiver generally must estimate the transit time. The estimation is done by calculating the difference between the value of the receiver's RTP clock and the value of the timestamp field in the RTP packet. If the clocks are not synchronized, then the transit time includes an unknown constant offset[7]. However, because only differences in transit times are

[7] Actually, the offset will most likely not be constant, since the sender and receiver clock most probably have different clock skew. However, since the transit time is only compared between two consecutive packets, the effect of the skew should be negligible.

compared, this offset will not matter. If packet number i is received at RTP time R_i, and contains timestamp S_i, then the relative difference in transit is computed as follows:

$$D(i,j) = (R_j - S_j) - (R_i - S_i)$$

Formula 2.3: Difference in transit times [1]

The interarrival time for each packet is calculated using the difference in relative transit times, D(i,j), for the current packet and the previous packet received. The value that is put in the interarrival jitter field is the current value of the jitter. The jitter is calculated as a moving average using the following formula:

$$J_i = J_{i-1} + (|D(i-1, i)| - J_{i-1})/16$$

Formula 2.4: Interarrrival jitter [1]

Timestamp of last sender report received (LSR) (32 bits).

This field contains the middle 32 bits of the NTP based timestamp received in the last sender report. If no SRs have been received, the field is set to zero.

Delay since last sender report (DLSR) (32 bits).

This field contains the time between receiving the last SR and sending this report, expressed in units of 1/65,536 second.

The version, padding, and length field are used in the same manner as in the basic RTCP header.

As described in [1], the information received in the RTCP RR can be used to estimate the network round-trip time. Upon reception of a Receiver Report, the server subtracts the LSR from the reception time of the RR to learn the time between sending the SR and receiving the report. The difference between this value and DLSR is the estimated round-trip time.

$$RTT = t_{arrival} - LSR - DLSR$$

Formula 2.5: Round-trip time [1]

2.4.1.2 RTCP Sender Reports

All active senders in an RTCP session periodically send sender reports to the other participants. These reports are used to synchronize multiple streams (e.g. audio and video) from the same source. If the sender is also receiving RTP streams, then the SR is followed by RR blocks, enabling the combining of the SR and RR(s) in a single RTCP packet.

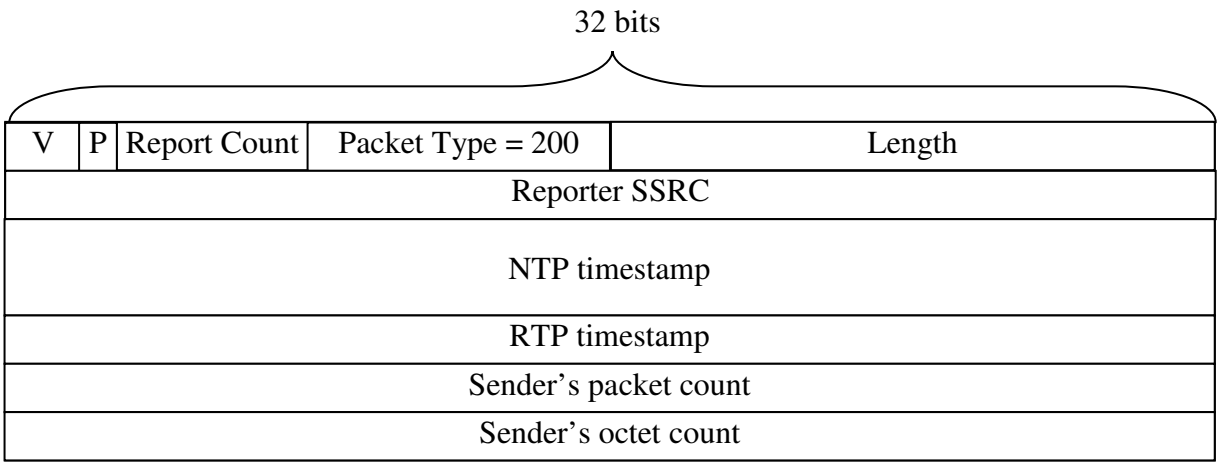

Figure 2.6: RTCP Sender Report.

NTP timestamp (64 bits). This field contains the time when the SR was sent in the *Network Time Protocol* (NTP) format[8], this does however **not** imply that source's clock is actually synchronized with a NTP server. Never the less, this field can be used to synchronize two streams from the same source (even though the source is not actually synchronized with an NTP server), in this case the difference between an absolute time and a (locally) relative time source is not a problem[9].

RTP timestamp (32 bits). This field contains the value of the source's RTP clock at the same instant as the NTP timestamp.

Sender's packet count (32 bits). This field contains the number of RTP packets generated by the source.

Sender's octet count (32 bits). This field contains the total amount of payload sent by the source, measured in octets (i.e., headers and padding are excluded).

2.5 Video compression

As the reader probably knows, a video consists of a sequence of (gradually) changing images (known as frames) displayed at high rate, for example 25 images per second. The human brain will interpret these changes as motion. Each frame is divided into a large number of small cells, known as *pixels*. Each pixel has a distinct colour and is represented by a number of bits.

[8] NTP represents time as a 64 bit number, where the upper 32 bits represents the number of seconds since January 1, 1900 and the lower 32 bits contains the fractions of a second [21]. This means that NTP will have a wrap-around-bug in the year 2036. This representation is the same as used by UNIX, except that UNIX starts counting at 1970 instead of 1900 (However, the UNIX timestamp does not wrap-around at 2106 (2036+70 years), but at 2038 since the UNIX timestamp uses the most significant bit to represent the sign).

[9] Since the timestamps are generated by the same clock, the fact that the clock is inaccurate does not matter because both timestamps have the same offset from the real time (assuming that there was no update to the clock between these two readings of it).

Commonly, a pixel is 24 bits with each colour component (red, green, and blue) represented by 8 bits [27].

A picture represented in this way takes quite a lot of storage space. Consider a picture consisting of 1024x768 pixels. This picture takes up 1024*768*24 bits = 18.9 Mb of raw storage space. A video with this spatial resolution and 24 bits per pixel at 25 frames per second would thus require 471.9 Mb per second. Since few people have an Internet connection of 500 Mbps, it is of course unfeasible to stream raw video. In order to reduce transfer times and reduce storage space requirements, video files are often compressed.

A *CODEC* (coder/decoder) can utilize an algorithm that exploits redundancy in the video to reduce the file's size or, in the case of streaming content, reduce the required data rate. A video sequence exhibits two kinds of redundancy, *spatial* (two adjacent pixels are likely to have the same or similar colour) and *temporal* (a pixel is likely to have the same colour in two consecutive video frames) [27]. Spatial compression may be done on each frame, using similar algorithms as for still images (such as JPEG).

The *Moving Pictures Expert Group* (MPEG) [28] method may be used to temporally compress a video sequence. In this method a video sequence may be compressed into three types of frames: I-frames (intra frames), P-frames (predicted frames), and B-frames (bidirectional frames).

I-frames are independent frames that are not based on any preceding or following frame. They must appear at regular intervals to handle sudden changes in the picture and to avoid loss propagation.

A P-frame is predicted from a previously decoded I- or P-frame, which is called the *reference frame*. A P-frame is divided into blocks of 16x16 pixels, called *macroblocks* and the reference frame is searched to find a similar block. The difference in position between these blocks is encoded as a "motion vector". Since the match between the macroblocks may not always be perfect, there exist some techniques to correct this problem. However, if no suitable block is found in the reference frame, then the macroblock is treated as an I-frame macroblock.

A B-frame is like a P-frame except that it references both preceding and following frames.

It is worth noting that there are additional approaches that can be used for video image sequence compression, one of these is model based coding/decoding. In this approach model parameters are extracted at the source and transmitted to the receiver which uses these model parameters to synthesize a video sequence. This technique has become increasingly popular due to the increasing performance of graphics processors (largely driven by gaming). Additionally, the emergence of physics accelerators allows the local computer to locally model fabric draping, body motion, etc. These techniques have the potential to allow both low data rates, scalable video resolution, and in the case of 3D model - even allow the viewer to choose their own perspective on a scene. However, these modelling techniques lie outside the scope of this thesis as we have assumed a typical cellular phone handset is being used as the client for playout.

2.6 Audio compression

Before audio may be sent over the Internet it must first be digitized (i.e. the analogue input must be converted to a digital signal). This is done by sampling the audio signal periodically. Voice is typically sampled at 8000 samples per second and each sample is 8 bits. The rate of the digital signal will thus be 8000*8 = 64 kbps. Music is commonly sampled at 44,100 samples/second with each sample being 16 bits. This produces a digital signal of 705.6 kbps for mono and 1.411 Mbps for stereo.

Different compression techniques can be selected depending on whether the audio is speech or music. Speech is often encoded using predictive encoding, where only the difference between the prediction and the samples is encoded [27]. Common speech CODECs are *Adaptive Multi-Rate* (AMR) and G.729.

Music, on the other hand, is usually encoded using *perceptual encoding*. The idea behind this encoding mechanism is the fact that the human auditory system is limited causing some sounds to be masked by others [27]. Masking may happen both in frequency, when a loud sound in one frequency masks a softer sound at another frequency (e.g. it is usually impossible to understand speech in night clubs when loud dance music is played), and in time (a loud sound may numb our ears for a period of time after the sound has stopped). This is usually exploited by allocating few bits to sounds which are hard to perceive and more to sounds which are audible. *MPEG-1 Audio Layer 3* (MP3) coding and *Advanced Audio Coding* (AAC) are examples of perceptual coding.

Audio CODECs typically produce streams at a constant bit rate, but may be configured to achieve different rates. AMR, for example, has several different bit rate modes ranging from 4.75 kbps to 12.2 kbps[10]. In this thesis, we do not perform adaption for audio. One of the reasons for this is that audio typically consumes much less bandwidth than video. However, most of the techniques that can be applied for improving the quality of video may also be used for audio.

Interactive speech applications often make use of voice activity detection in order to avoid having to send any traffic when a participant is not speaking. While this may might also be used for streaming audio, we will not consider it in this thesis, since we will focus on streaming audio and video – where it is expected that there is always some audio and video which must be sent.

2.7 3GPP Packet-switched streaming service

The *Third Generation Partnership Project* (3GPP) [2] is a collaboration between numerous telecommunication standards bodies with the responsibility to create technical specifications for 3G networks. 3GPP TS 26.234 [16] is a technical specification on how to provide transparent end-to-end packet-switched streaming. This section will present some of the more interesting features included in this specification.

2.7.1 Extended RTP profile for RTCP-based feedback (RTP/AVPF)

The media source can use information gained via the RTCP RRs to estimate the network conditions under which the client(s) is (are) operating. Unfortunately, the limited transmission

[10] In addition to low rate AMR, there are also *AMR Wide Band* (AMR-WB) CODECs such as G.722.2. These wideband codes are designed for high fidelity applications, such as teleconfencing and entertainment audio.

frequency of RTCP packets, hinders the source from quickly adapting to events at the receiver side [25][29].

To combat this problem, Ott, et al. [25] propose an RTP profile, which allows the receiver to imminently notify the sender of certain events. This proposal introduces a means for the recipients to either acknowledge the reception of an RTP packet (ACK) or notify the sender that a certain packet has not been received (NACK). Furthermore, three RTCP modes are defined. These modes govern the frequency of the receiver reports and what triggers their transmission.

2.7.2 3GPP PSS NADU APP-packet

The *Next Application Data Unit* (NADU)-packet [16] contains information regarding the state of the client's playout buffer. The server can use this information to model the client's buffer and adjust its sending rate to avoid over or under running it. Figure 2.7 and Figure 2.8 show the format of an RTCP APP-packet and the NADU APP-packet respectively.

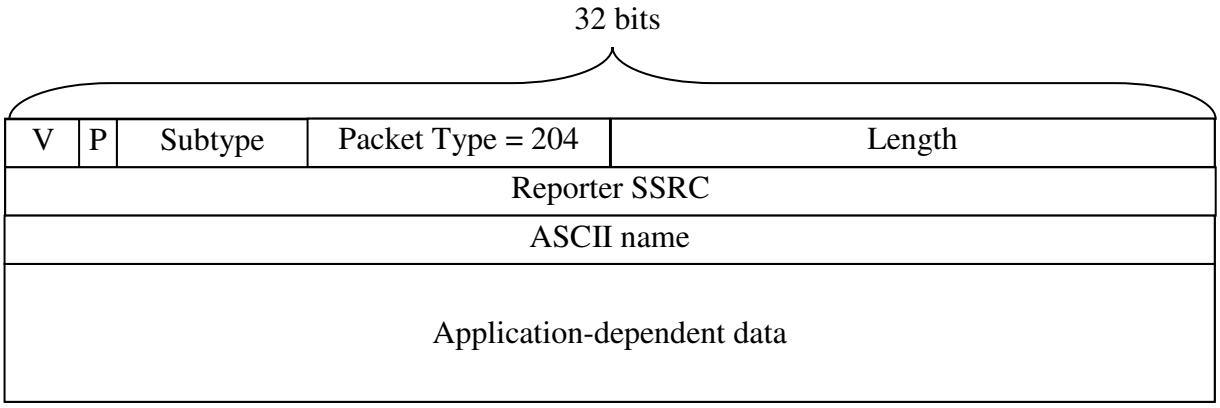

Figure 2.7: RTCP APP-packet

Subtype (5 bits). This field replaces the Item Count field in the generic RTCP header and contains a type value identifying the particular application-dependent extension.

ASCII name (32 bits). This field holds a four-character string uniquely identifying the extension. For the NADU-packet, this is "PSS0".

All other header fields are used as previously described.

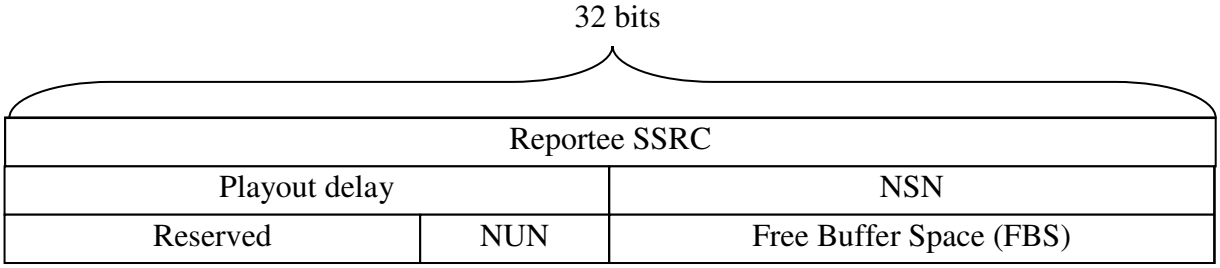

Figure 2.8: The application-dependent part of the 3GPP PSS NADU APP-packet

Reportee SSRC (32 bits).	The SSRC of the stream source.
Playout delay (16 bits).	This field contains the time between the generation of this packet and the scheduled playout time of the next *Application Data Unit* (ADU, see section 2.3) to be decoded. If this period is undefined, for instance because the client buffer is empty, then the reserved value of 0xFFFF shall be used.
NSN (16 bits).	The value of this field is the sequence number of the next ADU to be decoded. If no ADU is scheduled for decoding, then the value of the 16 least significant bits in the extended highest sequence number plus one shall be reported.
NUN (5 bits).	The unit number of the next ADU within the RTP-packet to be decoded. Exactly what constitutes an ADU is left up to the media encoding format.
Free buffer space (16 bits).	The number of complete 64 bit blocks available in the client's playout buffer.
Reserved (11 bits).	These bits are reserved for future use and should always be set to zero.

2.7.3 RTCP Extended Reports

RFC 3611 [30] describes an additional RTCP packet type, called *RTP Control Protocol Extended Reports* (RTCP XR), which provides seven additional RTCP report blocks. The report blocks include a Statistics Summary Report Block (which provides some additional feedback statistics, such as the number of duplicated packets and the TTL values observed at transport layer) and Voice over IP (VoIP) Metrics Report Block (which may be used to monitor the quality of VoIP sessions). RTCP XR-packets are identified by the value 207 in packet type field in the RTCP header.

2.7.4 3GPP PSS Quality of Experience (QoE) metrics

3GPP's *Transparent end-to-end Packet-Switched Streaming* (PSS) [16] specification defines an optional RTSP feature, called *Quality of Experience* (QoE) metrics. This feature allows the client to provide the server with feedback regarding metrics such as time between receiving two error-free frames (corruption duration), the duration of a rebuffering event, and the number of RTP packets lost in succession. For more information about the QoE metrics see [16].

2.8 Streaming over cellular networks

Due to the nature of cellular networks, streaming multimedia content to mobile terminals is a more challenging task than to a wired Internet user. Previously, the available bandwidth in cellular networks has been significantly less than the broadband connections that most people have in their home networks; but, new advances such as HSDPA (discussed in section 3.3.3) considerably reduce this gap. Thus, today the difference in the maximum data rates available to a user via the latest third generation cellular network and the rate that is generally available to fixed internet users is not as significant as previously. Thus both fixed and wireless networks

frequently offer sufficient data rates and capacity to support both interactive and streaming multimedia sessions.

However, wireless users frequently experience large fluctuations in their throughput [31]. The available throughput depends upon several factors, including the radio technology used (e.g. GPRS or WCDMA), the distance from the base station, and the number of users in the cell (and how these users are using the network). Furthermore, when a user moves between wireless cells, temporary network outages may occur due to handovers between the base stations.

Because of these facts, it is important to understand the characteristics of the wireless channels, thus the next chapter will look more deeply into some of the more common *radio access networks* (RANs) in use today.

Chapter 3: Cellular Radio Technologies

Since the specific aim of this thesis project was to investigate if one can reliably detect a change in available bandwidth while streaming video over cellular networks, it is imperative to fully understand the characteristics of the underlying radio technologies. In this section, some of the most commonly used technologies (specifically GPRS/EDGE and WCDMA/HSDPA) are examined. Before presenting the details of these technologies, section 3.1 discusses their basic network architecture.

3.1 The GSM network architecture

Global System for Mobile communications (GSM) is the most common wide area mobile telephony technology. In fact, during the second quarter of 2008, roughly 80% of all wide area mobile subscriptions were GSM subscriptions [32]. Because both GPRS and *Universal Mobile Telecommunications System* (UMTS), a widely used technology for 3G networks, both extend the original GSM core network, a basic understanding of the GSM core network architecture is necessary. Figure 3.1 illustrates the key elements in the GSM network architecture.

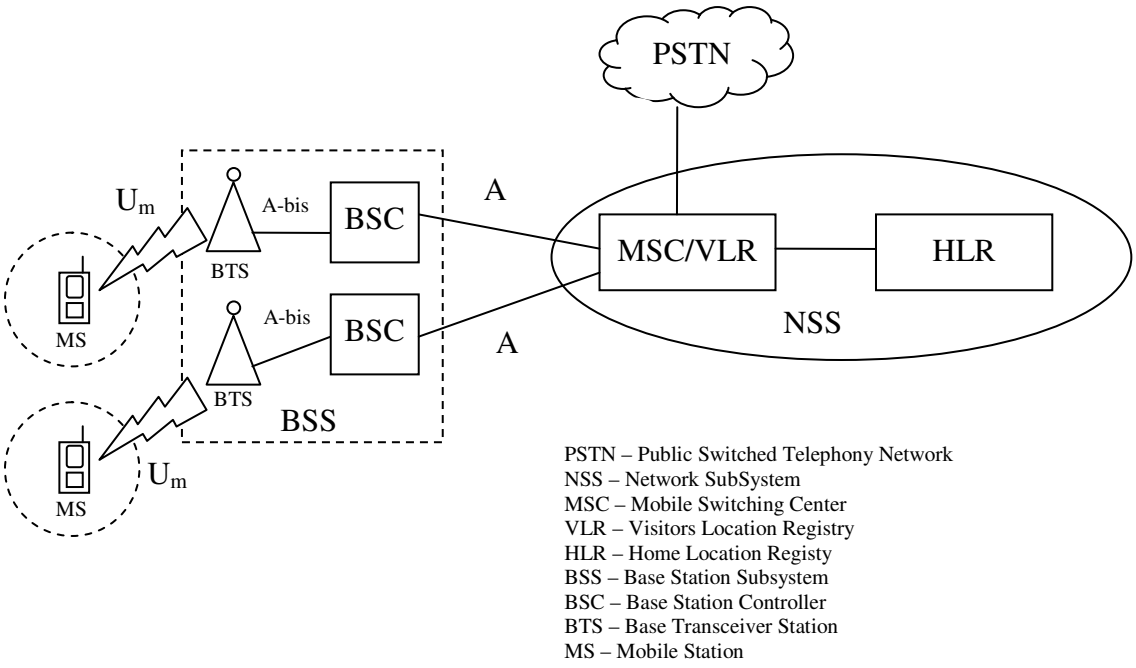

Figure 3.1: GSM Network Architecture

The *Mobile Switching Center* (MSC) connects the core network with wired telephony networks and other circuit-switched cellular networks. The MSC is responsible for the main switching functions of the network. The *Home Location Registry* (HLR) and *Visitors Location Registry* (VLR) are used to keep track of each mobile user, so that calls can be established even if the user is connected to another network. The *Base Transceiver Station* (BTS) contains the hardware and software needed to communicate with the *mobile stations* (MSs) via the air interface, U_m, while the *Base Station Controller* (BSC) handles channel allocation/release and handoff management.

If a BSC is only connected to a single BTS, they are usually co-located (i.e. located in the same place), rendering the A-bis interface unnecessary [34]. GSM utilizes a combination of *Frequency Division Multiple Access* (FDMA) and *Time Division Multiple Access* (TDMA) in order to share the air interface between users within a cell [34]. In Europe, GSM operates in the 900 and 1800 MHz bands [20]. The frequency spectrum allocated to a GSM operator is divided into an uplink band (in the 900 MHz band: 935-960 MHz) and a downlink band (890-915 MHz) [18] (with similar allocations in other bands). The frequency bands are further divided into 124 pairs of duplex channels, each spaced by 200 kHz [18]. The radio channels may be shared using time division multiplexing among users. Each channel has eight *time slots* in a time *frame*, which is approximately 5 ms in duration [22]. Each operator is allocated a range of up and down frequency channels.

3.2 Global Packet Radio Service (GPRS)

Global Packet Radio Service (GPRS) was introduced in 1999 to provide improved packet-switched services for GSM networks [18]. The main goal of GPRS was to better accommodate the bursty traffic patterns of web traffic and file transfers [19]. Unfortunately, the designers of GPRS considered packet based services to be delay **tolerant** and thus gave circuit-switched traffic precedence [35]. This will of course have negative affects on the packet-switched traffic that actually is delay sensitive, such as streaming and interactive multimedia. An advantage of the GPRS design is that it is "always on", which avoids long setup times[1].

GPRS was designed to interoperate with the existing GSM infrastructure, but requires some additional physical nodes as depicted in Figure 3.2. These nodes are each described in subsequent paragraphs.

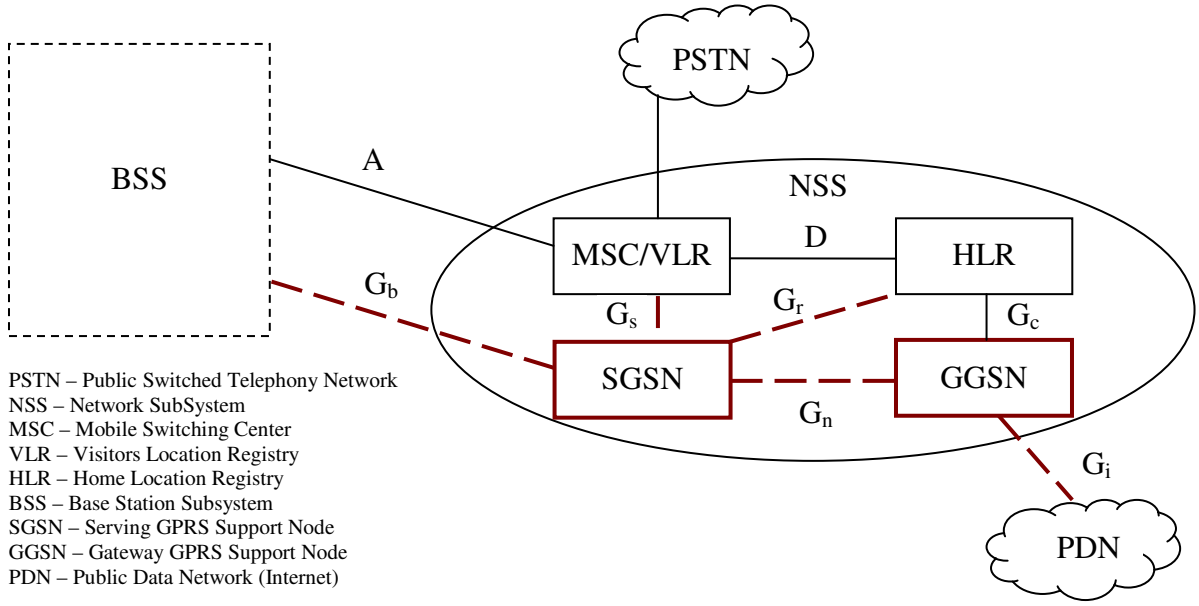

Figure 3.2: GPRS Core Network

[1] When establishing a circuit-switched call in GSM, the setup time can be as long as ten seconds [20].

The *Serving GPRS Support Node* (SGSN) is connected to the BSCs via frame relay links. The SGSNs are used in combination with GPRS tunnels to tunnel the packets through the GSM core network to the appropriate *Gateway GPRS Support Node* (GGSN). The GGSN provides Internet connectivity. The main functions provided by an SGSN are security, mobility management, and session management. As the GGSN is the gateway between the GSM/GPRS network and the Internet, it provides functions such as routing, packet screening, and address mapping (between a possibly private GPRS operator's network address space and public IP addresses assigned to this GGSN) [34]. DHCP and DNS-functions (for use within the operator's GPRS network) are often integrated in the GGSN [34].

Figure 3.3 shows how packets are routed between a server connected to the Internet and a mobile station. Note that the connection is assumed to already be established. It is usually required that the MS initiates all traffic, since the mobile device might not be assigned an IP address until it has data to transmit. The setup procedure for GPRS is described in section 3.2.2.

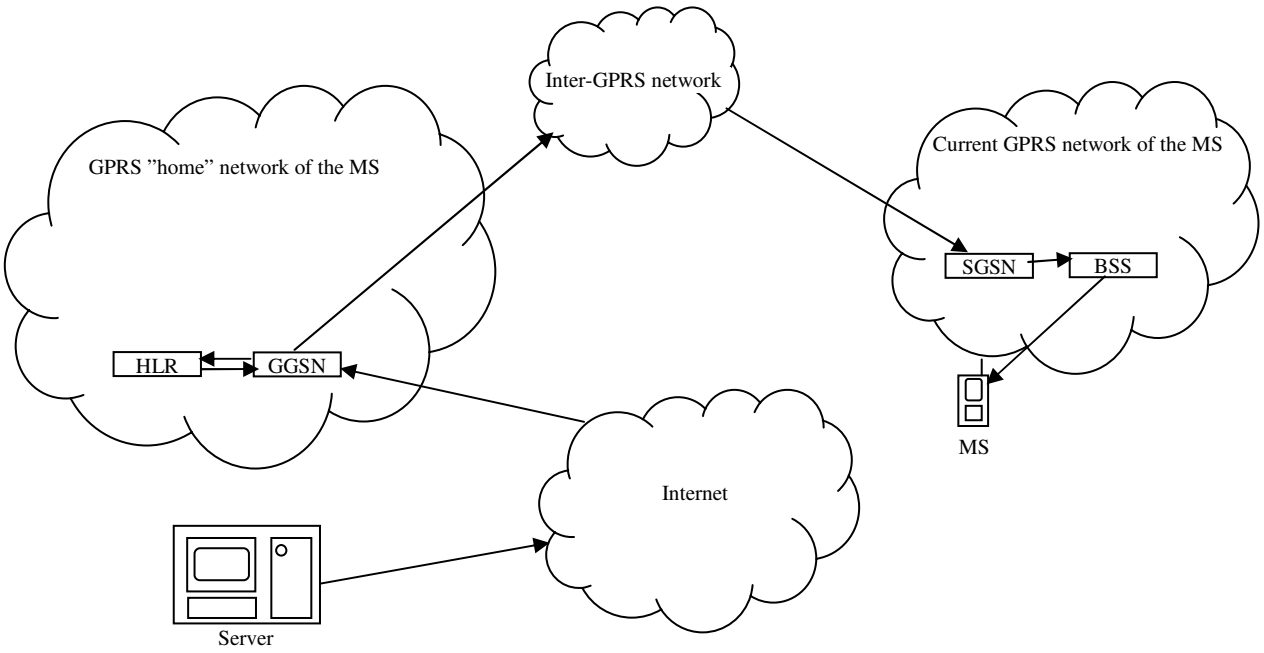

Figure 3.3: GPRS Routing [36]

When the server wants to send data to the MS, it transmits an ordinary IP packet. The mobile stations IP address is in the same subnet as its “home” GGSN, therefore packets will be routed through the “home” GGSN to and from the Internet. The GGSN will consult the Home Location Registry to find out which network the MS is currently attached to and which SGSN is currently responsible for this MS. The packet is then tunneled via the *GPRS Tunneling Protocol* (GTP) from the GGSN to the destination SGSN. The SGSN detunnels the packet and forwards it to the BSS, which forwards it via the BTS to the MS.

Even though they utilize the same core network, older GSM handsets are not capable of GPRS, but need to be upgraded to (replaced with) GSM/GPRS-enabled versions [34]. This is due to the fact that GPRS has some features that GSM lacks. For instance, the GPRS specification mandates

Automatic Re-Transmission Request (ARQ) of corrupted frames, while GSM handsets do not support ARQ. GSM MSs only use a single timeslot. GPRS, on the other hand, allows a mobile station to use several time slots in order to increase throughput [34]. High-Speed Circuit-Switched Data (HSCSD) capable GSM handsets use multiple time slots in order to support higher data rate circuit-switched communication. However, HSCSD devices continuously use the number of channels which they have been allocated - thus such a device can use between 2 and 8 time slots - but no other devices will be able to share these time slots once allocated to the HSCSD device.

The available bandwidth in a GPRS cell depends upon the distance between the MS and the base station (as well as impairments caused by the surrounding environment, such as mountains, buildings, and vehicles). This is due to the fact that GPRS utilizes different coding schemes depending on the received power (as seen at the MS and the Base Transceiver Station) [34]. Initially there were four different coding schemes and their characteristics are described in the Table 3.1. Note that additional coding schemes have subsequently defined; including schemes which are designed for real-time traffic (See section 3.2.4).

Coding Scheme	CS1	CS2	CS3	CS4
Data rate/time slot (kbps)	9.05	13.4	15.6	21.4
Error correction	Highest	Yes	Yes	No
Link budget	135dB	133dB	131dB	128.5dB
Maximum cell range	450m	390m	350m	290m

Table 3.1: Characteristics of GPRS coding schemes [34]

The mobile station chooses the appropriate coding scheme based upon the received power from the BTS [35]. Since the probability of packets being corrupted due to noise increases with decreasing received power from the base station, GPRS tries to combat this by adding additional error correction when lower power is received (which often corresponds to increasing distance). Close to the base station the signal quality is generally very good and because of this no error correction is provided in CS4. However, CS4 still uses interleaving to reduce the effect of burst errors; unfortunately this causes long minimal delays [38]. The theoretical maximum throughput in GPRS is 171.2 kbps (8 time slots * 21.4 kbps/time slot = 171.2 kbps), but in practice a maximum of roughly 50 kbps should be expected, due to a variety of reasons such as radio interference and cell load [37].

3.2.1 GPRS channels

In GPRS a separate physical channel, the *Packet Data Channel* (PDCH), is dedicated to handle packet data [34]. Several different *logical channels* are then mapped on top of this physical channel:

- *Packet Data Traffic Channel* (PDTCH) is the channel used by GPRS to transfer user data [18]. The total system capacity is efficiently used by time slot sharing between multiple users (i.e. TDMA) and, if sufficient resources are available, a single user can simultaneously occupy several PDTCHs.

- *Packet Random Access Channel* (PRACH) is used by the mobile station to initiate data transfer to the BTS or for signaling.

- *Packet Paging Channel* is used by BTS to "wake up" an MS, both for circuit-switched and packet-switched services.

- *Packet Access Grant Channel* (PAGCH) is used in the establishment of a (packet) data transfer to reserve resources.

- *Packet Notification Channel* is used to notify a group of MSs before initiating transmission of multicast packets.

- *Packet Broadcast Control Channel* (PBCCH) is used to broadcast packet-data-specific information to the mobile stations in a cell.

- *Packet Associated Control Channel* (PACCH) conveys signaling information between the BTS and a single MS.

- *Packet Timing Advance Control Channel* (PTACCH) is used to measure and control the *timing advance*. As the physical channels are time shared among the users in the cell, a mobile station quite far away from the BTS might not propagate its data across the medium before the start of the next time slot. If another MS transmits in that time slot the packets will interfere with each other [39]. In order to avoid this, the BTS utilizes this channel to measure the propagation delay, then tells the mobile station how far ahead of its time slot it should start to send data (this is referred to as the timing advance).

3.2.2 GPRS setup procedure

Before a mobile station can use GPRS services, it must first attach to the GRPS network. This is accomplished by registering with a Serving GPRS Support Node. The SGSN performs authentication and authorization functions and assigns the MS a *Packet Temporary Mobile Subscriber Identity* (P-TMSI)[2]. The SGSN also copies the *user profile* from the HLR. The SGSN maintains this profile information about the user until the mobile station performs a detach operation.

In order to communicate with a *Packet Data Network* (PDN), such as the Internet, the mobile station also requires a *Packet Data Protocol* (PDP) address. For the purposes of this report, the PDN is always the Internet, i.e. the PDP is IP. Hence the MS needs to have an IP address (which is allocated by the GGSN). The information needed to identify the GPRS session between the MS, GPRS network and the PDN is called a *PDP context*. This information is stored in the MS, SGSN, and GGSN [36]. It is important to note that a given MS may have multiple PDP contexts in use, since one of the elements in each context is the requested quality of service (see the next section) and each PDP context has only a single quality of service.

[2] A mobile station is uniquely identified by its *International Mobile Subscriber Identity* (IMSI). For security reasons, (to avoid unauthorized tracking of users) the IMSI is not sent over the radio interface [33]. Instead, the MS is assigned a *Temporary Mobile Subscriber Identity* (TMSI), which is unique within the cell, by the VLR at inter-VLR location updates. P-TMSI is the packet-switched equivalent of TMSI.

3.2.3 Quality of service in GPRS

Since not all IP-traffic is best-effort, some limited support to overcome this is provided in GPRS [36]. GPRS Release 1998 defines five *Quality of Service* (QoS) attributes: precedence, delay, reliability, mean throughput, and peak throughput. The value of these different attributes are negotiated by the MS and network when the MS attaches; the attributes are collectively stored as the *QoS profile* within a PDP context. The guaranteed characteristics provided by the QoS attributes can be found in [36]. What a QoS profile for a streaming session might look like is presented in section 5.1.7.

3.2.4 Enhanced Data rates for the GSM Evolution (EDGE)

The data rates provided by GPRS are considerably lower than the rates provided by modern broadband connections. In order to reduce this gap, *Enhanced Data rates for the GSM Evolution* (EDGE) was introduced. Through the use of a new modulation method and more sophisticated modulation and coding schemes (specifically 8-PSK, and later 16QAM and 32QAM), EDGE is able to provide higher throughput than both GPRS and HSCSD. Whereas GPRS had four coding schemes (CS1, CS2, CS3, and CS4), EDGE provides nine *Modulation and Coding Schemes* (MCS) with throughputs ranging from 8.4 to 59.2 kbps [40]. This gives EDGE-enabled handsets a (theoretical) maximum throughput of 8 timeslots * 59.2 kbps = 473.6 kbps, but in practice the throughput achieved is about 128 kbps (with three time slots allocated) or about 100 kbps (with two timeslots allocated) [40]. Another improvement provided by EDGE is that the erroneous frames may be retransmitted using another coding scheme than originally used to transmit the frame.

3.3 Universal Mobile Telecommunications System (UMTS)

Even though GSM and other 2G technologies supported packet-switched traffic, the data rates provided were very low. The data throughput in the initial release of GSM, for instance, was limited to 9.6 kbps [34]. While GPRS improved the performance for packet data traffic, significantly higher throughput than GPRS provided was needed in order to provide a richer set of services. The third generation (3G) of cellular technologies was designed to provide high enough data rates to enable multimedia services such as video telephony.

Universal Mobile Telecommunications System (UMTS) is a 3G system developed by 3GPP [2] UMTS is the largest 3G standard and almost ¾ of all deployed 3G systems are UMTS systems [42]. UMTS can interwork with the GSM/GPRS core network (although, due to the higher data rates provided the core network may require upgrading in order to handle the additional load). The main air interface used in UMTS networks is *Wideband Code Division Multiple Access* (WCDMA) [41]. Figure 3.4 depicts the new radio access network in UMTS (the *UMTS Terrestrial Radio Access Network* (UTRAN)) and how it connects to the GSM/GPRS core network.

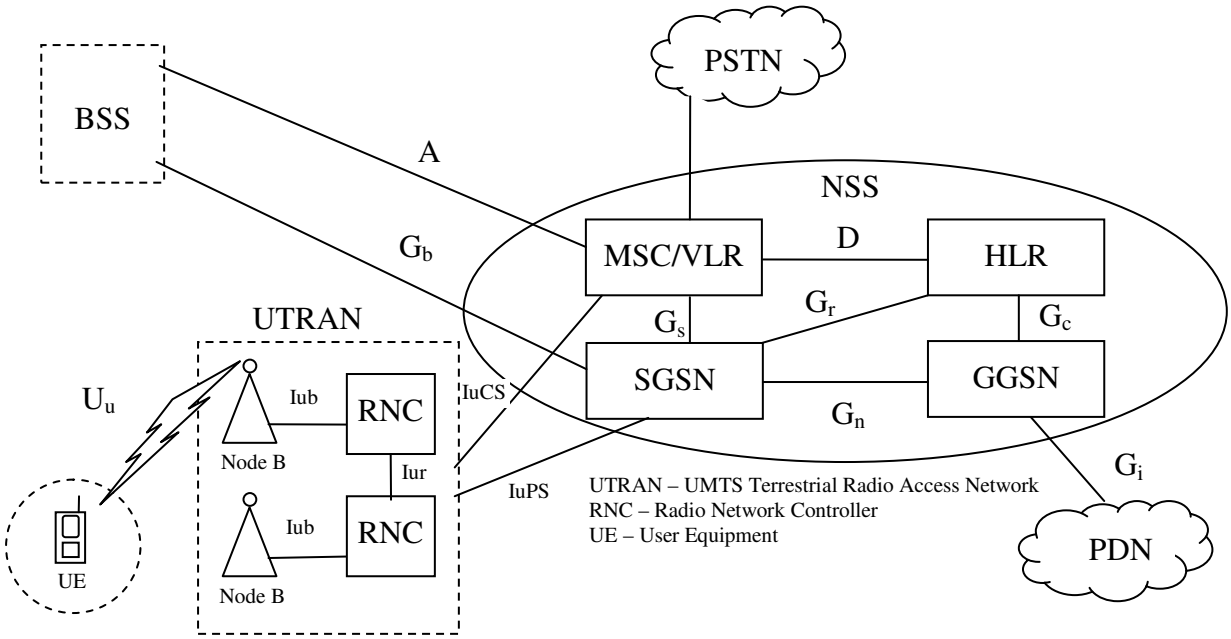

Figure 3.4: GSM/GPRS/UMTS Network [33]

One clear distinction between the GSM BSS and the UTRAN is that the *Radio Network Controllers* (RNCs, the UMTS name for a BSC) may be directly connected through the Iur interface. Whereas in GSM BSCs are not directly connected to each other. Base stations are referred to as Node Bs in UMTS and cell phones, laptop computers, or any other equipment connected to the UMTS network are refereed to as *User Equipment* (UE).

While in GSM/GPRS the air interface was shared by both time and frequency multiplexing, UMTS's air interface (WCDMA) uses *Code Division Multiple Access* (CDMA) to allow simultaneous access by multiple UEs in the same frequency band. In CDMA each sender encodes its signal with a unique code, the so called *spreading-code*. The receiver can decode the received signals by correlation with the spreading-codes. A more detailed description of CDMA in the context WCDMA can be found in [41].

When a user is moving, the handset will eventually move beyond the reach of the base station that it is currently utilizing. In order to maintain connectivity to the cellular network, the user's handset must connect via another base station. If the user is currently engaged in a telephony conversation or data packet transfer, it is not acceptable from a quality of service point of view to simply terminate the connection due to the user's movement. The technique to transfer ongoing connections to another base station is referred to as a *handover* or *handoff*. Though GPRS supports handovers, the user can **not** utilize the network during the handover process, this is referred to as a *hard handover*. WCDMA supports a technique known as *soft handover*. When a user moves into a region with overlapping cells, the UE is able to be connected to multiple cells. This enables the UE to seamlessly switch from one cell to another.

3.3.1 WCDMA channels

WCDMA comes in two variants, *Frequency Division Duplex* (FDD) and *Time Division Duplex* (TDD). The difference between these two variants is how the uplink and downlink are separated, either sending and receiving is done in different frequencies (FDD) or during different time

periods (TDD). WCDMA FDD is the most common implementation [41] and is therefore the implementation that will be discussed here. For simplicity, WCDMA FDD will be denoted WCDMA hereafter.

In Europe, WCDMA operates in the 2GHz band and is allocated 60 MHz for uplink (1920-1980 MHz) and 60 MHz (2110-2170 MHz) for the downlink [25]. The spectrum is split into 12 pairs of 5 MHz duplex bands that can be allocated to different operators. Examples of how the bands can be allocated to different operators can be found on page 4 of [25]. In Finland, for instance, four operators are licensed to operate UMTS systems and has each been allocated three 5 MHz duplex bands.

The physical layer of WCDMA offers a number of *transport channels* to be used by the upper layer protocol. A transport channel defines the *characteristics of the data transfer* and the transport channel is mapped onto physical channels internal to the physical layer. A physical channel in WCDMA is characterized by its frequency and its spreading code. For the purposes of this thesis, the transport channels are quite interesting and will be described in some detail in the following paragraphs.

3.3.1.1 Transport channels

There are two basic classes of transport channels in WCDMA: *dedicated* and *common channels*. A dedicated channel is a link used only for communication between the base station and a particular UE, while common channels are shared among several UEs. There are only a single type of dedicated channel and six types of common channels. Some of these transport channels are described below:

- *Dedicated Channel* (DCH). A dedicated channel is allocated to a single user and carries all information received from the higher layers, i.e. user data as well as higher layer control information. The physical layer does not distinguish between the data it carries, thus both control information and data are carried in the same way [41]. The DCH is mapped onto the *Dedicated Physical Data Channel* (DPDCH) physical channel and some of its features include fast power control, fast data rate change on a frame-by-frame basis, and support for soft handovers.

- A *Forward Access Channel* (FACH) is a downlink channel used to carry control information to UEs in the cell, but may also carry small amounts of data. Multiple FACHs may coexist within a single cell. In the case of multiple FACHs, they usually operate with different data rates in order to reach different parts of the cell [41]. FACH does not support soft handovers.

- *Random Access Channel* (RACH). This channel works in the same manner as the FACH, but in the opposite direction, i.e. the uplink. In order for a system to work properly, the RACH must be able to be heard throughout the entire cell, posing severe data rate restrictions.

- The *Uplink Common Packet Channel* (CPCH) is used to carry uplink packet data. The difference between CPCH and RACH is that a CPCH communication may last several

frames while RACH transmissions are limited to one or two frames. CPCH also supports features such as fast power control and a collision-detection mechanism.

- A *Downlink Shared Channel* (DSCH) is the downlink equivalent of CPCH.

In UMTS, logical channels are mapped onto the transport channels. A logical channel is an abstraction that describes *what type of data* that is being transmitted. The logical channels are divided into two groups: *control* and *data* channels. There are two data channels: *Dedicated Traffic Channel* (DTCH) and *Common Traffic Channel* (CTCH). Both DTCH and CTCH may be mapped to a FACH, DSCH, or DCH channel in the downlink and RACH, CPCH, and DCH in the uplink.

3.3.2 Quality of service in UMTS

The UMTS standard provides richer support for QoS than was present in GPRS. The UMTS QoS mechanism divides data traffic into four classes: *conversational, streaming, interactive,* and *background*, each with distinct characteristics. The main distinguishing property between the different classes is their delay sensitivity. Table 3.2 shows some of the parameters (and their possible values) of the different QoS classes. The QoS class and its parameters are converted to GPRS PDP Context parameters and stored in the QoS profile for the user.

Parameter	Conversational	Streaming	Interactive	Background
Maximum transfer delay	80 ms (lowest possible setting)	250 ms (lowest possible setting)	-	-
Guaranteed bit rate	Up to 2 Mbps	Up to 2 Mbps	-	-
Traffic handling priority	-	-	1,2,3	-
Allocation/retention priority	1,2,3	1,2,3	1,2,3	1,2,3

Table 3.2: UMTS QoS classes and their main parameters [41]

The value given for the maximum transfer delay defines the required 95th percentile of the delay, thus longer delays may occur for individual frames, but statistically the 95% percentile is bounded to the stated value. Each class can be further divided into three allocation/retention levels defining the traffic's priority in the allocation and release of resources. The interactive class also provides three different traffic handling priorities, which are used to determine which packet to drop in case of congestion (with priority 3 packets being dropped first).

The conversational class is intended for person-to-person multimedia communication such as *Voice over IP* (VoIP). This traffic requires low delay and nearly symmetrical up-link and down-link bandwidth. The end-to-end delay is restricted by what is noticeable by the human ear (and eye in case of video telephony users). Tests have shown that more than 400 ms delay results in poor user perception [41].

The streaming class relaxes the delay requirements a bit and is intended for server-to-person real-time traffic, such as video streaming. This would seem to be the most interesting traffic class with regard to this thesis. However, as explained later in this section, it is rarely used in practice. All traffic is instead mapped onto a best-effort bearer. Thus, this QoS class will not be considered further in this thesis after the end of this section.

The interactive class is optimized for request-response traffic such as web traffic, where the response should be delivered within a certain bounded time, such as web browsing. The background class assumes that there are no delay requirements and is suitable for services such as bulk data transfers, e.g. file transfer.

The QoS classes are realized in the Gateway GPRS Support Node using a mechanism (such as) depicted in Figure 3.5. In this figure we can see that the GGSN performs traffic classification and packet scheduling before tunnelling the traffic to the SGSN.

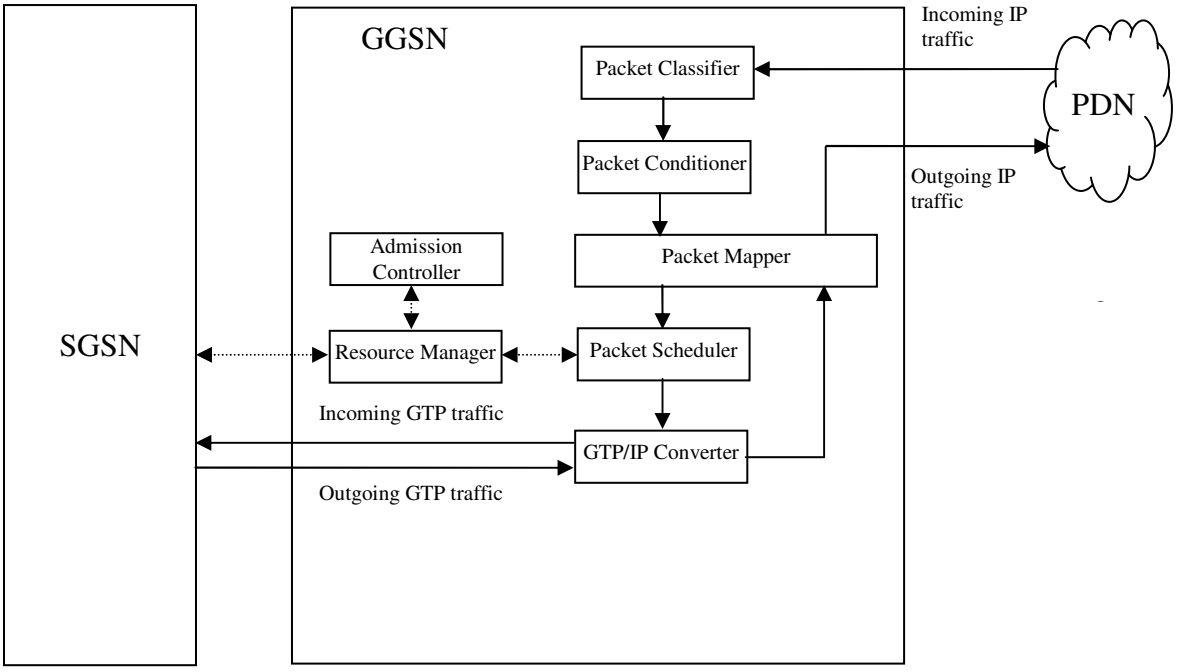

Figure 3.5: GGSN QoS Architecture [33]

The *Packet Classifier* maps incoming data traffic to the corresponding PDP context (see section 3.2.2). If no PDP context is found for a particular packet, it is (silently) dropped.

After the traffic has been mapped to a PDP context, the *Packet Conditioner* makes sure that the traffic for this particular Packet Data Protocol context does not exceed the maximum bit rate allocated for it. If there is sufficient capacity, then the packet is queued for later transmission. If a particular user exceeds his or her maximum bit rate for a period of time, the queue may overflow in which case packets are (silently) dropped.

In the next step the *Packet Mapper* sets the QoS parameters, such as delay and bit error rate for the packet, and passes the packet on to the *Packet Scheduler*. The Packet Scheduler consults the *Resource Manager* to find out which resources are available and gives each packet a delivery priority based on its QoS parameters and the available resources.

In the last step the GTP/IP converter encapsulates the IP packet into a GTP packet and sends it off to the SGSN.

In a UTRAN, a *Radio Access Bearer* (RAB) is set up to provide QoS guarantees. A RAB acts like a pipe with certain characteristics [41][43]. The UTRAN may provide RABs that are matched to the different traffic classes (conversational, streaming, interactive and background) [41].

A streaming RAB will provide the user with a guaranteed bit rate and delay. A streaming RAB will only be set up at the client's request. Unfortunately, according to Ericsson experts, the streaming RAB is rarely implemented (at least not in the Stockholm area). Furthermore, it is not possible to modify the bit rate of a streaming RAB; the guaranteed bit rate negotiated during establishment will be used throughout the RAB's life time. This means that if a streaming bearer is used, then content rate switching is **not** useful, it merely becomes a question of discovering the guaranteed bit rate and keeping the content rate slightly below that rate.

According the Ericsson RAN experts with whom the author has spoken, the interactive RAB is used instead. This makes interactive bearers more interesting for this thesis. The interactive bearer runs on top of a WCDMA dedicated channel and has three different bit rates in the downlink: 384 kbps, 128 kbps, or 64 kbps and one bit rate in the uplink (64 kbps). Which bit rate is used depends upon the user's transmission rate as well as the level of congestion in the cell. If resources are available and the user is sending at a rate above 90% of its allocated bit rate, then the Node B will upswitch this user to a higher rate (assuming, of course, that the user is not already allocated 384 kbps). If the user is not utilizing his Radio Access Bearer sufficiently or if there are too many users in the cell, then the bit rate may be downswitched, figure 10.27 in [41] shows how the bit rate in a DCH may be changed as a new user enters the cell.

3.3.3 High Speed Downlink Packet Access (HSPA)

Release 5 of 3GPP's WCDMA introduces *High Speed Packet Access* (HSPA), which is a concept that provides higher throughput in UMTS networks. HSPA builds upon methods developed for EDGE, which provide higher throughput for GPRS. Currently HSPA achieves a peak data rate of 7.2 Mbps (theoretically). Systems which implement HSPA are often marketed as "mobile broadband" or "Turbo-3G". The peak data rates experienced by users are about 2-3 Mbps [44]. HSPA consists of an improved downlink, HSDPA (*High-Speed Downlink Packet Access)*, and an improved uplink, HSUPA (*High-Speed Uplink Packet Access*). The first wave of HSPA networks usually only support HSDPA and simply 3G in the uplink.

HSDPA introduces a new transport channel: *High-Speed Downlink Shared Channel* (HS-DSCH). In standard WCDMA, the RNC controls the transport channels; but the HS-DSCH is controlled directly by the Node B. This requires more intelligence in the Node B, but reduces the delays due to retransmissions, since errors can be detected earlier and the retransmission performed locally, hence more quickly [41].

The *Transmission Time Interval* (TTI) defines how often the upper layers deliver data to the physical layer. In HSDPA the TTI is decreased to 2 ms, compared to the 10, 20, 40, or 80 ms used in WCDMA [41]. This feature allows HSDPA to more quickly discover variations in user and radio conditions and to allocate resources accordingly [26]. Furthermore, packet scheduling for the HS-DSCH takes place in the Node B. For every TTI, the scheduler decides which user gets to send data and at what speed [45].

3.4 Long-Term Evolution (LTE)

Long-Term Evolution (LTE) is a 4G technology intended to supersede UMTS. LTE is currently undergoing standardization by 3GPP and the initial deployment is expected by 2010 with a larger deployment in the following 1-2 years [46]. The goal of LTE is to increase data rates, decrease latency, and improve user mobility.

While UMTS was designed to inter-operate with the old GSM core network, LTE requires an upgrade of the core network, called *Evolved Packet Core* (EPC). The aim of this is to reduce the number of network components, simplify functionality, and provide handover to other fixed line and wireless technologies [46]. The RAN in LTE is referred to as the *Enhanced UTRAN (E-UTRAN)*.

LTE is designed to minimize the effects of handovers and the interruption time caused by handovers is targeted to be less than a 2G circuit-switched handover [46]. Furthermore, handovers to 2G/3G systems are designed to be seamless. LTE is expected to provide data rates of 100 Mbps in the downlink and 50 Mbps in the uplink, while the latency will be less than 100 ms.

3.4.1 LTE Core Network Architecture

The LTE network architecture is designed to reduce the operators' cost of owning and operating the mobile network. This is achieved by allowing each operator to have a separate core network (CN), while the E-UTRAN is jointly shared by the operators. This is possible due to the fact that enhanced Node B's (eNBs) are able to connect to several CN entities [46].

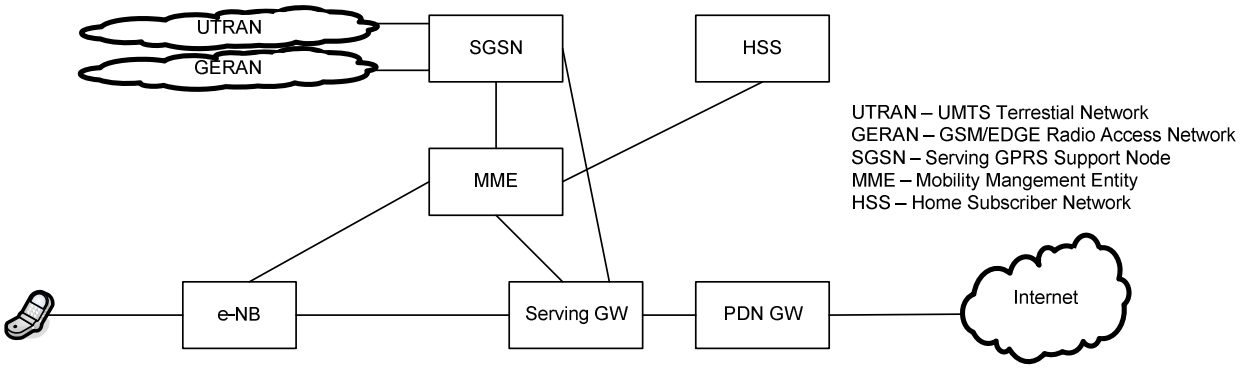

Figure 3.6: LTE Network Architecture

The network components in a LTE network are:

- *Enhanced Node B* (e-NB). In LTE, all of the RNC functionality is moved to this enhanced Node B, which means that the RAN only consists of a single type of node. This enhanced Node B also integrates the SGSN functionality, so there is no longer a separate SGSN.

- *Serving Gateway* (SGW). This node acts as the mobility anchor during handovers, handles paging of idle UE's, and manages and stores the PDP context.

- *Mobility Mangement Entity* (MME). This is the main control node in the LTE core network. It is involved in the bearer activation/deactivation procedure. The MME is responsible for choosing the SGW for each user and performing authentication of UE's by interacting with the HSS.

- *Packet Data Network Gateway (PDN GW)*. This node enables the UE to connect to other packet data networks, such as the Internet. The PDN GW is responsible for security functions such as packet filtering, packet screening, and policy enforcement.

Chapter 4: Ericsson Streaming Server

Ericsson Research has developed a streaming server implementing an adaptive steaming algorithm [46], which uses information from the loss fraction (LF), extended highest sequence number (extHSNR), last sender reports (LSR), and delay since last sender report (DLSR) fields in RTCP receiver reports in order to estimate the state of the network. From these fields, several indicators are derived, e.g. an estimate of the amount of media time currently stored in the network buffers (referred to as the *network buffer media time* (NBMT)). If any of the indicators suggests that the network is unable to sustain the current transmission rate, the bit rate is downswitched. Upswitching is performed under normal conditions on a regular basis. The algorithm is designed to work with little or no knowledge about parameters such as the type of RAN the client is connecting through, the network buffering capacity, the network delay, and the client pre-buffering time.

4.1 The synthetic source

To aid testing, a feature in Ericsson Research's implementation of this streaming server dynamically generates synthetic media. The video consists of a uniformly colored intra picture, followed by inter pictures each coloring an additional macro block until the entire screen has changed color. The color of the macro blocks is dependent on the bit rate, which makes it easy to visually see when bit rate adaption takes place.

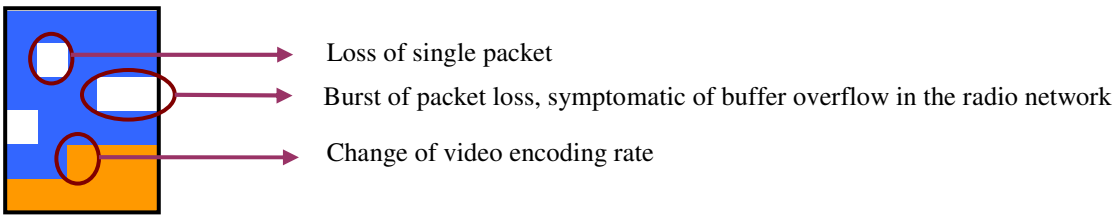

Figure 4.1: Example output of the synthetic source in the client

The synthetic media also includes *Adaptive Multi-Rate* (AMR) encoded audio at 12.2 kbps. The audio is a beep at the start of the picture and after 1/3 and 2/3 of the picture. The synthetic media source provides an API which allows you to set the total session transmission rate, e.g. if instructed to generate media at 32 kbps, it will generate audio and video at rates such that audio bit rate + video bit rate + IP/UDP overhead = 32 kbps (i.e. the total transmission rate is 32 kbps).

4.2 RTSP client

The streaming server also comes with a simple RTSP client, which may be used to test the streaming server. The RTSP client simply dumps the contents of the RTP files to the screen. It is possible to configure the frequency at which the client should send receiver reports.

Chapter 5: Experiments with Streaming Over Different RANs

Chapters 2 and 3 examined the protocols utilized for streaming media and the inner workings of cellular systems. The purpose of this background has been to investigate what characteristics and unique properties the different cellular technologies possess and how they might affect the quality of a streaming video session. How these properties might be conveyed to the streaming server using the feedback information in RTCP has also been studied. We have seen that characteristics such as round-trip time, delay jitter, and packet loss may be measured/calculated from the RTCP Receiver Reports.

This chapter will investigate how streaming video behaves and performs over a wide area cellular network. Measurements will be carried out in both live and emulated cellular networks, in order to quantify the characteristics of the radio technologies. The purpose of these measurements is to learn more about the different RAN technologies and examine how the measured properties are affected by different RANs, different media bit rates, the time of the day (i.e. cell load). These results may, for instance, be used to find appropriate values for algorithm thresholds.

For comparison, Table 5.1 below present data collected in previous measurements (the citation in the table indicates the source of this measurement).

	GPRS	EDGE	WCDMA	HSDPA
Theoretical max throughput	171.2 kbps	473.6 kbps	2 Mbps	7.2 Mbps
Measured maximum throughput	40 kbps [44] 85 kbps [48]	200 kbps [44] 236 kbps [48]	350 kbps [44]	2-3 Mbps [44]
Measured average throughput	36 kbps [40] ~45 kbps [48]	130 kbps [49] ~180 kbps [48]	213 kbps [49]	1 Mbps [45]
Allocation delay	negligible	negligible	DCH: 900 ms [41] FACH: negligible	negligible
Round-trip delay	1000 ms [50] 700 ms [44]	~500 ms [48]	DCH: 180 ms [41] FACH: 250 ms [41]	50 ms [45] 150 ms [44]

Table 5.1: Measured characteristics in mobile networks

The allocation delay refers to the delay incurred by the first packets due to resource reservation in the network. The one-way delay is in general equal to one-half of the round-trip delay.

5.1 Measurements in a live network

5.1.1 Measurement setup

The streaming server, which is currently being developed at Ericsson Research, is installed on a server (an Intel Pentium 4 clocked at 3.40 GHz, 1 GB RAM, running Windows XP SP2) accessible from the Internet. A SonyEricsson K800i was used as the terminal for the first set of measurements. The media player was the built-in K800i player. Both video and audio were generated using the "synthetic source" feature in the streaming server. The synthetic source was described in section 4.2. The measurements were carried out in Ericsson Research's offices in Kista and the mobile phone was connected to Telia's commercial cellular network. Figure 5.1 depicts the experimental setup.

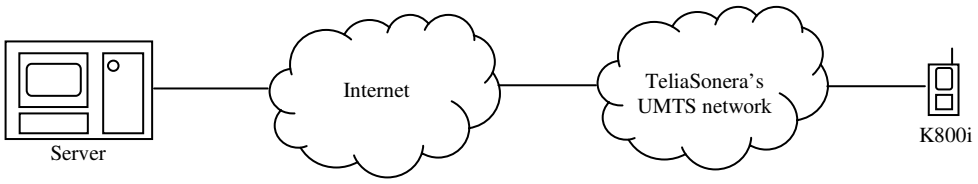

Figure 5.1: Measurement setup. The Internet cloud corresponds to all internetworking elements between the server and the GGSN of TeliaSoneras UMTS network.

Two types of measurements have been carried out:

- *Stability tests.* The performance of the streaming video at different bit rate was evaluated. The video was streamed with constant bit rates during a five minute period. The purpose of this test was to investigate the network behavior at different bit rates. Assuming that the maximum data rate for the network is somewhat fixed, there should be less unused bandwidth when the transmission rate is increased. We were intrested to see if this effect could be noticed.

- *Maximum bit rate test*. In order to see how the network reacts to an increasing transmission rate, the video bit rate was increased in steps until the network was unable to provide sufficient throughput, as detected by increasing RTP packet loss and round-trip delay.

5.1.2 What was measured

The K800i only supports the standard RTP protocol without extensions, such as the NADU APP-packet (however, this was not a concern, since handling of NADU-packets was not yet implemented in the server). Moreover, a streaming server must face the reality of heterogeneous mobile clients, thus clients will have different degrees of support for protocol extensions.

The estimated throughput (ETP) is calculated by looking at the extended highest sequence number in the RTCP RRs. The streaming server records the number of bytes sent at each RTP-packet and the reception time of each RTCP RR. If the value of the extended highest sequence number field of RTCP RR packet x is A and the value of the next RTCP RR packet (x+1) is B, then the throughput can be estimated according to Formula 5.1.

```
data_size = total number of bytes sent between packets with sequence number
            A and B
LF = the loss fraction reported in RR x+1

ETP = (LF * data_size)/(reception_time_{x+1} - reception_time_x)
```

Formula 5.1: Estimated throughput (ETP)

In order to investigate how the streaming session is affected by the different network loads at different times of the day, the measurements were conducted both in the morning, during lunch, and in the afternoon. The values of all fields in RTCP Receiver Reports (as well as the estimated throughput (ETP) and round-trip delay (RTD)) are recorded by the server and are plotted in

graphs at the end of the session. Note that even though the measurements in the following sections are referred to as measurements 1, 2, 3, and so forth, it does not imply that they were performed in this order or that no other measurements were performed in between; thus the numbering is only to more easily refer to them. The measurements presented are selected to illustrate different scenarios encountered during the measurements.

RTD_i, RTD_short, and RTD_long in the graph legends are the instantaneous RTD (RTD_i), and the average delay with a window of three seconds (RTD_short) and seven seconds respectively (RTD_long). Averaging over different time windows is done to illustrate the average behavior versus the instantaneous delay.

Since the transmission rate from the video source is not necessarily constant, the server also calculates the transmission rate. The transmission rate also has natural fluctuations; since not all packets are the same size (e.g. I-frames are larger than P-frames). The transmission rate is calculated every 200 milliseconds with a window of 10 seconds. It is this transmission rate that is plotted in the figures in this chapter. The calculated transmission rate does not include IP/UDP overhead.

For all measurements the audio used 16.08 kbps including IP/UDP overhead and the rest of the configured session transmission rate was used for the video stream and its protocol overhead.

5.1.3 GPRS Measurements

Measurement 1	
Time	2008-07-17 16:00
RAN	GPRS
Total session transmission rate	32 kbps
Average period between receiver reports	1.56 seconds
Duration	5 min

Table 5.2: Measurement 1.

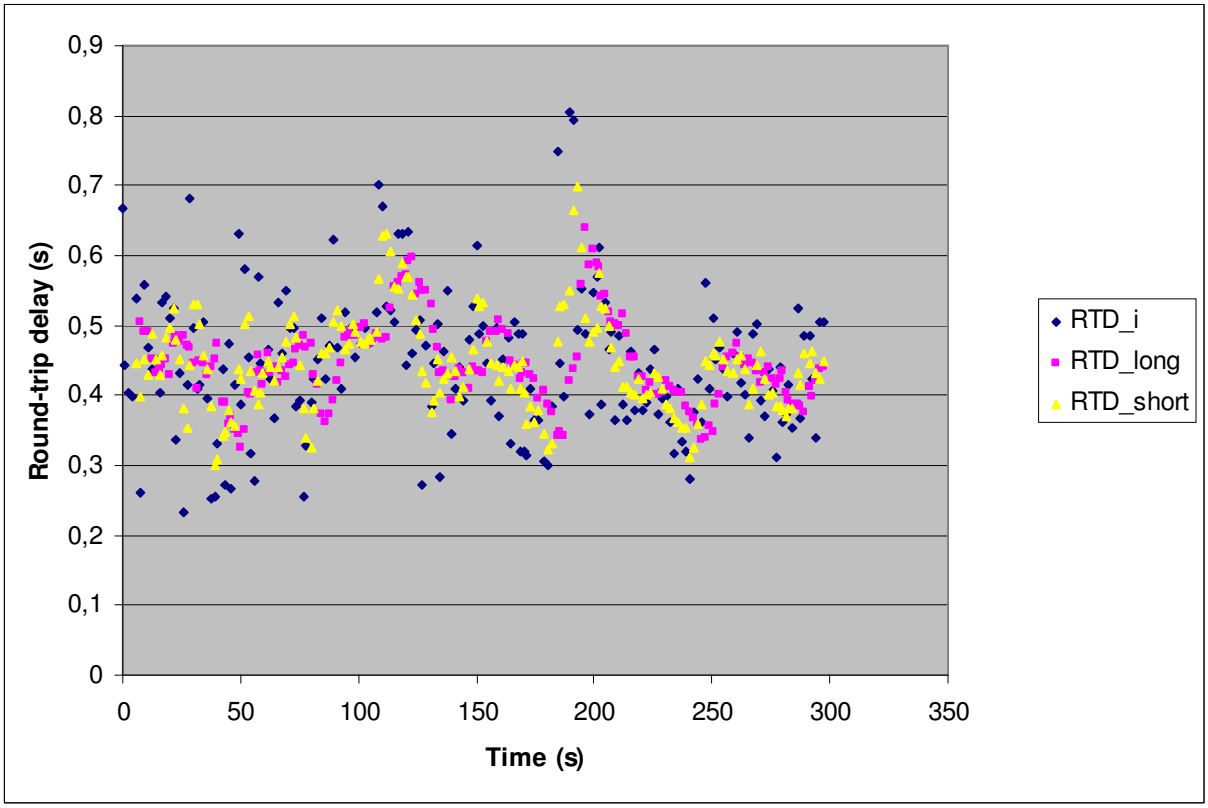

Figure 5.2: Video round-trip delay over during measurement 1.

Figure 5.2 show the round-trip delay for the video stream. As can be seen in the figure, the delay during this measurement varies within the interval 0.25-0.80 seconds with a mean of 0.45 seconds. This is lower than the round-trip times measured in [50] (which where greater than 1 second) even though both measurements used a similar measurement setup. It is difficult to know for certain what these differences are caused by, but one possibility might be a lighter network load during this measurement. The result does, however, correspond well to the delays reported in [44]. Figure 5.3 shows the round-trip delay for the audio stream during the session, which exhibits a similar pattern as the video round-trip delay.

Note that while interesting to calculate and plot, the round-trip delay is largely meaning less in terms of its direct effect upon a streaming session. A key item to note is that the delay is varying, and this delay variance (jitter) has to be hidden by the dejitter buffering. It is readily apparent that with several seconds of initial buffering (hence a dejitter buffer which has a capacity for several seconds of audio/video) that there is not a severe problem in always having content to feed to the local decoder. The second feature to be noted is that this delay variance means that the throughput is changing. This is described in more detail below and in the following figures.

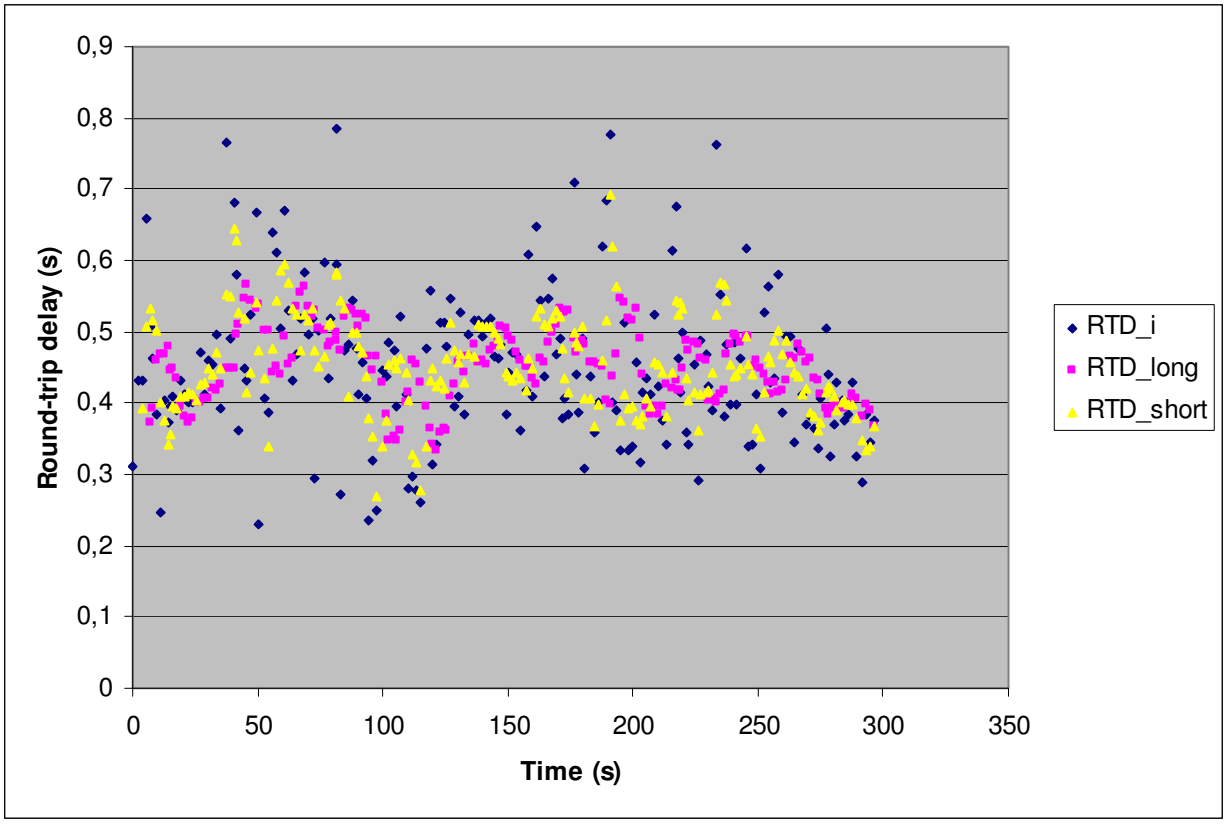

Figure 5.3: Audio round-trip delay during measurement 1.

Figure 5.4 shows the video throughput for the session. Every fourth receiver report an average value of these reports is calculated, this is the avgETP shown in the graphs. Figure 5.5 shows audio throughput during the session. Just as for the round-trip delay, the audio throughput exhibits the same pattern as the video throughput.

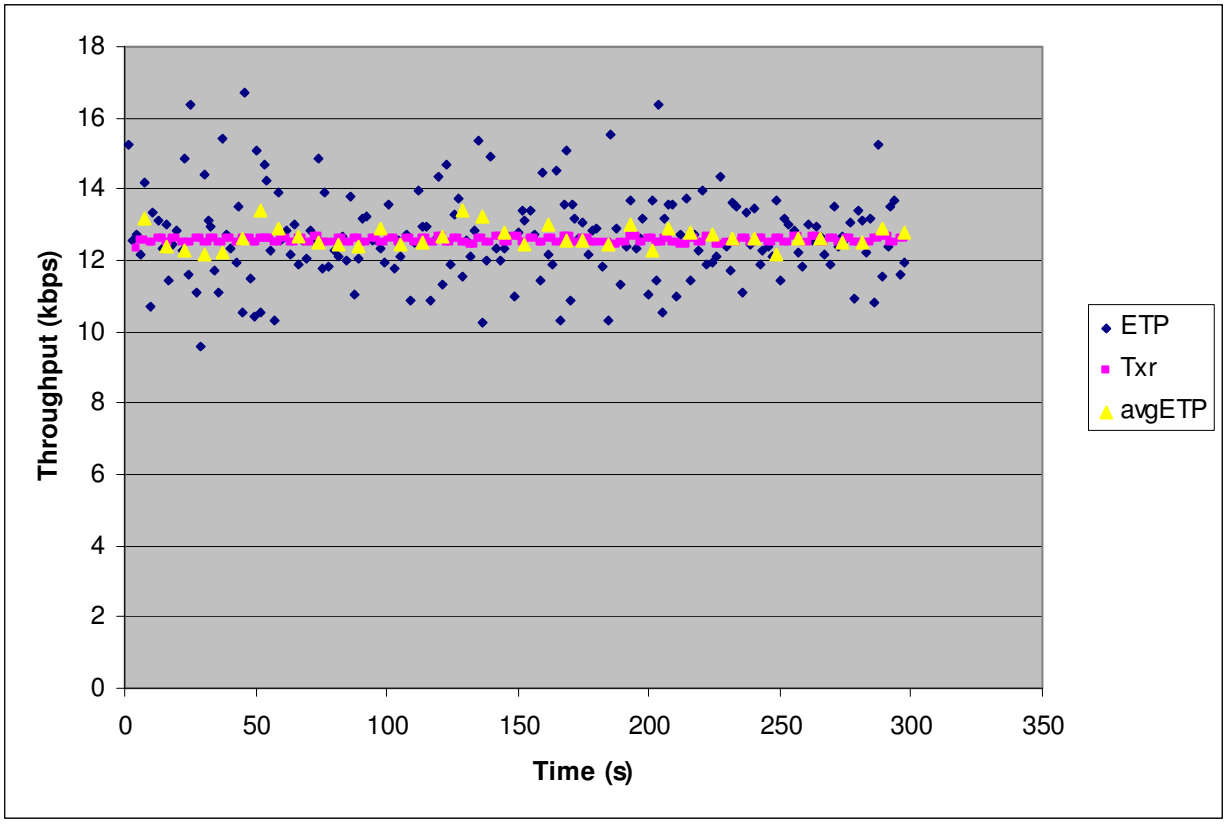

Figure 5.4: Video throughput during measurement 1.

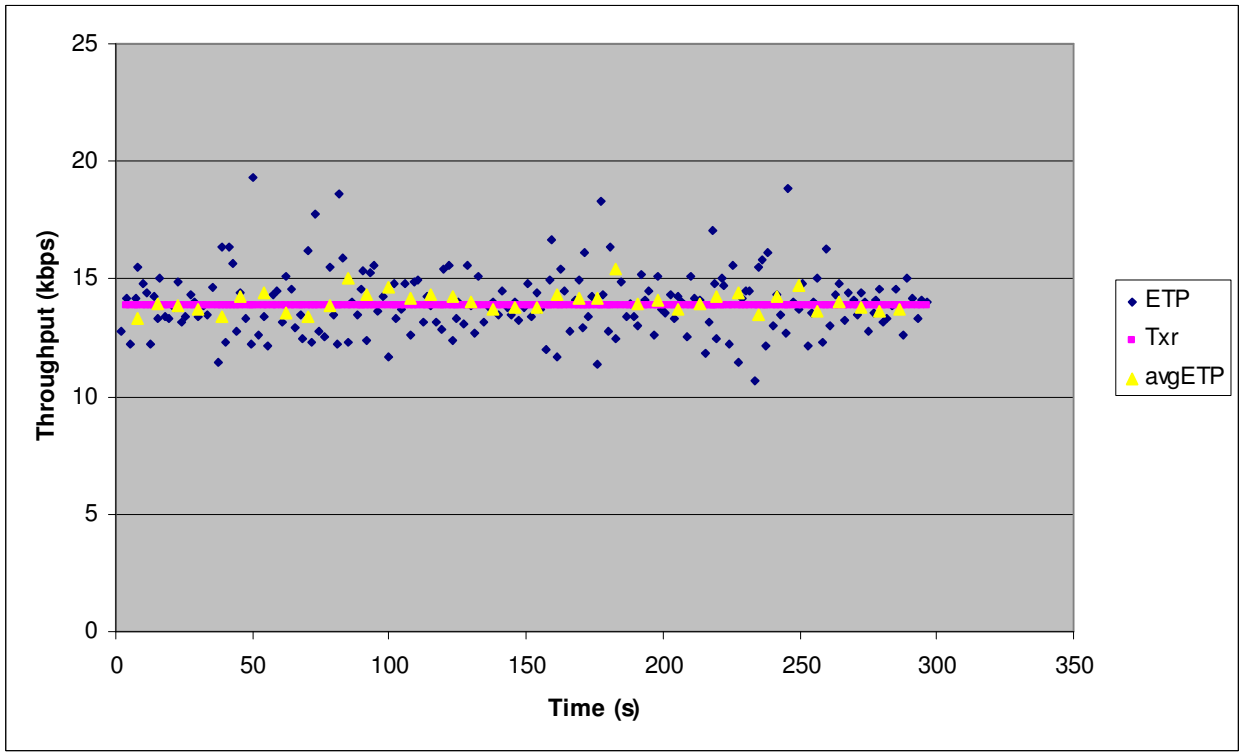

Figure 5.5: Audio throughput during measurement 1.

Measurement 2	
Time	2008-07-18 09:00
RAN	GPRS
Total session transmission rate	32 kbps
Average period between receiver reports	1.56 s
Duration	5 min

Table 5.3: Measurement 2.

Figure 5.6 shows the video round-trip delay during measurement 2. The measurement shows a large peak in round-trip delay after about 200 seconds and again after about 220 seconds. The throughput during these periods, depicted in Figure 5.7, resembles that observed during cell reselection in figure 6.23 of [51]. This might indicate that a cell reselection is taking place. Another possibility might be that a voice call was established in the cell. This will decrease throughput, since voice calls have higher priority in GPRS. This illustrates that the throughput and delay jitter in GPRS are highly variable and unpredictable.

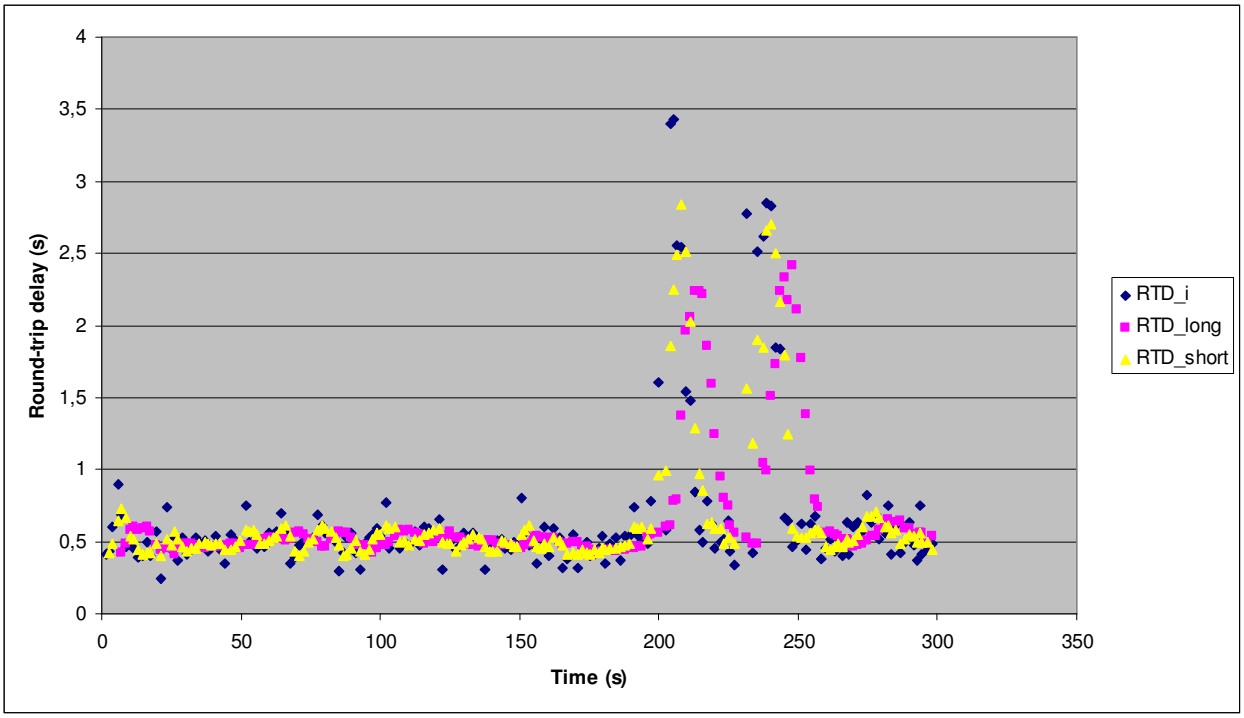

Figure 5.6: Video round-trip delay during measurement 2.

In Figure 5.7, the throughput follows the same pattern as measurement 1, until a sudden drop after 200 seconds and another one after 220 seconds. The drops are followed by a peak in the throughput. A possible reason for this might be that the buffers in the radio network are being filled while a cell reselection takes place and the packets are then released once the connectivity has been restored.

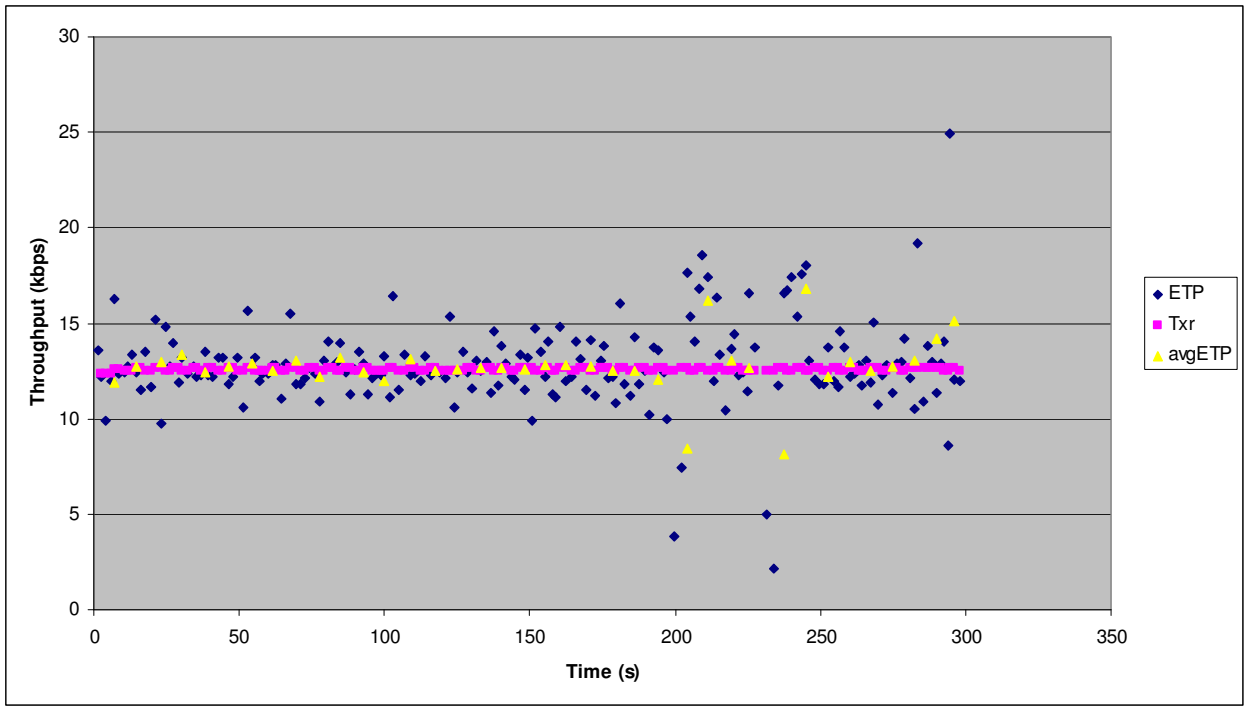

Figure 5.7: Video throughput during measurement 2.

Figure 5.8 and Figure 5.9 shows the occurrence of packet loss during the session for audio and video respectively. The packet loss shown in the figure is the difference between the values in the cumulative loss fields of this receiver report and the previous. At time 235 s, the cumulative loss field in the RTCP RR was 18 (it had previously been 0 throughout the session), indicating that 18 packets were lost during these spikes. In the terminal, these losses could be visually observed by a burst of “black holes” as described in 4.1. We can see in Figure 5.9 that the number of packet losses decreases after the initial reported loss. This indicates that the RTP packets were reordered (see the description of the cumulative loss field in section 2.4.1.1).

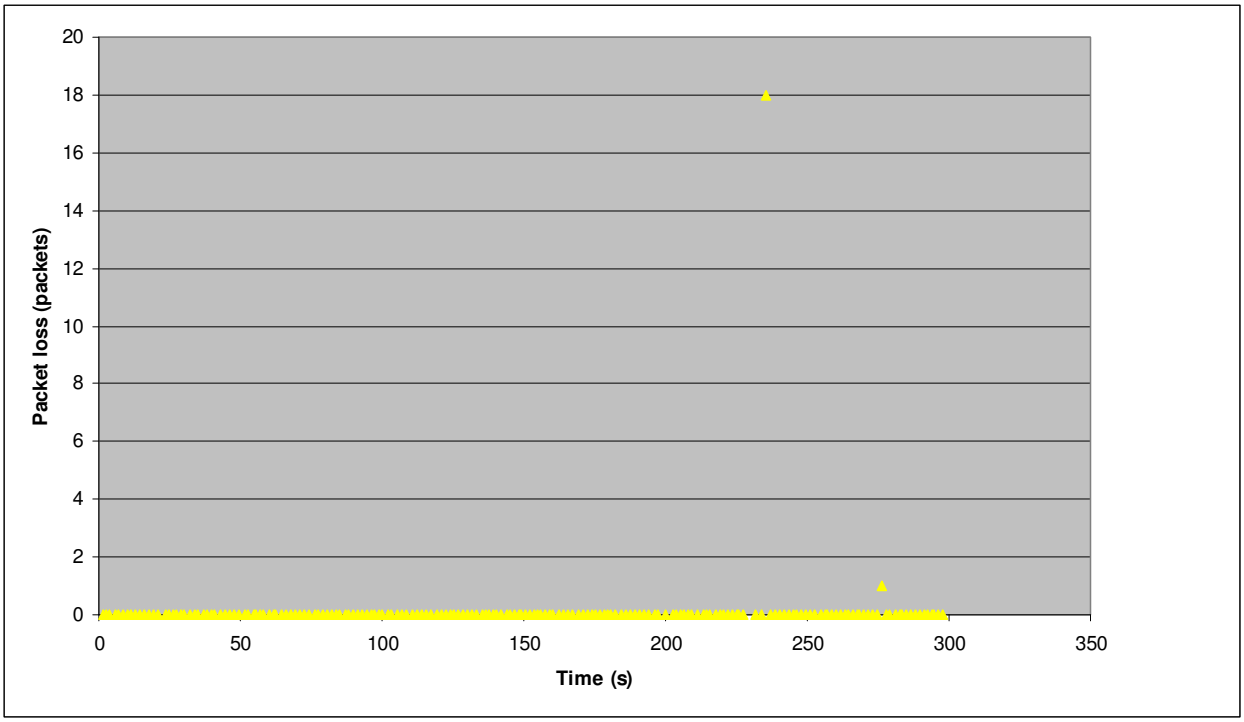

Figure 5.8: Video packet loss during measurement 2

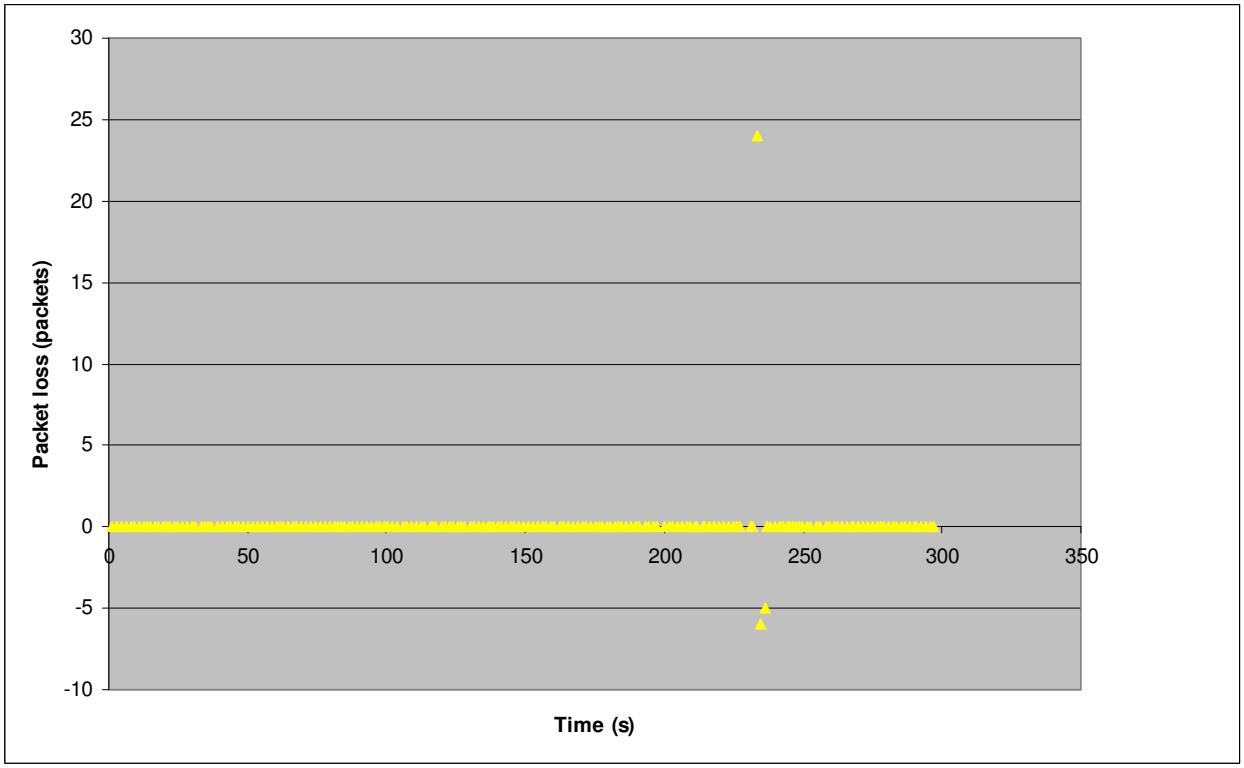

Figure 5.9: Audio packet loss during measurement 2

Measurement 3	
Time	2008-07-18 13:45
RAN	GPRS
Total session transmission rate	28, 32, 44, 50, and 64 kbps (increased once a minute)
Average period between receiver reports	2.02 s
Duration	5 min

Table 5.4: Measurement 3.

Figure 5.10 shows the round-trip delay for one of the maximum bit rate tests. The transmission rate was increased every minute[1]. The transmission rates used were 28 kbps, 32 kbps, 44 kbps, 50 kbps, and finally 64 kbps. The Txr curve shows the video transmission rate as measured by the server and the total transmission rate is indicated in the figure. The red circles and text shows what the total session transmission rate was changed to.

The round-trip delay peaked at 5.8 seconds 230 seconds into the measurement, and then dropped to about 4.5 seconds for the remainder of the session. It is clear that it is not feasible to transmit at 50 kbps or above when the user is connected via GPRS. This corresponds quite well to the numbers shown in Table 5.1.

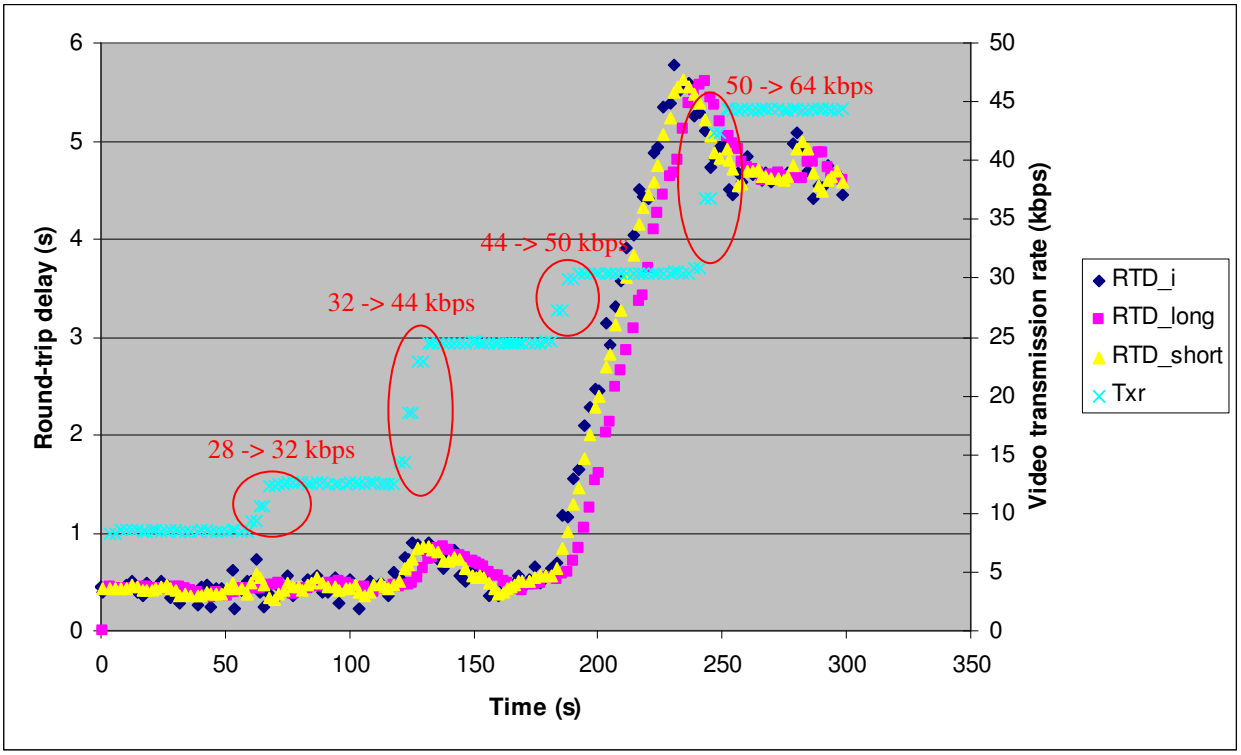

Figure 5.10: Video round-trip delay during measurement 3.

Figure 5.11 shows the video packet loss during the session. After 225 seconds, the first packet losses were reported from the client (a total of 333 packets were lost during the session). The transmission rate, on the other hand, started to increases rapidly after the session bit rate was set

[1] Actually the bit rate (quality) of the video was increased so that the total transmission rate equaled that rate.

to 50 kbps after 180 seconds. This means that it is possible to prevent packet loss due to network buffer overflow by monitoring the round-trip delay.

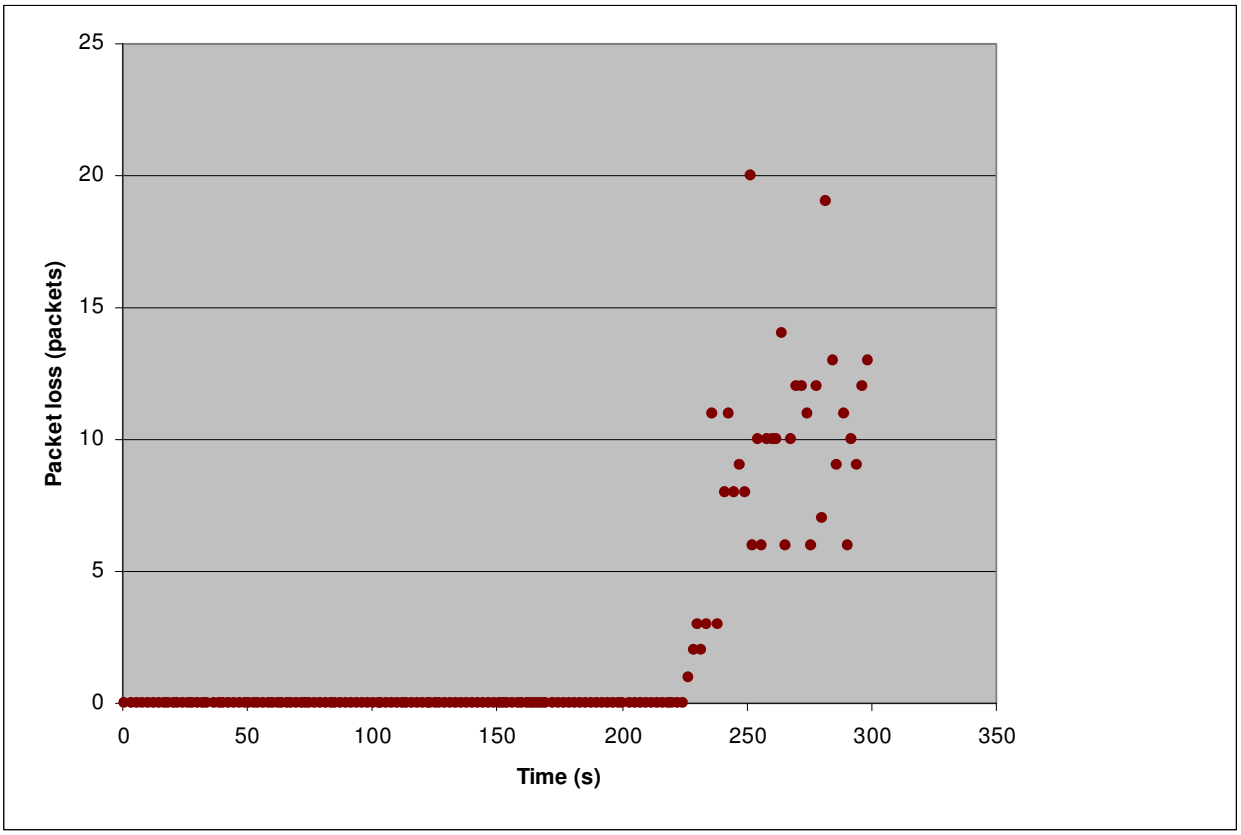

Figure 5.11: Video packets lost during measurement 3

Figure 5.12 shows the video throughput during measurement 3. We can see that there was a rather sharp limit for what throughput the network could deliver at that time. When the total transmission rate was increased to 50 kbps (by increasing the video bit rate to 30 kbps) 180 seconds into the session, the achieved video throughput was about 27 kbps. The estimated audio throughput during the same time period (depicted in Figure 5.13) was about 13 kbps. This indicates that the maximum total throughput achieved for the session was around 40 kbps (excluding lower layer overhead). This corresponds well to the values presented in Table 5.1.

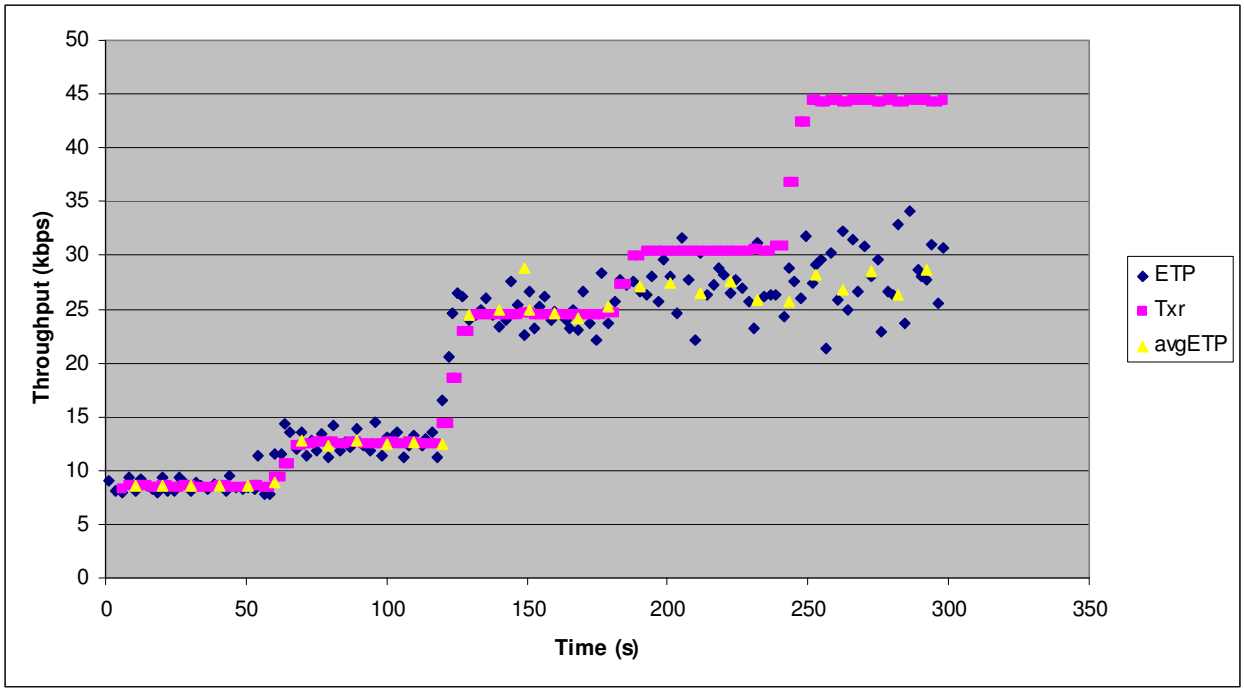

Figure 5.12: Video throughput during measurement 3.

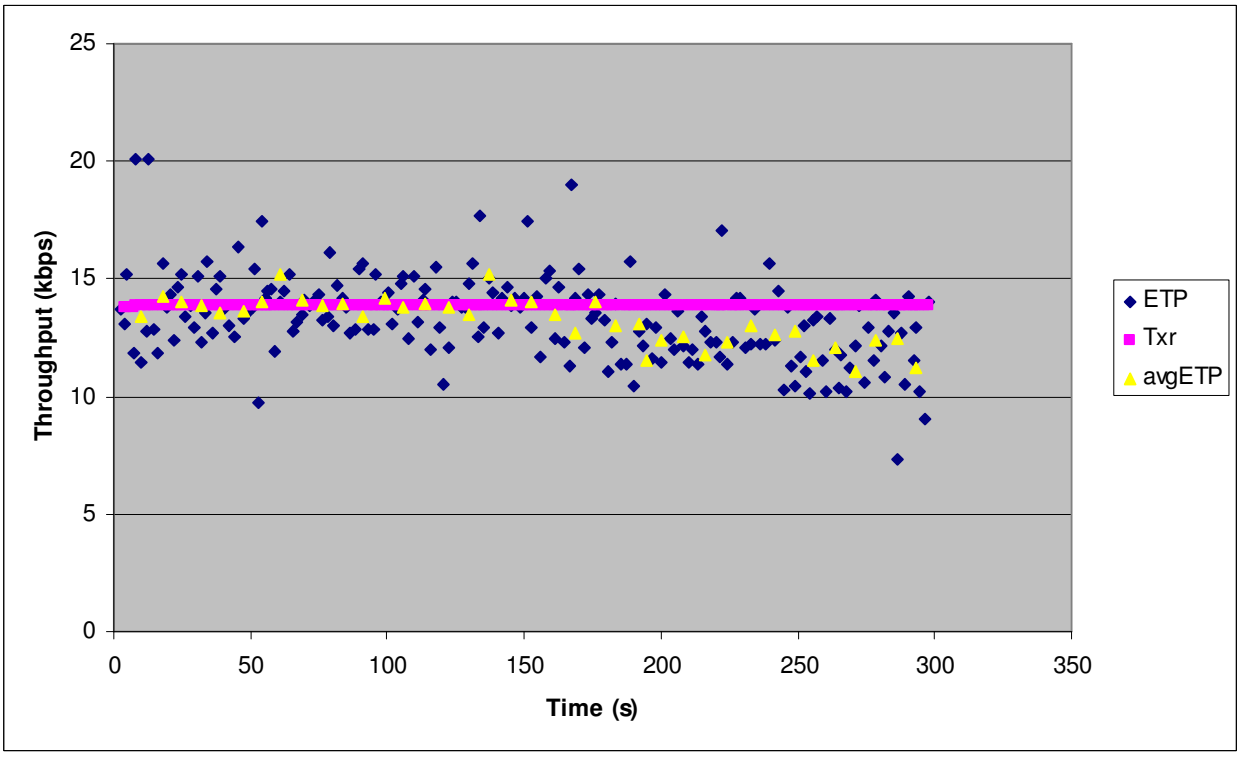

Figure 5.13: Audio throughput during measurement 3

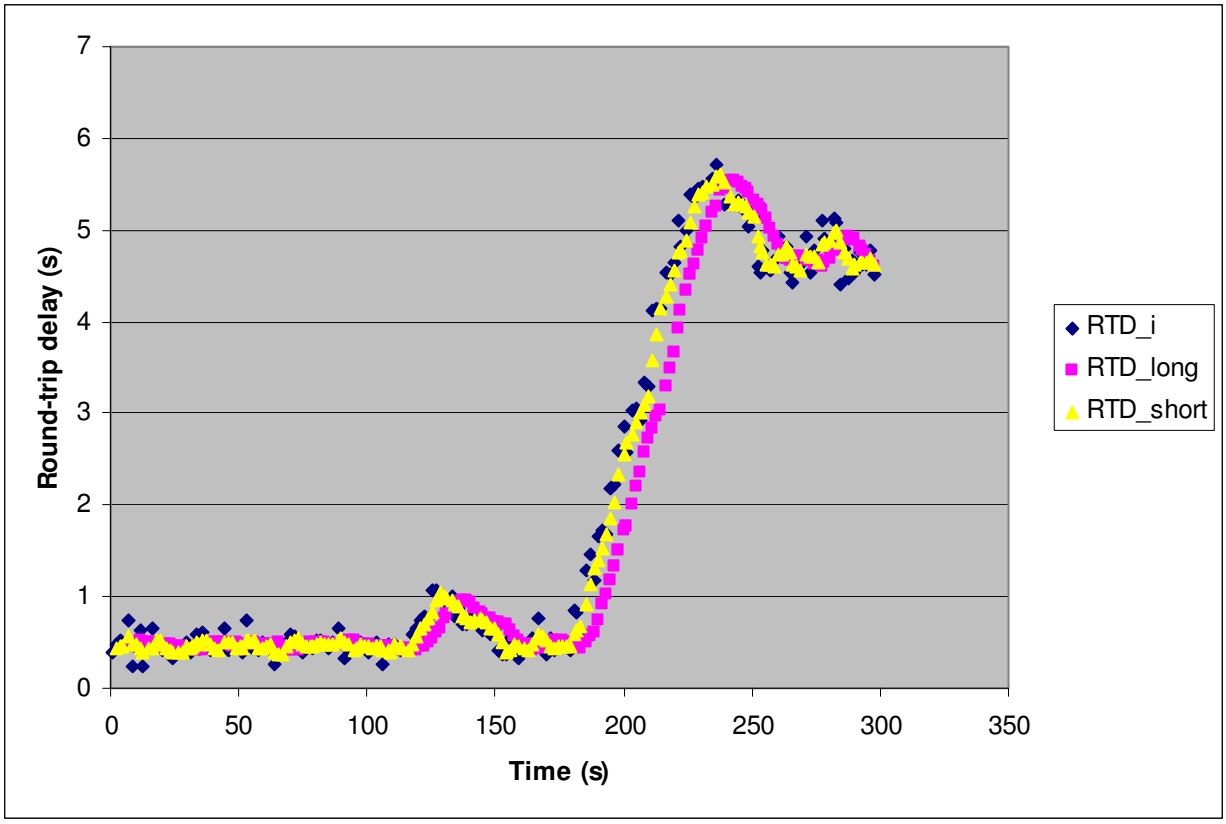

Figure 5.14: Audio round-trip delay during measurement 3.

Figure 5.14 shows the audio throughput during measurement 3. After 180 seconds (the time at which the video bit rate was increased) the throughput starts to sink below the transmission rate.

5.1.3.1 GPRS measurement conclusions

As the measurements provided above indicate, the latency in GPRS networks fluctuates quite a bit, but is usually concealed from the user due to the use of a dejitter buffer. However, sudden large spikes may occur (as shown in Figure 5.6), which may cause buffer underrun in the receiving client. Furthermore, the measurements show that it is possible to detect that the bandwidth is insufficent by observing both the estimated throughput and the estimated round-trip delay. A reaction to insufficient bandwidth can be seen in both the RTD and the ETP well before this causes packet loss. Unfortunately, the reverse is not true; neither the round-trip delay nor the estimated throughput seems to be affected when moving between rates which the network *can provide*. This means that the round-trip delay does not decrease when more bandwidth is available and that the estimated throughput does not have higher peaks as had been expected before the measurements. The main reason for this is that the normal operating mode has network buffers that are close to empty, thus further decreasing the delay is not possible - since the delay is already minimal and not related to the link throughput.

5.1.4 UMTS measurements

During these measurements, the same setup and client as for the GPRS measurements were used; but the K800i was now set to use both 3G and GSM (it had previously been set to use only GSM).

Measurement 4	
Time	2008-07-17 16:30
RAN	UMTS
Total session transmission rate	32 kbps
Average period between receiver reports	1.59 s
Duration	5 min

Table 5.5: Measurement 4.

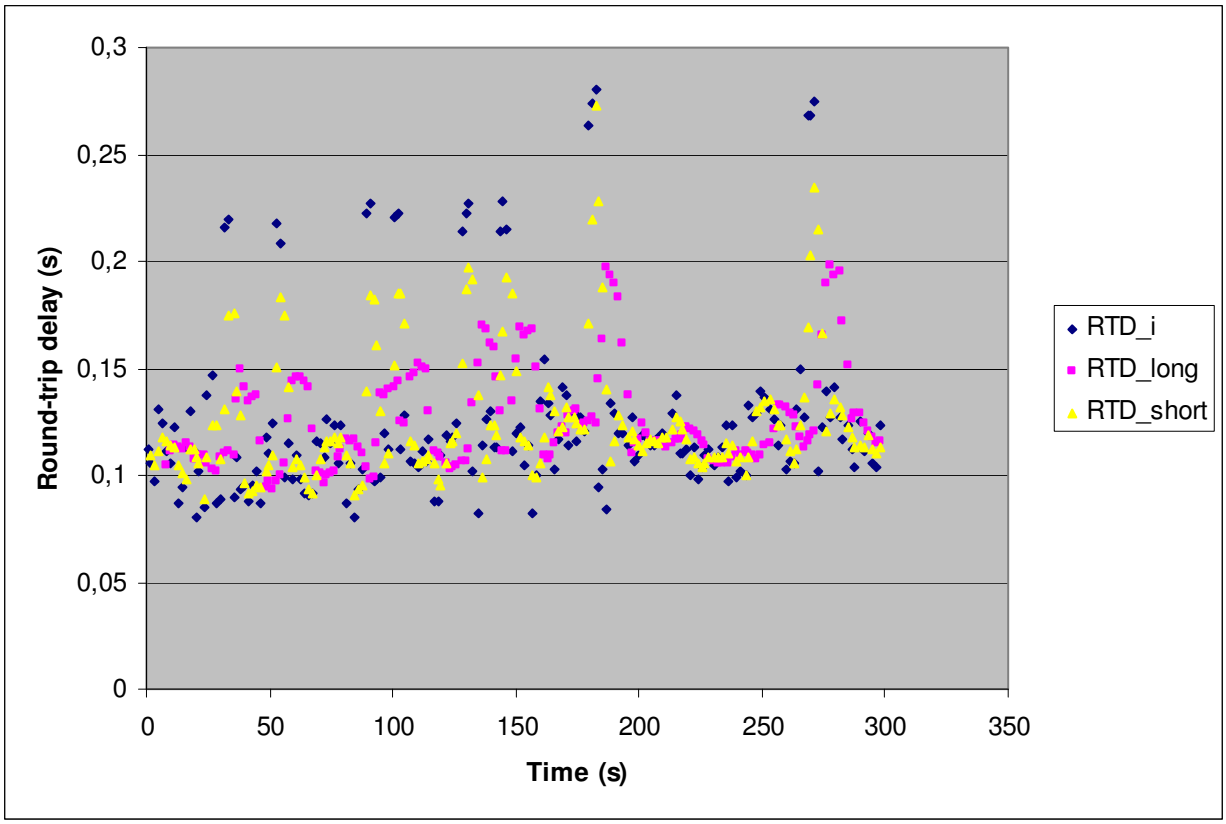

Figure 5.15: Video round-trip delay during measurement 4.

Figure 5.15 shows the round-trip delay during measurement 4. These measurements show that the round-trip delays are significantly shorter for UMTS than for GPRS. The peak delays in this measurement, are slightly above 250 ms, which is roughly comparable to the *lowest* delays seen over GPRS. The large delay spikes are most probably due to link-layer retransmissions, but contrary to what the reader might think at a first glance at these graphs, it is not due to bursts of errors that consecutive packets are delayed. Remember that the round-trip delay is only an estimate and is the sum of the downlink delay for the SR and the uplink delay for the RR. The fact that several delay spikes occur in a row is due to the fact that the receiver reports are sent more frequently than sender reports, it simply indicates that a sender report has been retransmitted, hence the receiver reports based on it all show increased round-trip delay. This is

more evident in the following measurements when the transmission rate is increased, which causes the K800i to transmit RRs more frequently. During this session, a total of three RTP packets were reported as lost in the RTCP receiver reports.

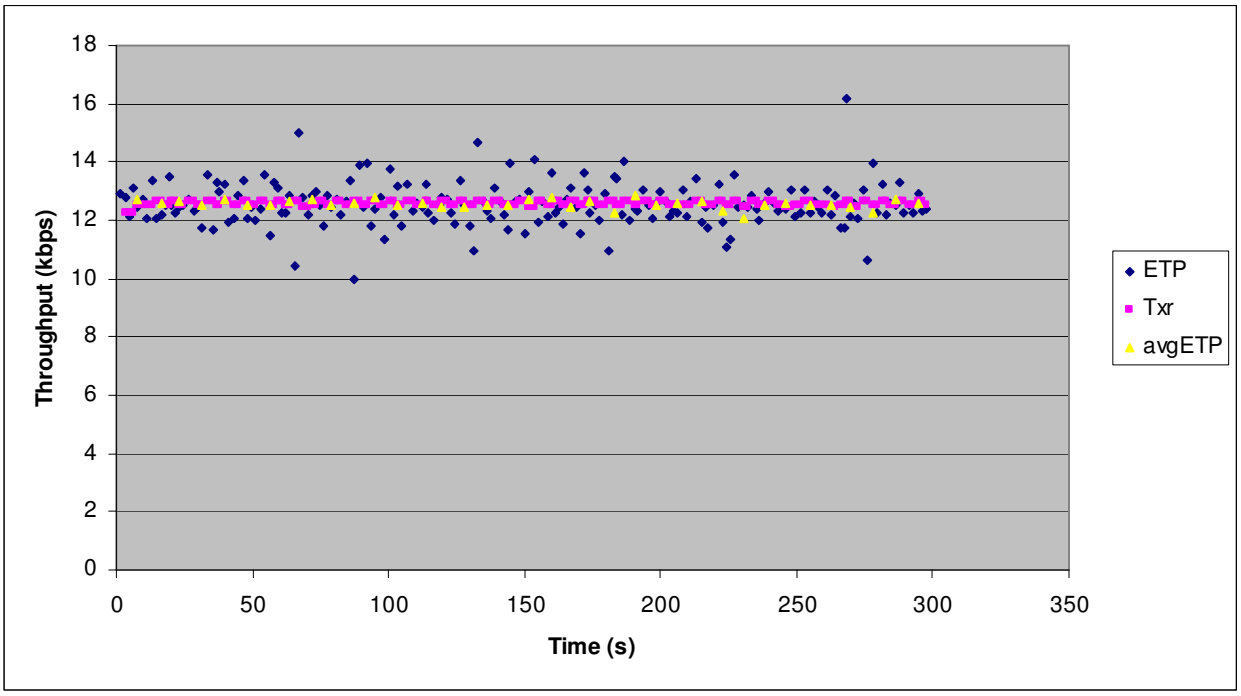

Figure 5.16: Video throughput during measurement 4.

Figure 5.16 shows the estimated throughput during this measurement. As can be seen, the throughput is quite stable, with no noticeable high peaks or large drops in the throughput. This is of course expected, as 32 kbps is considerably lower than the 64 kbps typically allocated to an active user. Both the round-trip delay and estimated throughput for the audio followed the same basic pattern as for the video.

Measurement 5	
Time	2008-07-17 11:20
RAN	UMTS
Total session transmission rate	64 kbps
Average period between receiver reports	0.51 s
Duration	5 min

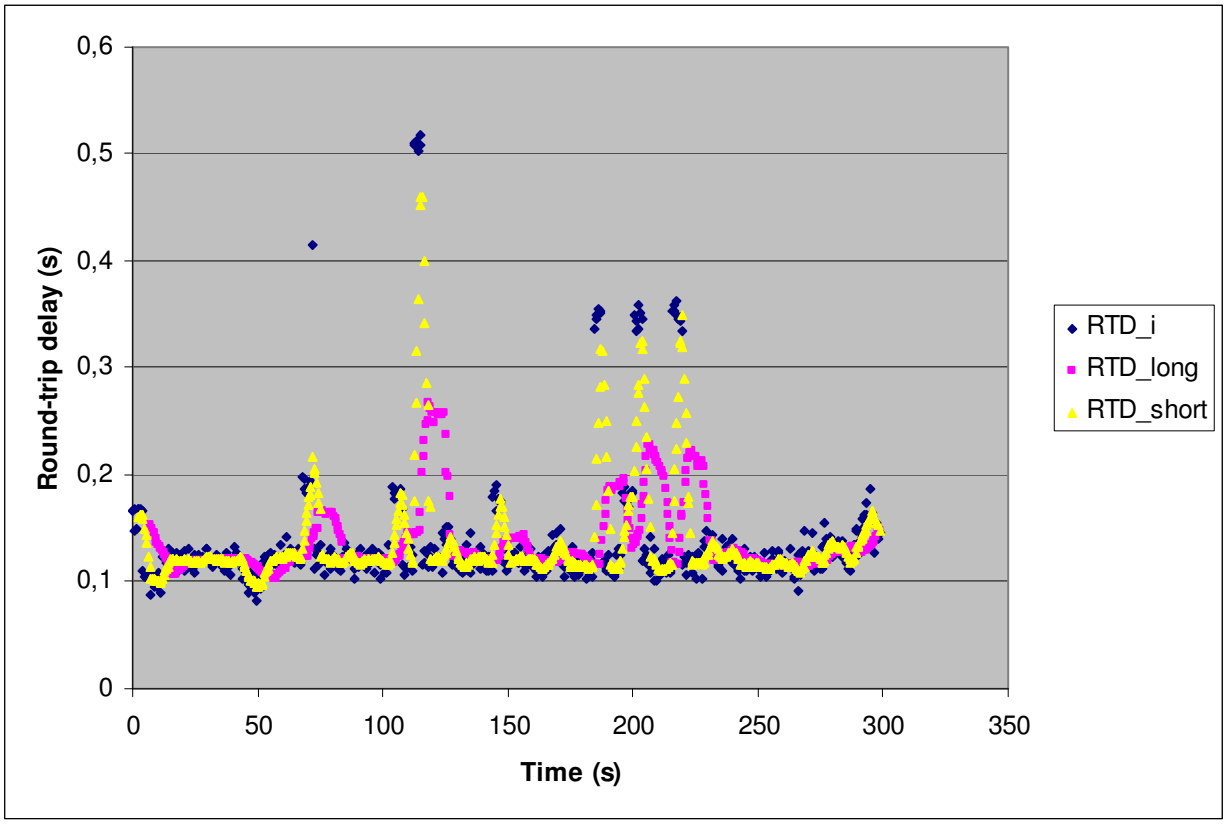

Figure 5.17: Video round-trip delay during measurement 5.

In this measurement, the transmission rate was increased to 64 kbps. In Figure 5.17, we can see that the peaks in the round-trip delay are now higher, with the highest value reaching slightly above 500 milliseconds. The clustered peaks in the RTD_i curve are caused by retransmitted SRs, while the single peak about 70 seconds is caused by a retransmitted receiver report.

As Figure 5.18 shows, 19 packets were lost during the measurement; but there does not seem to be a correlation between these losses and the round-trip delay. Furthermore, the packet losses do *not* occur in bursts, instead only a single packet is lost at each instant. This suggests that the packet losses are not a result of buffer overflow in the radio network; but rather a result of losses along the Internet path (according to Ericsson RAN experts, packet loss in the UTRAN should only occur as a result of buffer overflow).

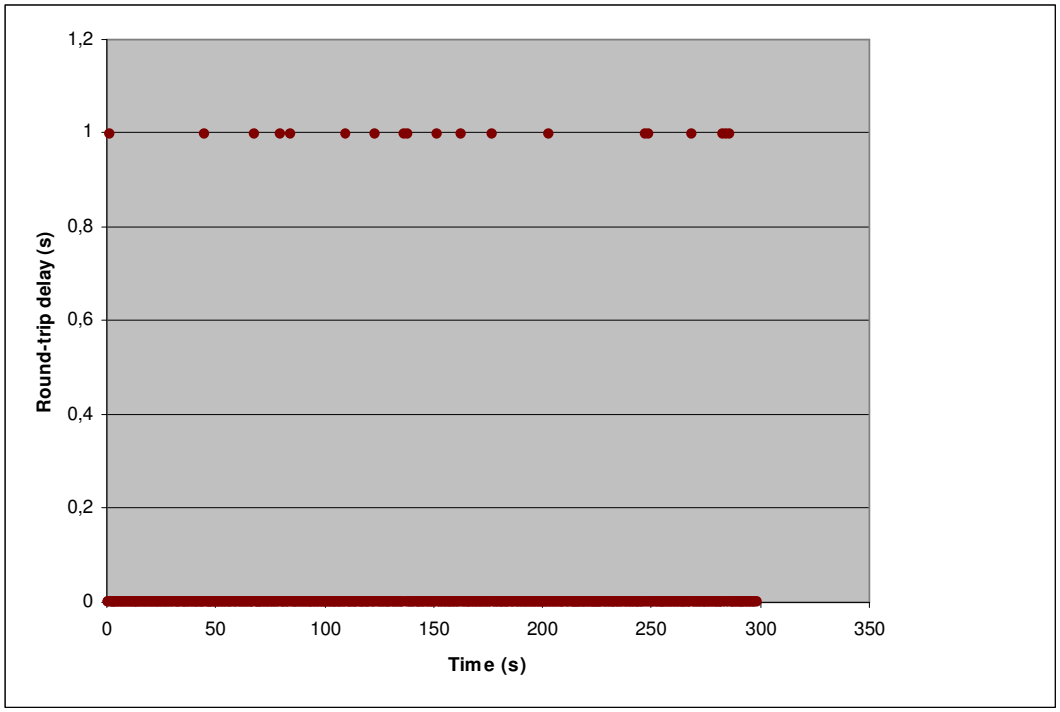

Figure 5.18: Video packet loss during measurement 5

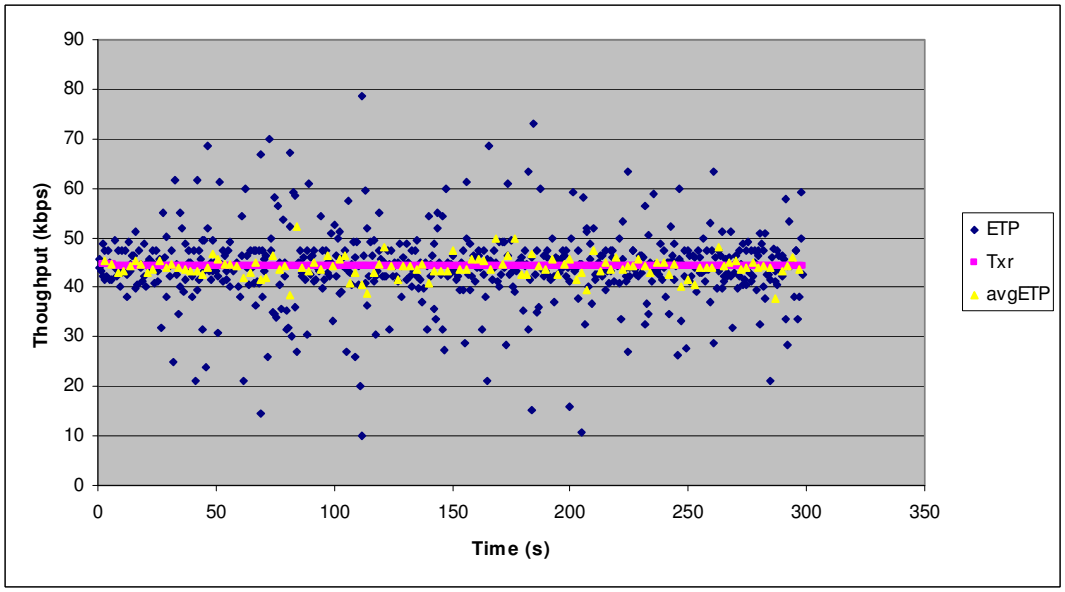

Figure 5.19: Video throughput during measurement 5.

The estimated throughput during this measurement (shown in Figure 5.19) has a higher variance than in measurement 4. The reason for this is that the K800i bases the receiver report frequency based on the AS-header in the SDP. This is more thoroughly discussed in section 5.1.4.1.

Measurement 6	
Time:	2008-07-18 10:00
RAN:	UMTS
Total session transmission rate:	128 kbps
Average period between receiver reports	0.21 s
Duration:	5 min

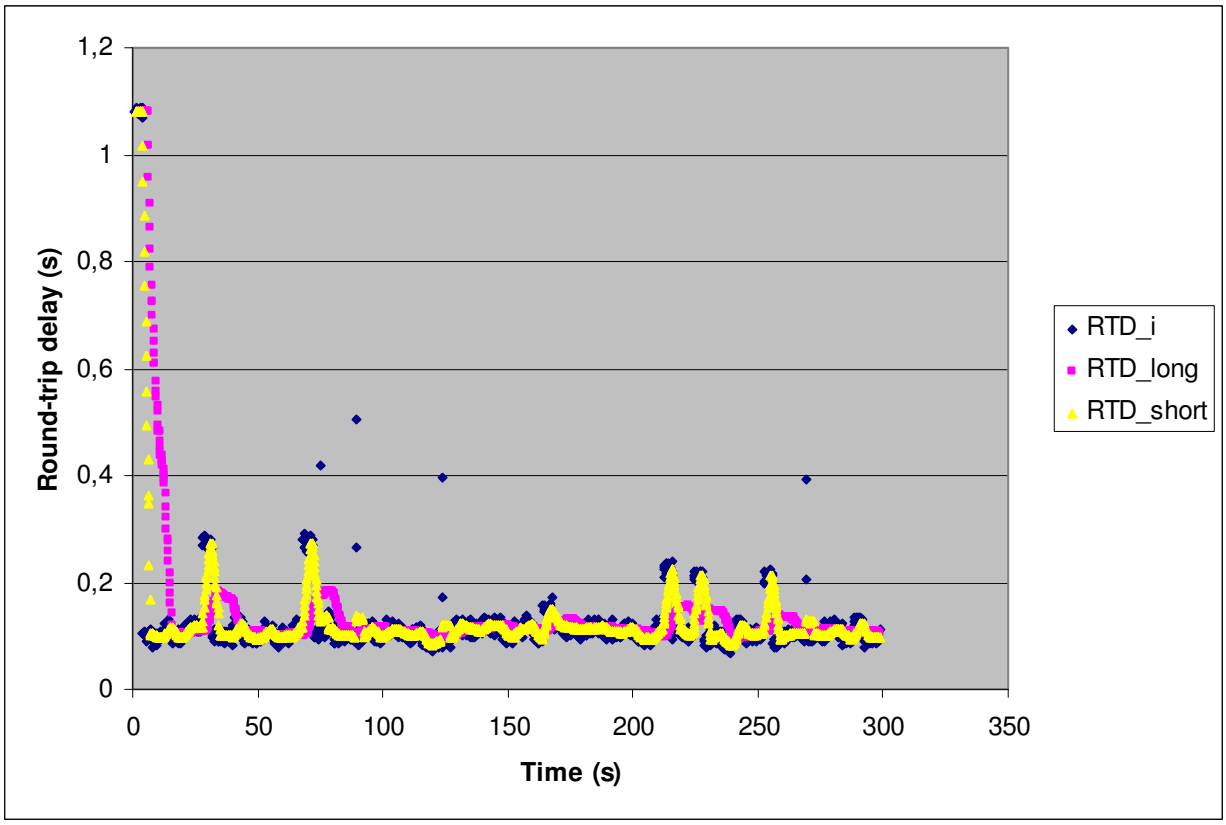

Figure 5.20: Video round-trip during measurement 6.

Figure 5.20 shows the round-trip delay during measurement 6. In the figure, we can see that there is a large delay spike (over one second) at the start of the transmission. This behavior is observed for sessions where the initial bit rate is around 100 kbps or more and also when there is a large increase in bit rate; for instance from 50 to 250 kbps. This likely caused by packets being queued while the data rate of the channel is 64 kbps and then subsequently released when the channel rate has been reconfigured to 384 kbps. For a more thorough discussion regarding RAB channel rate reconfiguration, see section 5.1.7.

The number of retransmissions and their corresponding peak delay is not much different from the 64 kbps case. Only a single packet is lost during this session.

Figure 5.21 shows the estimated video throughput during the session. The figure shows that the K800i is sending receiver reports more frequently and this causes larger fluctuations in the estimated throughput. The average throughput is calculated with a window of 1 second.

Certain extreme values in the ETP (700 kbps after 90 seconds, 1930 kbps after 124 seconds, 1446 kbps after 269.27 seconds, and 451 kbps after 269.30 seconds) have been excluded. These extreme values are naturally not realistic and are caused by the high frequency of receiver reports. Remember that the time between the receptions of two receiver reports is used in the calculation in the ETP (see Formula 5.1). The rate of the receiver reports in this measurement is so high that a retransmitted RR will arrive almost simultaneously as the following RR. We can see that all of these extreme values are preceded by retransmitted RR (as indicated by singles peaks in the round-trip delay in Figure 5.20).

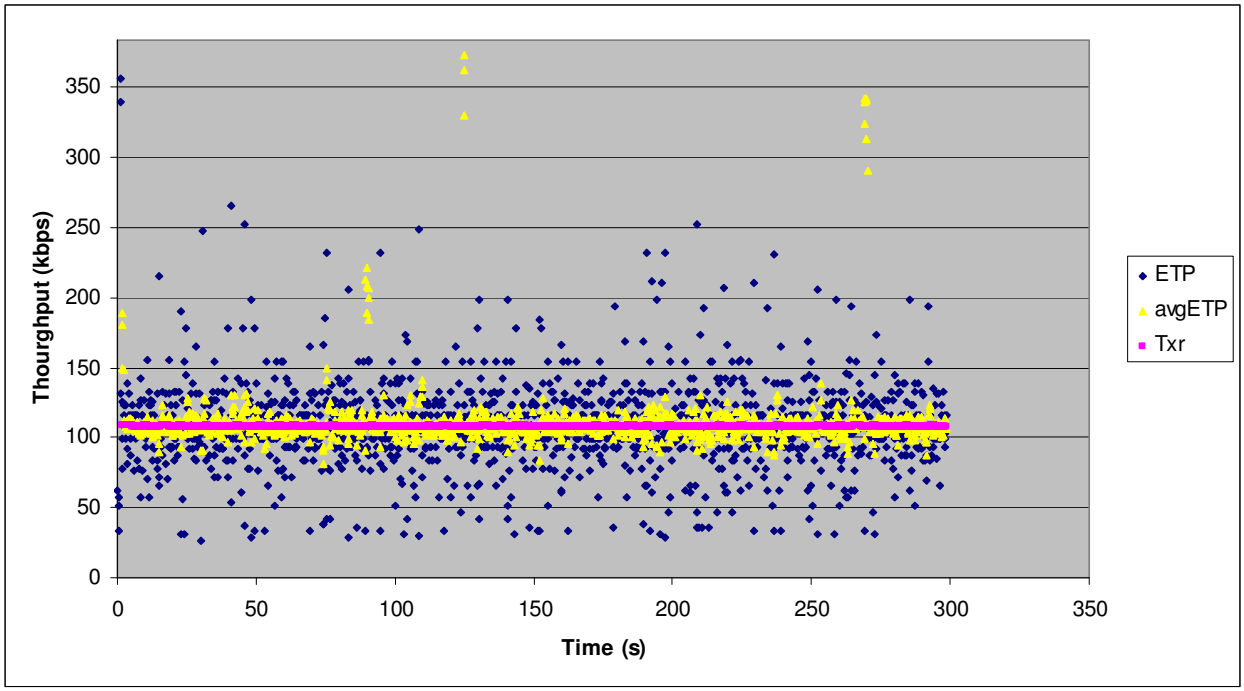

Figure 5.21: Video throughput during measurement 6.

Measurement 7	
Time	2008-07-18 10:15
RAN	UMTS
Total session transmission rate	256 kbps
Average period between receiver reports	0.11 s
Duration	5 min

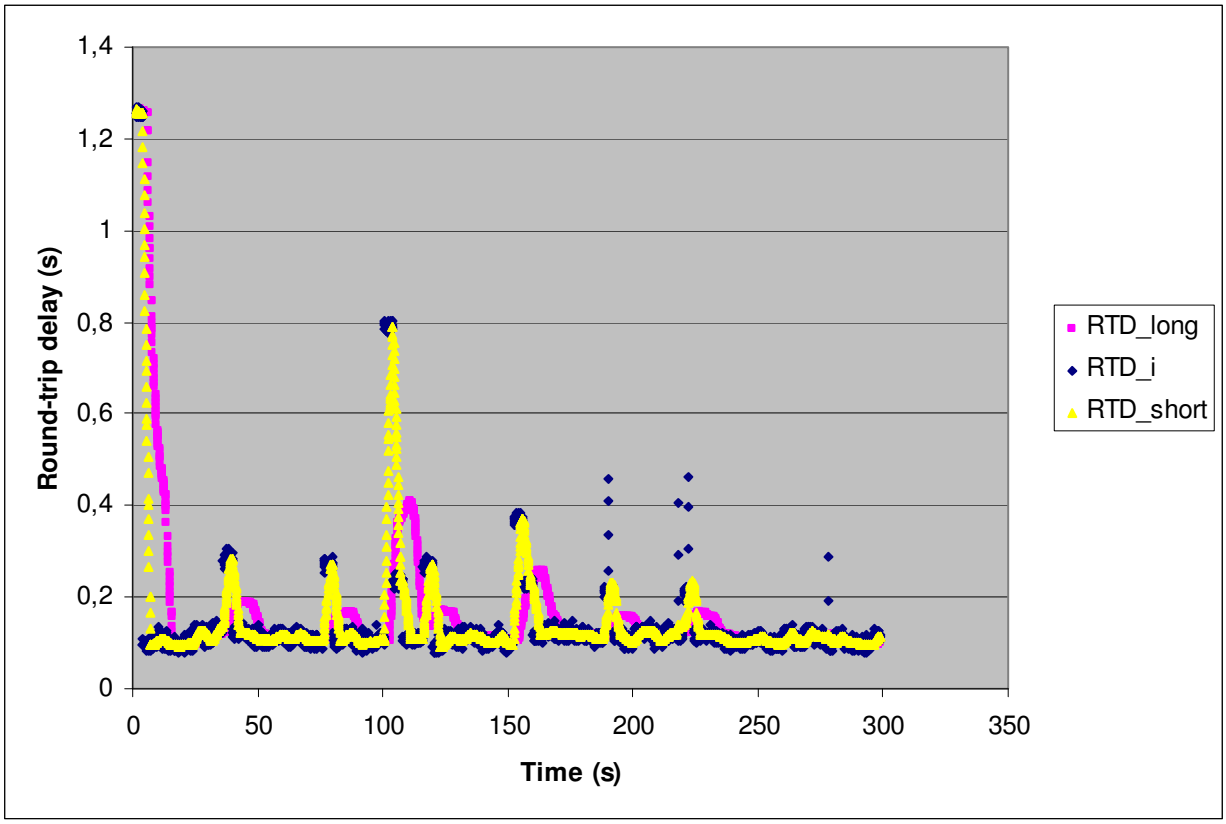

Figure 5.22: Video round-trip delay during measurement 7.

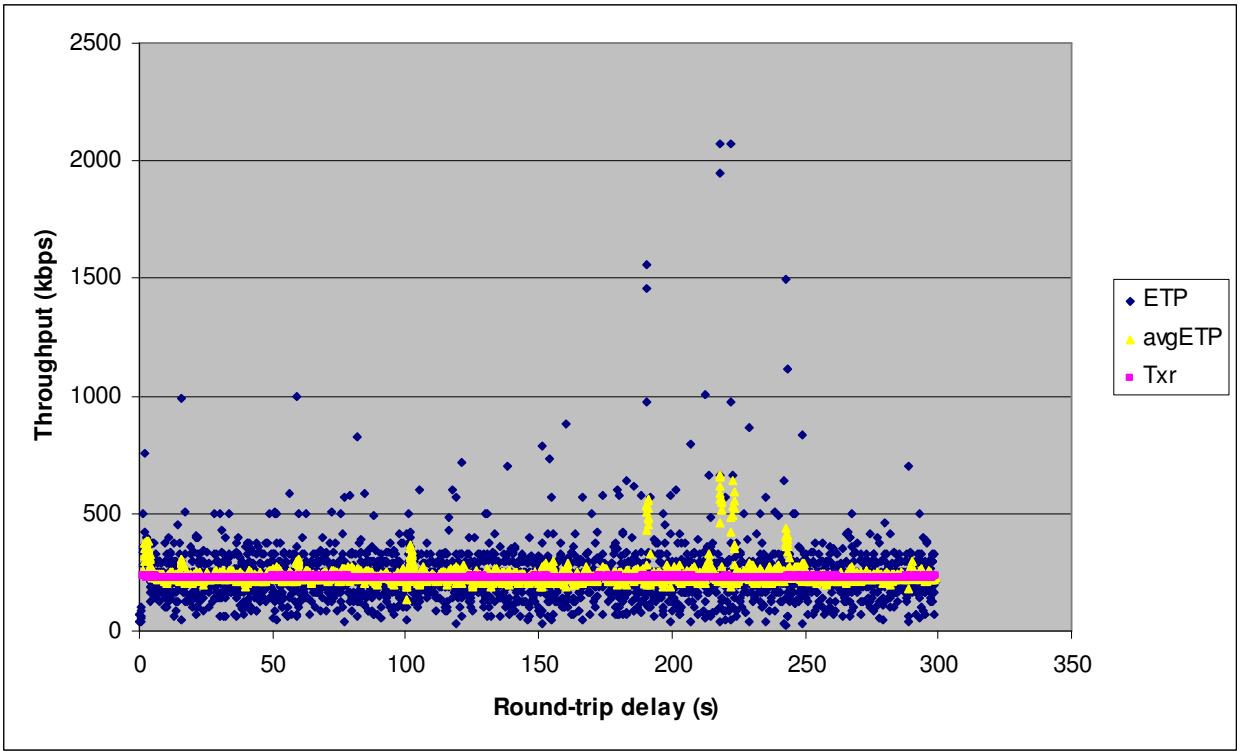

Figure 5.23: Video throughput during measurement 7.

Figure 5.22 presents the round-trip delay during this measurement. We can notice a large delay in the start of the session in this measurement as well. Figure 5.23 shows the estimated video throughput during the measurement.

Measurement 8	
Time	2008-07-18 14:20
RAN	UMTS
Total session transmission rate	30, 60, and 120 kbps (increased once a minute)
Average period between receiver reports	1.80 s
Duration	3 min

Table 5.6: Measurement 8.

Figure 5.24 shows the round-trip delay during measurement 8. Both of the increases in transmission rate should cause radio channel data rate to be increased; but as seen in the figure, it is not noticeable in the round-trip delay. Figure 5.26 shows the estimated video throughput for the session. The reason the increased transmission rate appears in the ETP-curve before it appears in the transmission rate curve is due to the fact that the ETP-curve is based upon the last two receiver reports, while the transmission rate curve is calculated using a 10 second window.

A total of 31 packets were reported as lost during this session, pretty evenly spread out between the different transmission rates. Thus there was no indication that the packet losses were correlated with the transmission rate. Furthermore, as mentioned earlier, the packets are likely not lost on the radio link at all, but rather somewhere else along the path.

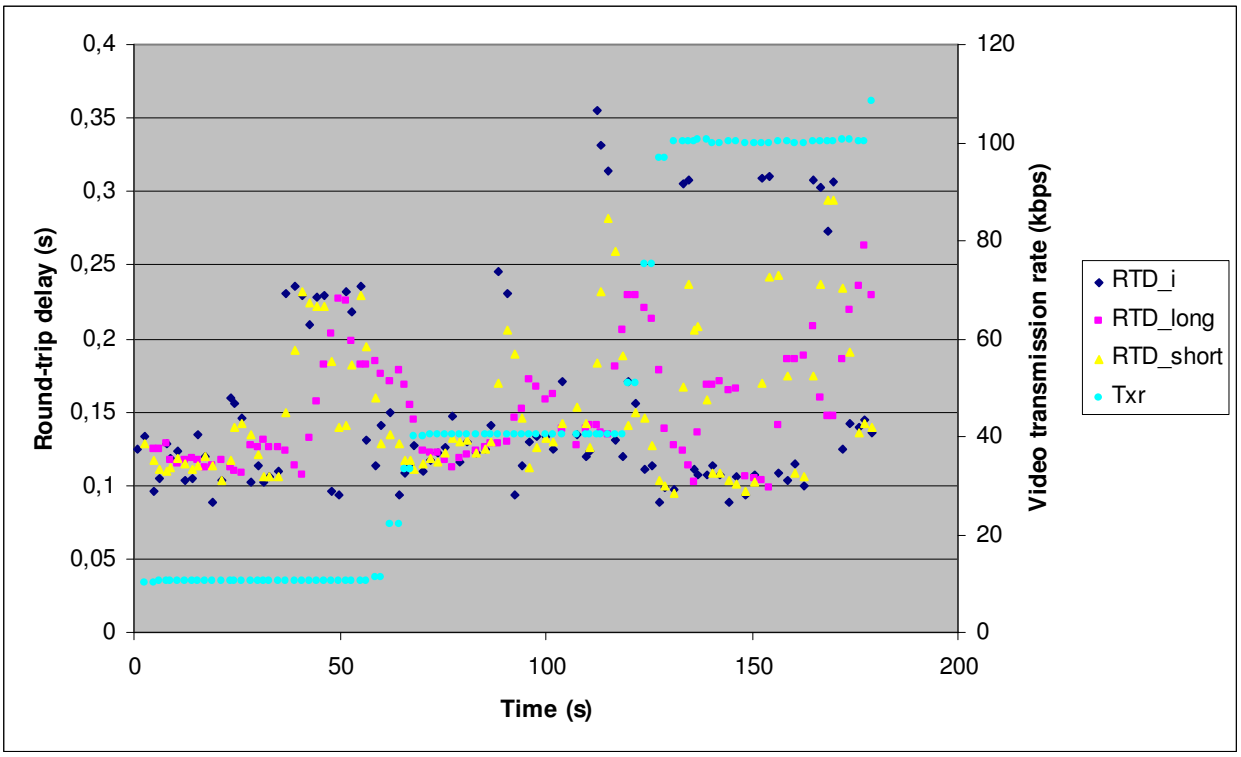

Figure 5.24: Video round-trip delay during measurement 8.

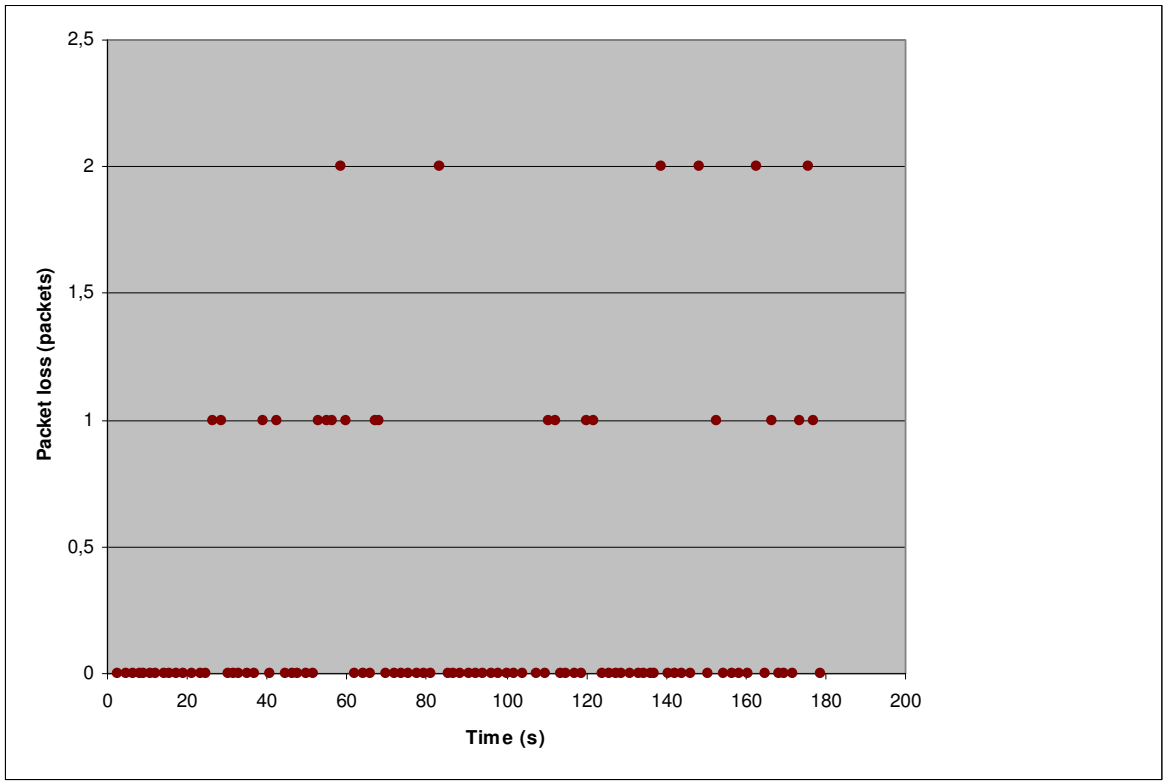

Figure 5.25: Packet loss during measurement 8.

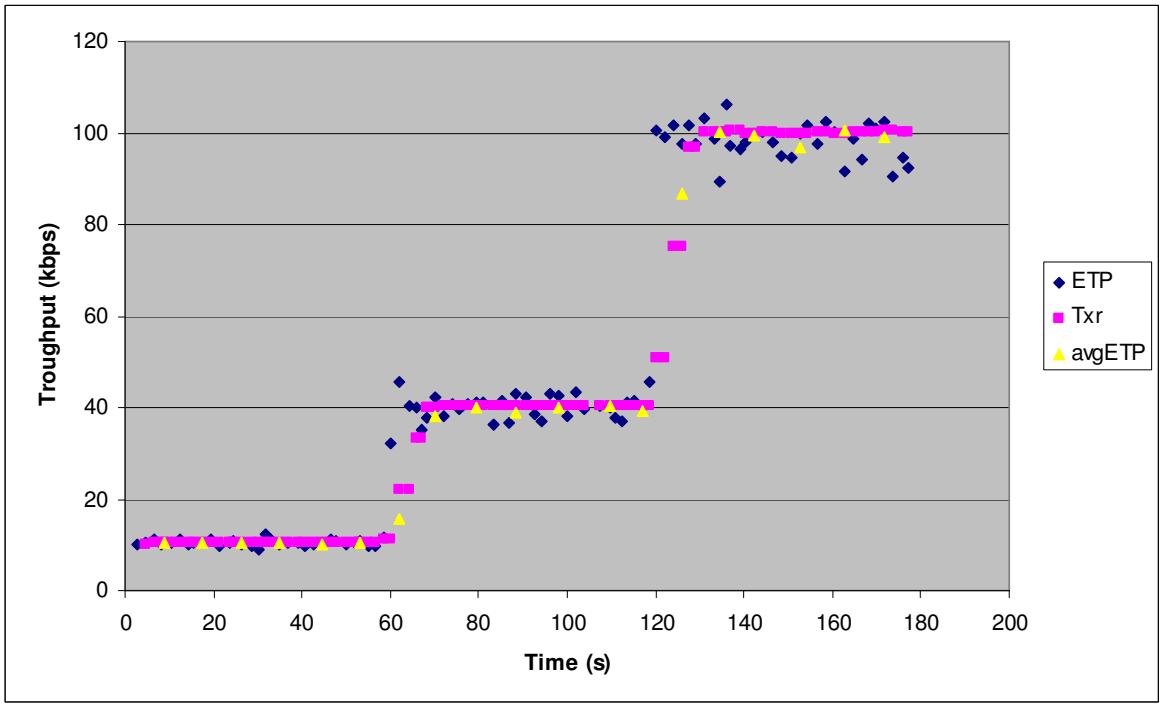

Figure 5.26: Video throughput during measurement 8.

5.1.4.1 UMTS measurement conclusions

The measurements have shown that the data rates provided by UMTS are significantly higher than for GPRS and can thus support media (such as video) content that requires higher bit rates. In UMTS, link-layer retransmissions can be detected by monitoring the round-trip delay. A single peak indicates that a receiver report has been retransmitted, whereas a number of consecutive peaks with little variance between them (a "square" form) indicates that a sender report has been retransmitted (assuming that the SR frequency is more than twice the RR frequency). During these measurements the RTCP sender reports were deliberately sent infrequent (once every 4.5 seconds) in order to be able to differentiate between retransmissions in the up and downlink. If the sender report frequency is increased, it will be harder to separate retransmissions on the uplink and downlink, but it will provide more fine-grained round-trip delay measurements. The measurements show that the K800i adapts its RR frequency based on the transmission rate of the server. A closer investigation shows that the client reacts to the bandwidth reported in the SDP-description (the server uses the b=AS header to notify the client of the required throughput for both audio and video). Table 5.7 shows what RR frequency the K800i uses for different reported bandwidth requirements.

A higher receiver report frequency means that the sampling rate for the round-trip delay and estimated throughput increases. The higher sampling rate means that fluctuations in the ETP are more noticeable. Sources of the ETP variance include:

- When the frequency of receiver reports is increased, the number of RTP packets received at the client between the transmission of each receiver report decreases. This reduces the statistical accuracy of the estimate.

- The flow of data is discrete since the encoded audio and video are packetized. This affects the throughput estimate if the receiver report frequency is high. A different number of packets may be sent during different intervals. Furthermore, the size of the data packets is not constant but may vary.

- There is an inherent variance in the end-to-end delay for receiver reports. This will be more noticeable when the RR frequency increases. Since the throughput estimation is based upon the time between receiving two receiver reports, the variance will affect the estimate but it will not actually affect the throughput.

- Receiver reports may be retransmitted, which increases the variance in end-to-end delay (see previous point).

- The bit rate of the video is not really constant, but actually varies with time. With a higher RR frequency these fluctuations are more accurately captured

Bandwidth	Average RR interarrival time	Standard deviation	RTP packets per RR[2]
32 kbps	1.6 seconds	0.30 seconds	24
64 kbps	0.5 seconds	0.10 seconds	8
128 kbps	0.2 seconds	0.05 seconds	3
256 kbps	0.1 seconds	0.05 seconds	3

Table 5.7: Frequency of receiver reports at different stated transmission rates (for the K800i).

5.1.5 EDGE measurements

During these measurements, a SonyEricsson C702 was used to measure the performance of streaming via EDGE since the K800i does not support EDGE. Otherwise the experimental parameters and configuration remained the same.

Measurement 9	
Time	2008-09-20 08:30
RAN	EDGE
Total session transmission rate	75 kbps
Average period between receiver reports	0.42 s
Duration	1 min

Table 5.8: Measurement 9.

Figure 5.27 shows the round-trip delay for the video packets when sending at 75 kbps. The delay is slightly lower and has slightly less variance than for GPRS. However, the round-trip delay is still significantly larger than for UMTS.

Figure 5.28 shows the video throughput during the session. The large spike after ~20 seconds is caused by two receiver reports arriving almost at the same time (they arrive 15 ms apart). A likely cause for this is the variance in round-trip delay. As seen in Figure 5.27 the second of these packets had a significantly shorter RTD (198 milliseconds compared to 411 milliseconds). These two receiver reports also have the same value in the LSR field meaning that they refer to the same sender report. This implies that the difference in round-trip delay between the two packets is only caused *by variance in the uplink.* The extreme value in the ETP is thus a purely an artifact of how the throughput is calculated.

[2] This is calculated as the difference in the extended highest sequence number field between two consecutive receiver reports. The value is the average value for an entire session.

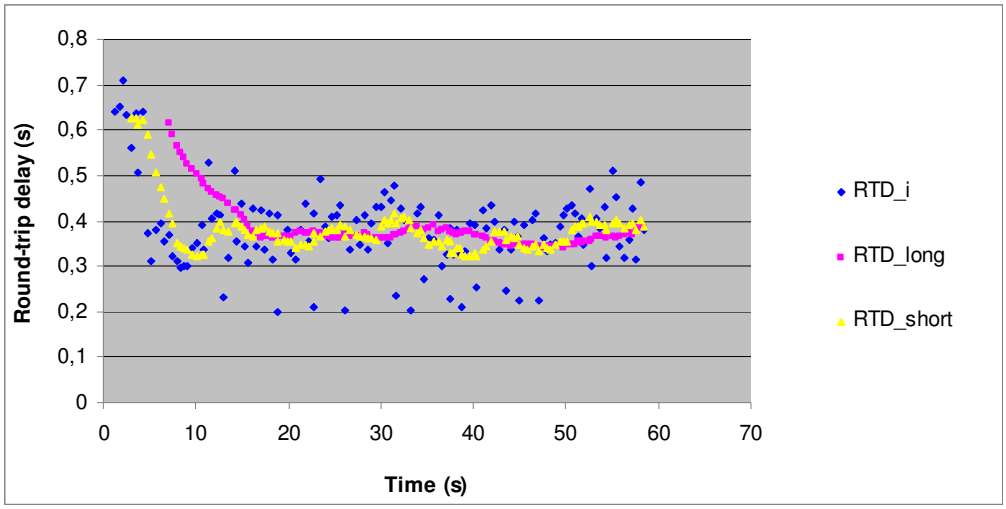

Figure 5.27: Video round-trip delay during measurement 9.

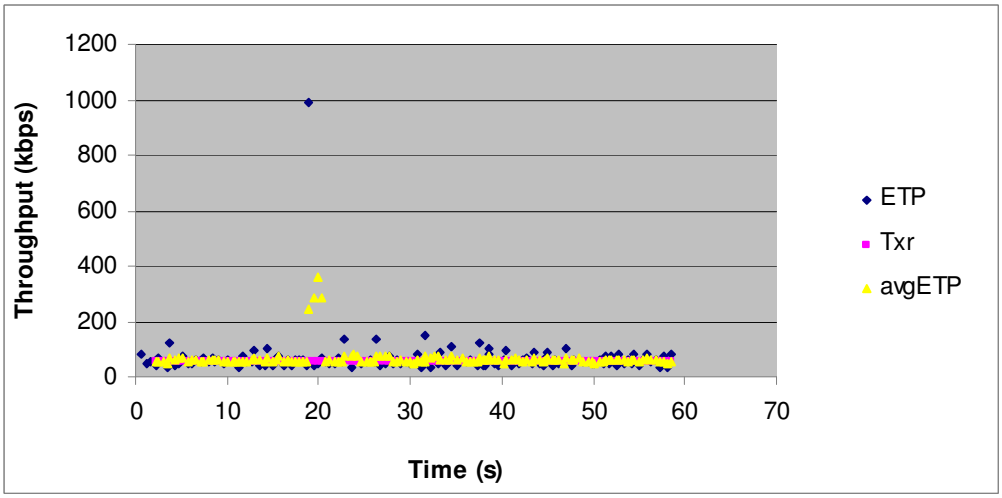

Figure 5.28: Video throughput during measurement 9.

Measurement 10	
Time	2008-09-20 08:45
RAN	EDGE
Total session transmission rate	100 kbps
Average period between receiver reports	0.33 s
Duration	1 min

This measurement exemplifies a temporary link outage in EDGE networks. Figure 5.29 shows the round-trip delay during the session. As seen in the graph, there are large delay spikes at the start of the transmission. Following these spikes where a number of packet losses (4 packets where lost after 7 seconds, another 4 after 20, and three after 28 seconds). After these spikes, no more packets where lost. These packet losses are illustrated in Figure 5.31.

When the delay is this high (2 to 3 seconds), it is reasonable to assume that the mobile device lost Internet connectivity during these periods. Consider the delay spike after ~5 seconds. By

examining the DLSR values of this and the following RR, it was determined that these receiver reports were sent 3.5 seconds apart (but they both refer to the same sender report) and they arrive at the server 600 milliseconds apart. This likely due to the fact that the delayed packet was buffered in the client while there was no connectivity, while the following packet did not leave the application layer until the connection was restored (with its DLSR field filled in before being handed over to the lower layer protocols). The fact that the transmission rate curve in Figure 5.30 also is broken during the outage does **not** imply that the server did not send any RTP-packets during this period. The interruption in the transmission rate curve is instead caused by the fact that the server is event-driven and calculates the transmission rate (as well as all other measured parameters) every time a receiver report is received.

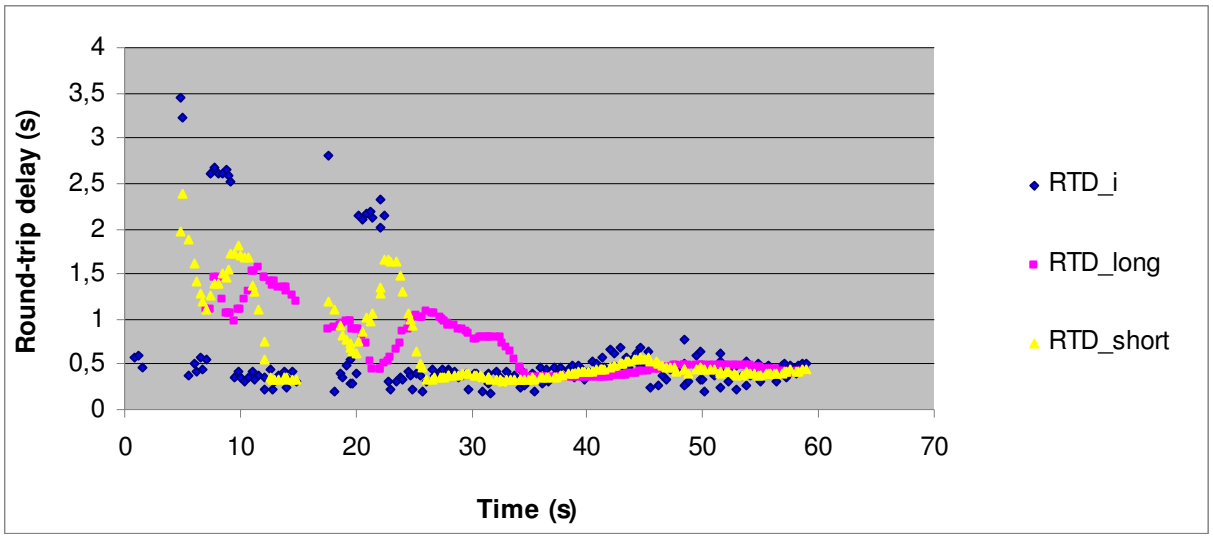

Figure 5.29: Video round-trip delay during measurement 10.

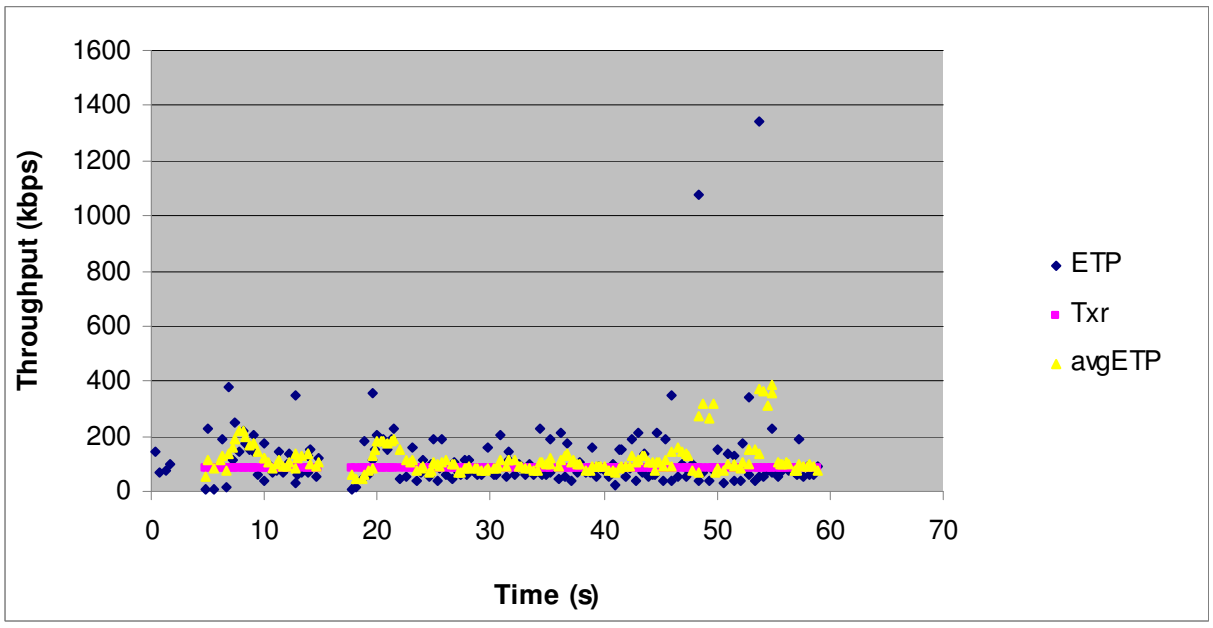

Figure 5.30: Estimated video throughput during measurement 10.

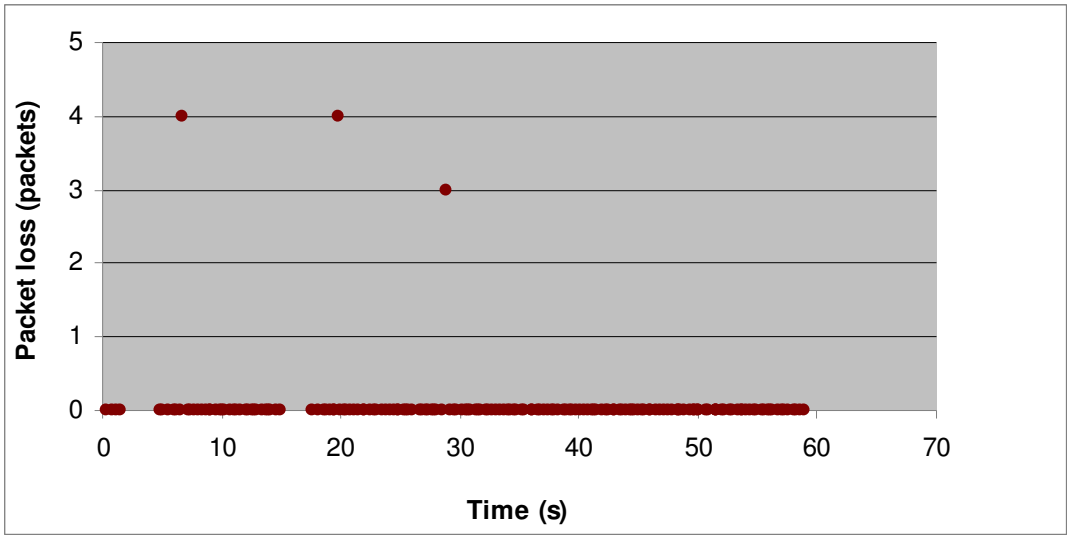

Figure 5.31: Video packets lost during measurement 10

Measurement 11	
Time	2008-09-18 16:00
RAN	EDGE
Total session transmission rate	100 kbps
Average period between receiver reports	0.31 s
Duration	1 min

This measurement shows the behavior at 100 kbps during more stable conditions. Figure 5.32 shows the round-trip delay, which does not differ significantly from the case where the transmission rate was 75 kbps (measurement 9).

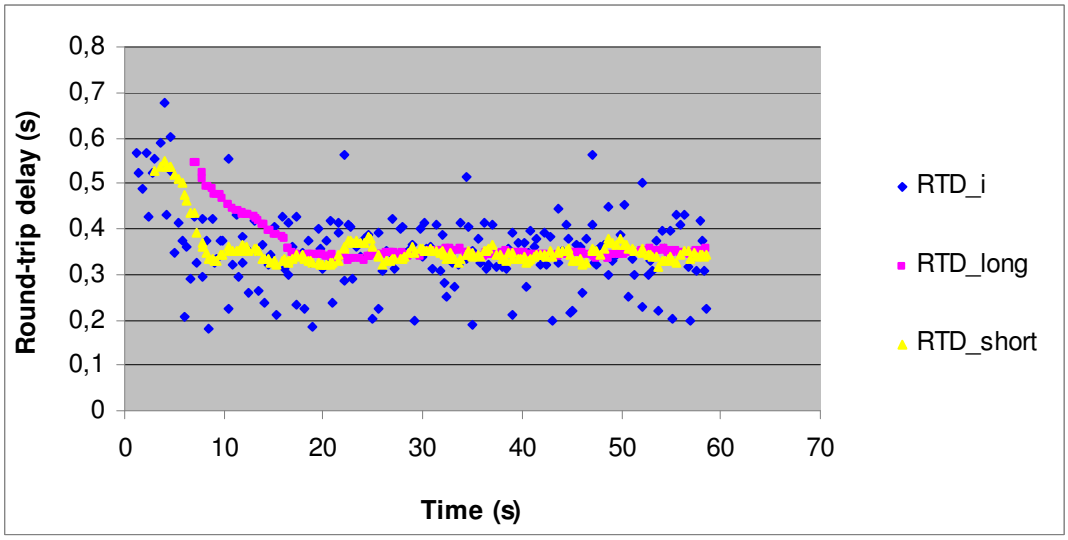

Figure 5.32: Video round-trip delay over during measurement 12.

Measurement 12	
Time	2008-09-20 09:00
RAN	EDGE
Total session transmission rate	80, 110, 140 kbps (increased every 20 seconds)
Average period between receiver reports	0.20 s
Duration	1 min

Figure 5.33 shows how the delay is affected when the transmission rate is increased. During this measurement, no packets are lost. The round-trip delay appears stable, even though the transmission rate is close to the throughputs seen in Table 5.1.

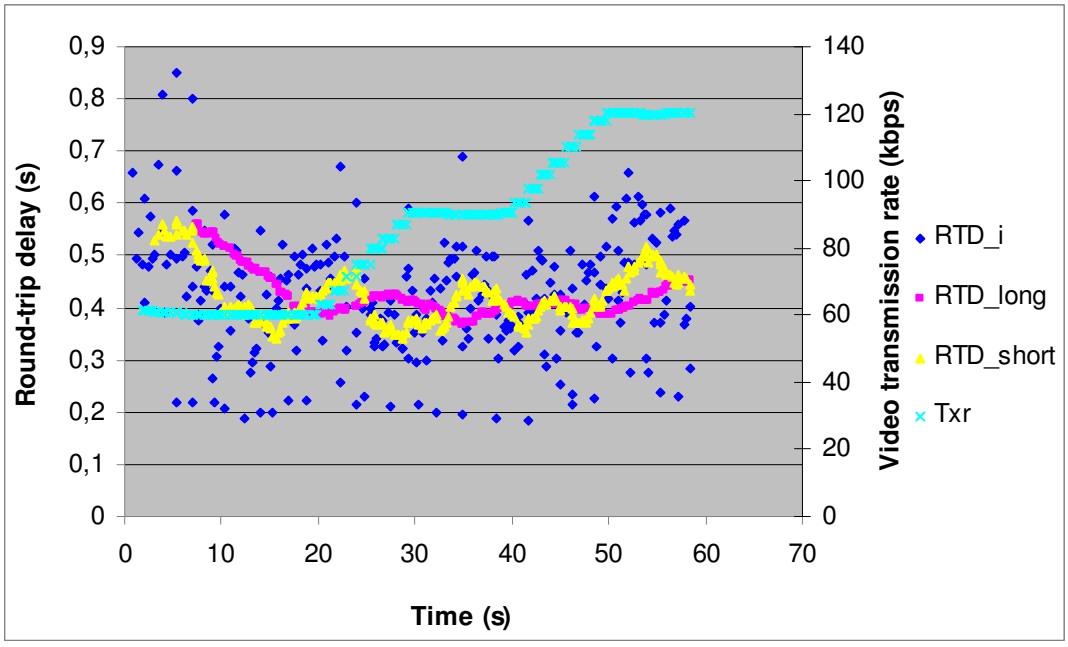

Figure 5.33: Video round-trip delay during measurement 11.

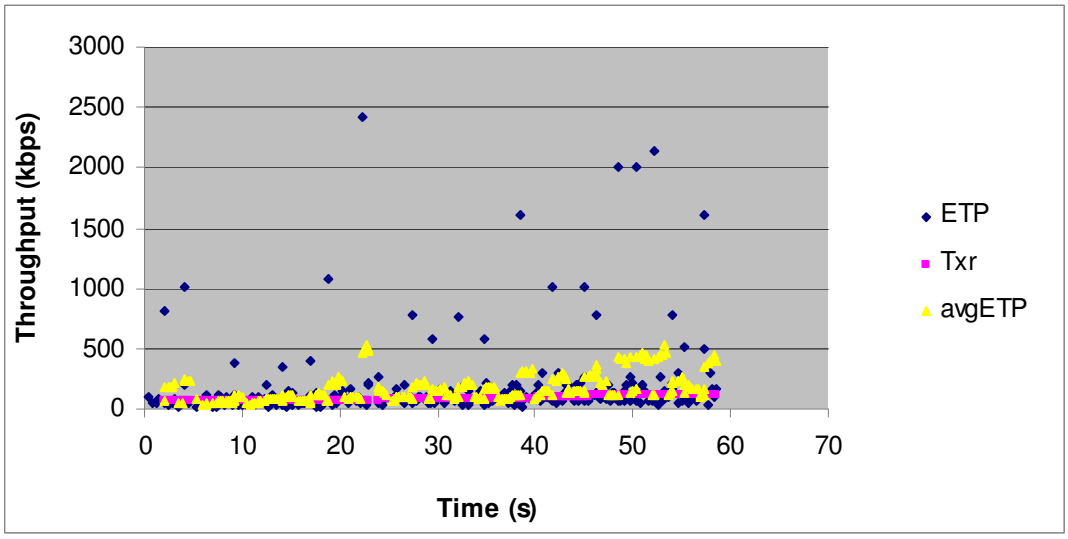

Figure 5.34: Video throughput during measurement 11.

5.1.5.1 EDGE Measurement Conclusions

As the measurements of EDGE show, the estimated throughput fluctuates quite a bit. This is caused by a number of different reasons (described in section 5.1.4.1). The reason for the high ETP peaks during the EDGE measurements is the combination of a link with high latency variance and frequent receiver reports. As shown in measurement 10, temporary link outages may occur during normal usage of EDGE. Unfortunately, it is not possible to prevent a client buffer underrun if the outages are too long as shown in measurement 10.

5.1.6 HSDPA measurements

Measurements of audio and video streaming over HSDPA were also done. The client was a desktop computer connecting to the mobile network through a "Turbo-3G" modem, the network operator was 3 (Tre). The media player used on the computer was Real Player 11.0.3a.

It is interesting to stream to a PC over HSDPA, since it is becoming increasingly popular to use "Turbo-3G" modems to provide Internet connectivity to laptop computers. One problem with the client being a PC is that most commercial media players (e.g. *VideoLAN Client* (VLC), *Quick Time Player* and *Real Player*) send *infrequent* receiver reports (about one every five seconds, as recommended by the RFC). This means that the accuracy of all estimations will be significantly lower, thus the server will not be able to react to changing network conditions as fast. It is also worth mentioning that Quick Time Player reports large and clearly incorrect values in the DLSR field in the RTCP RR (one of the reported values actually corresponded to 11 hours!). This means that using Formula 2.5 to calculate the round-trip time will yield incorrect values. A way to overcome this is to check the DLSR field in all incoming receiver reports and check for abnormal values (for instance DLSR values of over a minute). The incorrect DLSR values might then be estimated to a realistic value, for instance the average RR frequency (for all receiver reports referring to the same sender report, the reported value should increase with the RR frequency in every receiver report).

Measurement 12	
Time	2008-09-05 11:15
RAN	HSDPA
Total session transmission rate	30, 60, 120, 240, 480 kbps (increased every 20 seconds)
Average period between receiver reports	4.2 s
Duration	100 seconds

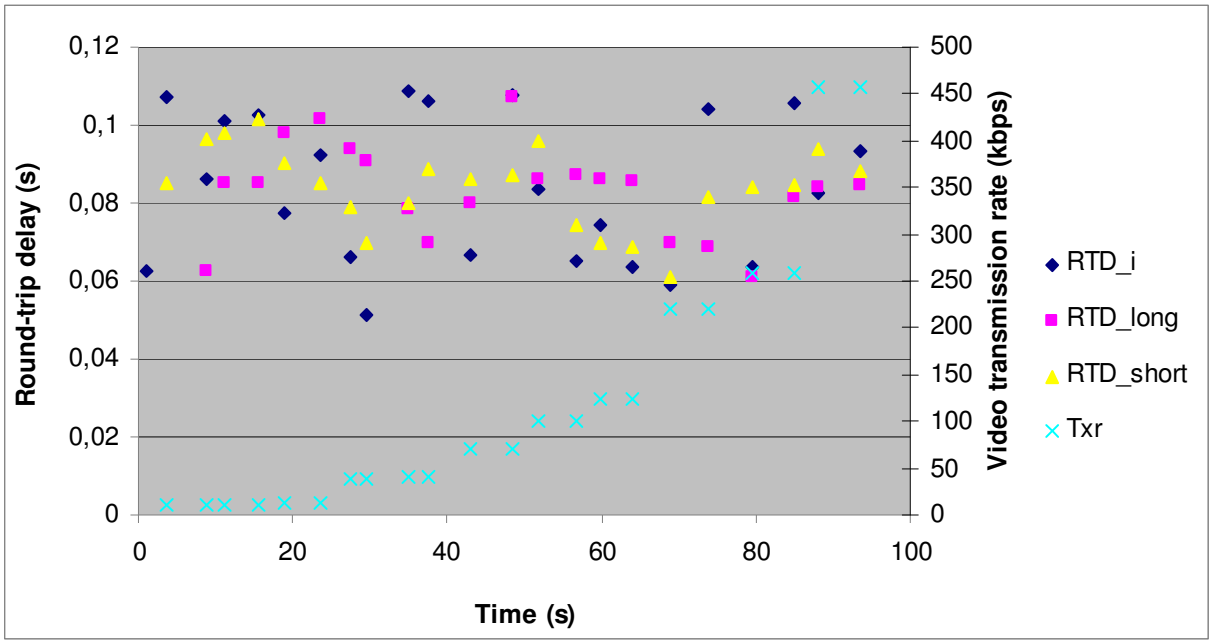

Figure 5.35: Video round-trip delay during measurement 12.

Figure 5.35 shows the round-trip delay during the session. As can be seen the delay is very stable at around 100 milliseconds. We see no significant spikes in the round-trip delay. No packets were reported as lost in the RRs. Figure 5.36 shows the throughput during the session, the low rate of receiver reports causes the estimated throughput to roughly follow the transmission rate without any significant spikes.

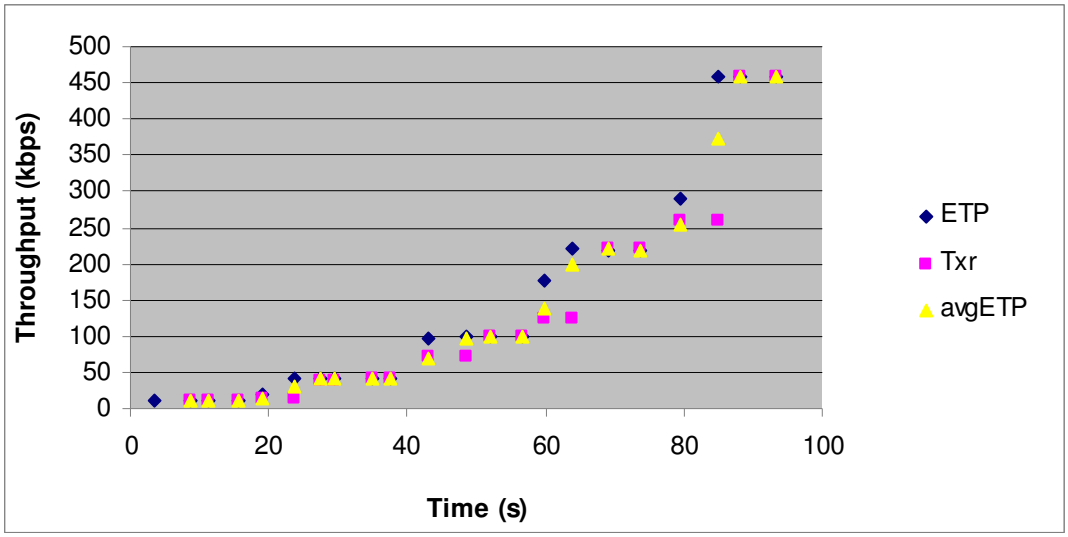

Figure 5.36: Video throughput during measurement 12

The low latency, fast retransmissions, and high throughput provided by HSDPA makes it ideal for streaming media. However, since the HS-DSCH is a shared channel, the main limiting factor in the HSDPA downlink will be the number of users in the cell. If more users enter the cell, then the available throughput for each user will decrease and if it falls below the source's transmission rate it should be possible to detect as an increase in the round-trip delay as for GPRS. Unfortunately, there is seemingly no good indicator that a user has left the cell (or stopped or reduced his or hers data traffic) and thus more bandwidth is available to the users who remain in the cell.

5.1.7 Examining WCDMA RAB assignment and QoS profile

Since the QoS profiles used by the client may impact performance, it is interesting to know which QoS profile and RAB were used during the measurements. It is typically not possible to determine this with an ordinary phone, but Ericsson has developed special test phones called TEMS terminals. TEMS terminals log radio protocol messages exchanged with base station and these logs can later be examination. During these experiments, a TEMS K800i phone was used.

Table 5.9 below shows the values in the "WCDMA PDP Context" box in the TEMS log (a "GSM PDP Context" box was also present, but it was identical to its WCDMA counterpart). The parameters of interest are the Delay, Reliability, and Precedence Class as well as the Mean and Peak Throughput and the radio priority. The delay class parameter basically tells us that we have no guaranteed maximum delay (however the delay class 3 specifies that the mean delay should be no more than 50 seconds [19]. For streaming this is equivalent to no guarantee, since the packet has surely missed its deadline after 50 seconds). The reliability class tells us that corrupted packets will be resent (Acknowledge RLC). The throughput parameters tells us that we have neither a maximum throughput (256 000 octets/s equals 2 048 000 bps which is the maximum theoretical throughput for WCDMA) nor a guaranteed throughput ("Best effort"). Radio priority is the lowest priority. To summarize, no guarantees what so ever are provided. For GPRS, the values in the QoS profile had the same value as for 3G, except for the peak throughput class which was set to 32 000 octets/s (256 kbps).

QoS parameter	Value
Active PDP context	1
PDP NSAPI	5
PDP SAPI	3
PDP Delay Class	3
PDP Reliability Class	Unacknowledged GTP and LLC; Acknowledge RLC, Protected data
PDP Precedence Class	Reserved
PDP Peak Throughput	Up to 256 000 octets/second
PDP Mean Throughput	Best effort
PDP Radio Priority	3

Table 5.9: WCDMA PDP Context

Tests were conducted in order to learn which RAB channel rate was used for different transmission rates and if these changes could be observed in RTCP feedback. The tests show that a 64/64 kbps RAB (i.e. 64 kbps downlink/64 kbps uplink) was allocated for a transmission rate of 50 kbps, a 128/64 kbps RAB for a transmission of 100 kbps, and a 384/64 kbps RAB for any transmission rate higher than about 115 kbps. Switching between these transmission rates causes a corresponding change in the radio bearer rate. Unfortunately, changes in the RAB Channel Rate do not seem to manifest itself in the RTCP feedback.

Worth noting is that the UMTS network uses an aggressive rate switching algorithm. The channel rate is always switched directly from 64 kbps to 384 kbps and is later downgraded to 128 kbps if the channel is not utilized sufficiently (e.g. for transmission rates of approximately 100 kbps and below). This is probably to better aid the TCP slow start algorithm [52].

5.2 Measurements in a WCDMA emulator

To aid in algorithm development, a network emulator called REDWINE was installed on a FreeBSD machine. REDWINE is propriety Ericsson software, which captures packets from the network interface and passes them through a number of modules before releasing the packet back onto the network. PortwineLite is a module for REDWINE, which emulates the WCDMA air interface. The delay on each packet introduced by PortwineLite is based upon trace-files, i.e. delays seen in a live network. Several trace-files for different bearer and error rates were available. The use of a network emulator is vital when tests are to be performed in a controlled environment. Additionally, because the tests can now be performed using PCs rather than phones it is possible to instrument the traffic measurements and even the applications much more thoroughly.

REDWINE/PortwineLite was used in order to test how the RTCP feedback is affected by different channel data rates and error rates. As a client for these tests, a simple RTSP client and RTP sink were used (part of Ericsson Research's streaming server software), running on a Windows XP machine (with an Intel Pentium 4 clocked at 3.4 GHz and 1 GB of RAM). The RTSP client was configured to send receiver reports every 250 milliseconds to match the rate of the reports sent by the K800i.

Figure 5.37 shows the round-trip delay when the transmission rate is kept steady at 50 kbps, while the bearer data rate is increased every 20 seconds (0-20 s: 64 kbps, 20-40 s: 128 kbps, 40-60 s: 384 kbps). The error rate during this session is 1%. It is not possible to detect when each switch was made by just looking at the graph, this means that it is not possible to detect increased bandwidth by monitoring the round-trip delay. Figure 5.38 shows the estimated throughput during the same session and it does also not seem to hint that more bandwidth has become available.

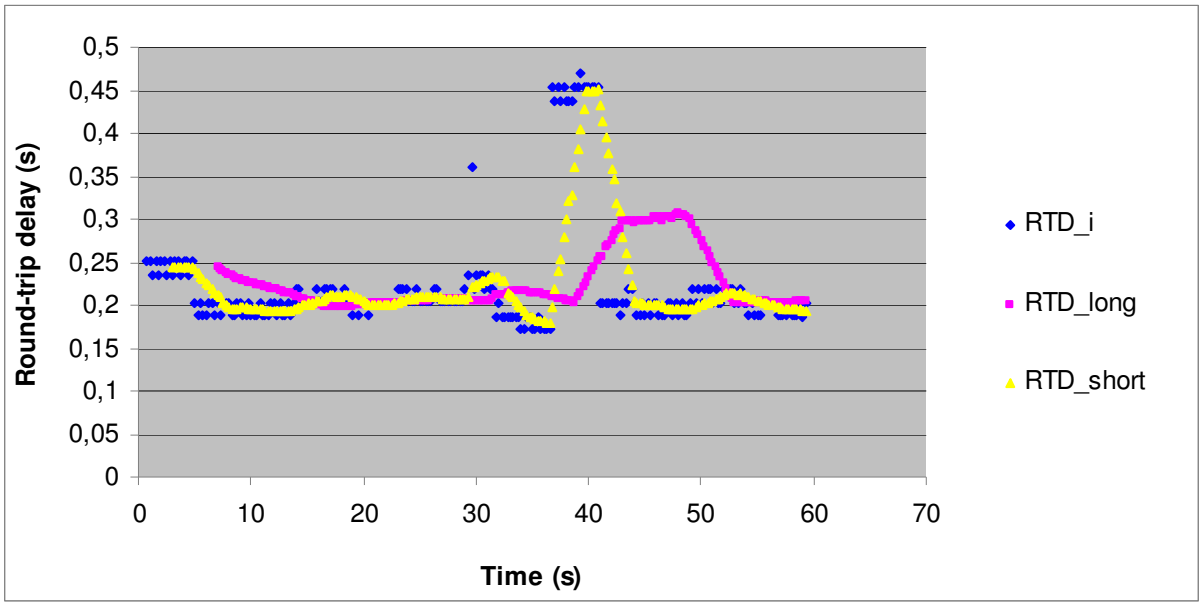

Figure 5.37: Video round-trip delay over REDWINE with increasing bearer rate

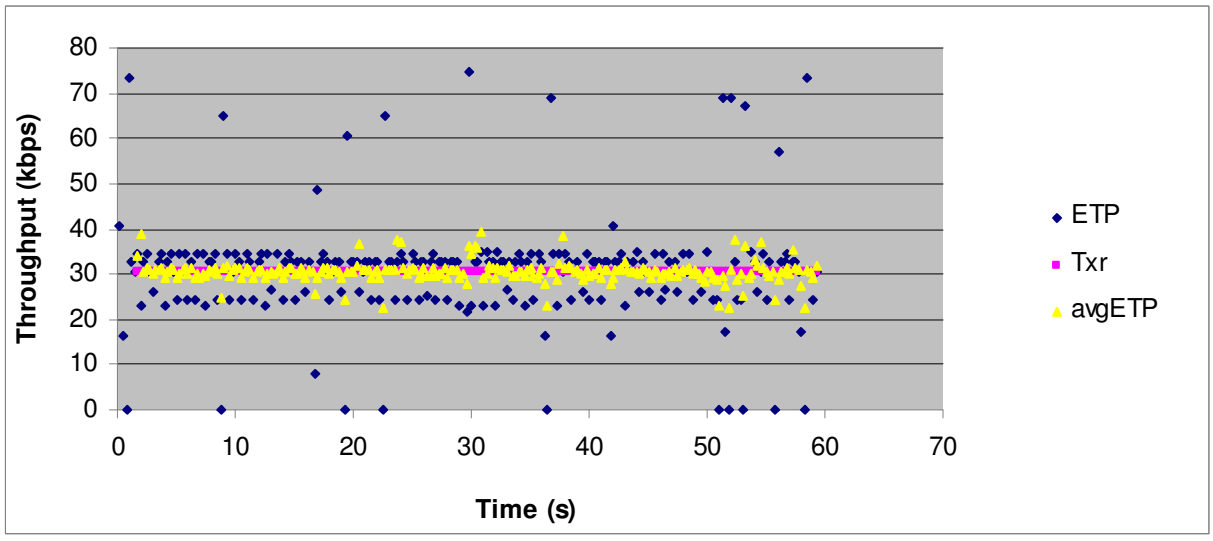

Figure 5.38: Video throughput over REDWINE with increasing bearer rate

Figure 5.39 and Figure 5.40 show the resulting graphs when the error rate is increased. The transmission rate was kept constant at 100 kbps and the channel rate at 128 kbps. After 30 seconds, the error rate is increased from 1% to 5%. This is most noticeable due to the number of increased RR retransmissions as indicated by the spikes in round-trip delay and estimated

throughput. This seems to indicate that changing radio conditions can be detected by monitoring the number of retransmissions. We can also see that the number of spikes in the estimated throughput increased, this is caused because more receiver reports are forced to be resent due to the higher error rate.

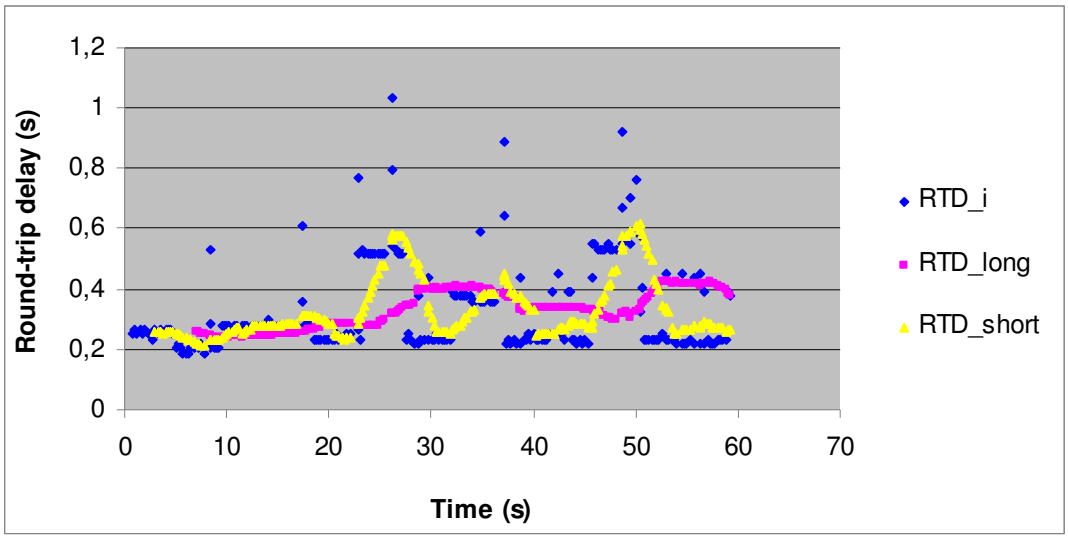

Figure 5.39: Video round-trip delay over REDWINE with increasing error rate.

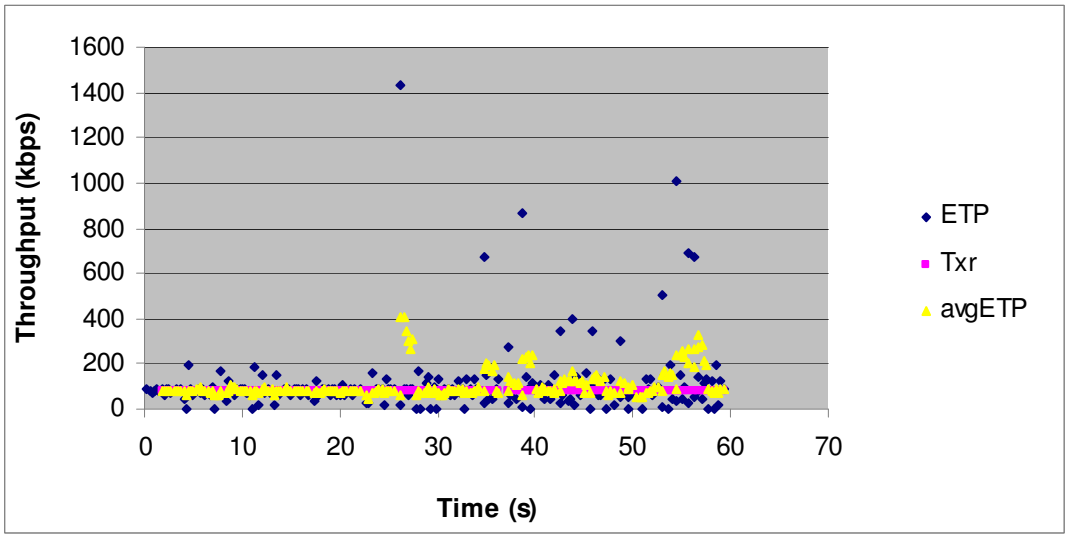

Figure 5.40: Video throughput over REDWINE with increasing error rate.

5.3 Summary and conclusions

In this chapter, the performance of video streaming in both live and emulated networks have been examined. The measurements show that increasing round-trip delay precedes packet loss, when the transmission rate exceeds the maximum available throughput. Furthermore, retransmissions in WCDMA networks can be detected by round-trip delay spikes (uplink retransmission) and temporary increases in delay (downlink retransmission). Retransmissions are an indication of the error rate, which limits the available bandwidth.

Unfortunately, the measurements did not provide any clear indicator of when there was increased bandwidth. It appears that it is not possible to detect the current bandwidth simply looking at the traffic which is sent/received, without performing some kind of transmission rate control. Furthermore, in UMTS networks, the interactive RAB bearer rate will not be increased unless the user's throughput is above a certain threshold, which means that additional bandwidth will not be available until the transmission rate is actually increased. Because of this, the rest of this thesis will focus on how to perform transmission rate control, when to perform it, and how the success of an upswitch may be evaluated. The remainder of this report will focus specifically on how to perform transmission rate control in order to provide bit rate adaptation to the available link bandwidth for live streaming multimedia contents.

Chapter 6: Detecting the RAN Type

If different algorithms, or algorithm settings, must be used for different RAN types, it is essential to know what RAN type or types (and thus what the potential data rates are available). The optimal way to achieve this is to use explicit signaling by the client. This approach will immediately give the server perfect knowledge of the RAN type; however, there is no widely supported mechanism for performing this.

If the server uses a phone database, the information in the database might be used to exclude some RAN types. Unfortunately, this will however only give the server knowledge about which RAN type the user *might* connect through, not which RAN the user *is* connecting through. It is, for example, possible that although the mobile phone supports HSPA, is actually connected through EDGE due to coverage issues. Moreover, a phone database must regularly be maintained in order to include new models.

A third possibility might be to look at the round-trip delay. As the measurements in the previous section showed, the RTD in UMTS/HSPA networks is lower than for GPRS/EDGE. Thus it should be possible to differentiate between them by looking at the average RTD. However, this is only true if the Internet delay is always of the same approximate size. If one user has an Internet delay of more than 100 ms and another user has almost zero Internet delay, it may, for instance, be difficult to distinguish between UMTS and EDGE.

Since the RAN types investigated have different degree of variance in their round-trip delay, it should be possible to determine which RAN type is in use by looking at the variance in round-trip delay. This may be detected by observing the variance of the round-trip delay[1]. Figure 6.1 shows the variance of the round-trip delay measured over GPRS, 3G, and HSDPA. As shown in the figure, the GPRS round-trip variance is significantly higher than for the other RAN technologies. Since EDGE's delay characteristics are similar to GPRS, the EDGE delay variance should resemble GPRS's. The two bumps on the 3G (WCDMA) curve are the effects of SR retransmissions.

The variance is calculated according to Formula 6.1, where N is the number of samples and $\bar{x}$ is the sample average. In the graphs below a window of three seconds was used to calculate the variance for each point.

$$Var(X) = \sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

Formula 6.1: Variance

[1] The assumption is that the delay variance in the Internet part of the path is insignificant. Otherwise, that variance will impact the overall variance and will result in it not being possible to tell which interface you are using

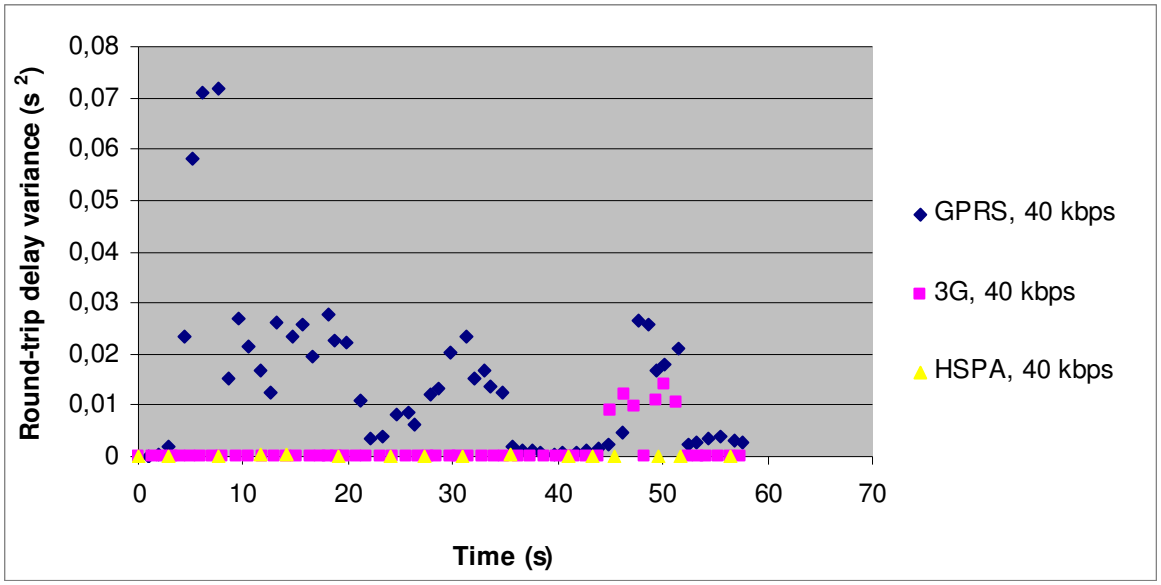

Figure 6.1: Round trip variance for different RAN types. Transmission rate 40 kbps

Intuitively, the reader might think that the jitter reported in the RTCP RR should follow the same trend (as the jitter reports the average time period between the receptions of two RTP packets). However, as shown in Figure 6.2, the difference between the jitter for the different technologies is not as clear. This is due to several things; for instance that the jitter is calculated as a moving average which smoothens out extreme values, that the variance is calculated using the round-trip delay and is thus a combination of both the variance in the uplink and in the downlink, and that the variance squares the difference, thus making it more detectable. Furthermore, the jitter is based upon the timestamp values in the RTP packets, but media data is not necessarily sent in playout order.

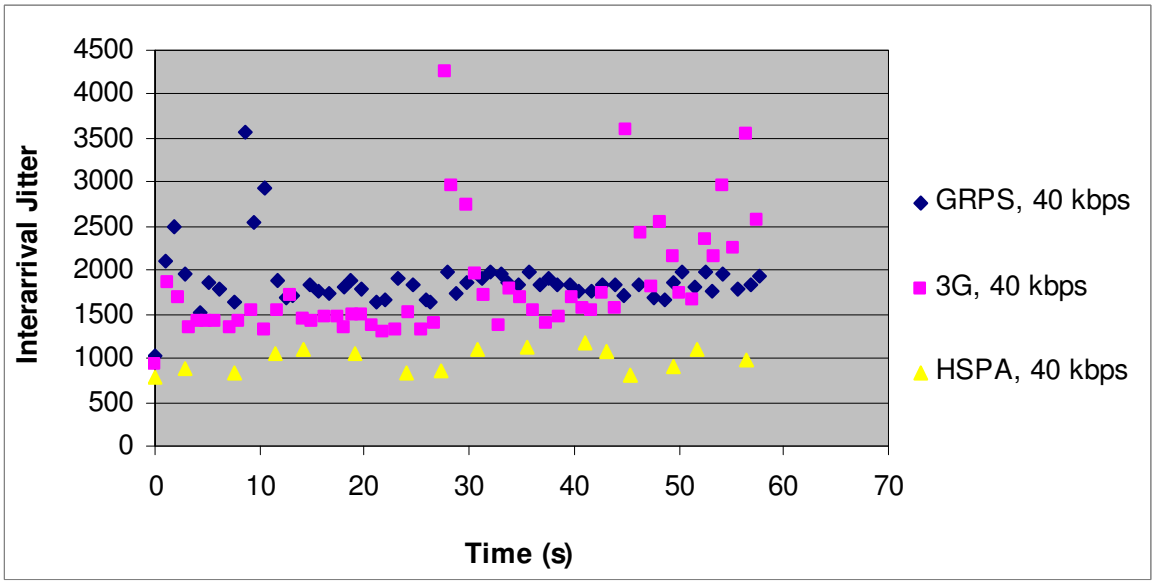

Figure 6.2: Interarrival jitter during the same sessions as Figure 6.1**.**

6.1 Implementation and Testing of the Detection Algorithm

An algorithm for determining the RAN type was implemented in the server by calculating the variance of the round-trip delay during the first six seconds of the session. The accuracy was tested using both the REDWINE/PortwineLite emulator and a live network. For testing via the emulator the rtspclient from Ericsson Research's streaming server package was used, while both a SonyEricsson K800i and a Nokia N73 handset were used for testing via a live network.

The algorithm was tested with different receiver report frequencies. This was done by modifying the rtspclient and changing the values included in the SDP, which the K800i seems to base its receiver report frequency on (as described in section 5.1.4.1). The Nokia phone did not seem to adjust its RR frequency based on the SDP, so it could only be used with its standard RR frequency (which is about 1 RR per second). Twenty test runs were made for each bearer and receiver report frequency. Sender reports were sent every 2 seconds. The variance threshold to differentiate between 2G and 3G was set to 0.0030 s^2.

Table 6.1 presents the result of the test. The results show that for higher *Block Error Rates* (BLER), the accuracy is very poor. But this is to be expected since the high error rate causes frequent retransmissions, which in turn increase the variance. Low variance indicates *high link quality*. In the live UMTS network and for 1% BLER in the emulator, the accuracy was very high (above 90%). For GPRS, the accuracy was also good with the almost all test being above the threshold (and thus classified as 2G networks).

Bearer	Average period between RR	Average variance	Below threshold
REDWINE, 128 kbps, 1% BLER	100 ms	0.0015 s^2	17 (85%)
REDWINE, 128 kbps, 1% BLER	249 ms	0.0002 s^2	20 (100%)
REDWINE, 128 kbps, 1% BLER	983 ms	0.0055 s^2	15 (75%)
REDWINE, 128 kbps, 5% BLER	109 ms	0.0252 s^2	3 (15%)
REDWINE, 128 kbps, 5% BLER	249 ms	0.0660 s^2	1 (5%)
REDWINE, 128 kbps, 5% BLER	983 ms	0.0254 s^2	1 (5%)
GPRS, SE K800i	292 ms	0.1241 s^2	1 (5%)
GPRS, SE K800i	968 ms	0.1919 s^2	2 (10%)
WCDMA, SE K800i	240 ms	0.0002 s^2	20 (100%)
WCDMA, SE K800i	990 ms	0.0005 s^2	19 (95%)
WCDMA, Nokia N73	905 ms	0.0030 s^2	15 (75%)
EDGE, Nokia N73	857 ms	0.1061 s^2	4 (20%)

Table 6.1: Round-trip variance for different receiver report frequencies

The algorithm currently uses the variance to skip probing at a low rate for users connected via high quality links. If the variance is below a threshold, the algorithm immediately switches up to the highest content rate below 384 kbps (WCDMA RAB maximum data rate). Test in the live network in Kista has shown that the upswitch to the 384 kbps RAB in WCDMA incurs a delay of about 1 second. This is illustrated in Figure 6.3. The media bit rate was initially set to 36 kbps and the variance was calculated after 6 seconds as previously mentioned. Since the variance was low, the algorithm switched to a higher media bit rate (250 kbps). The "CR"-curve is the normalized curve of the current media bitrate.

About three seconds after the switch to a higher video bit rate, a large peak in the RTD appeared. This behavior has consistently been seen throughout the thesis work, when a large switch in bit rate occurs. The most likely reason for this is that delay is introduced by packets being buffered while the channel data rate is being reconfigured. This is also the reason for the delay peaks at the beginning of the transmissions during the measurements in the previous chapter. These peaks are not seen when the bit rate is slowly increased up to high rates.

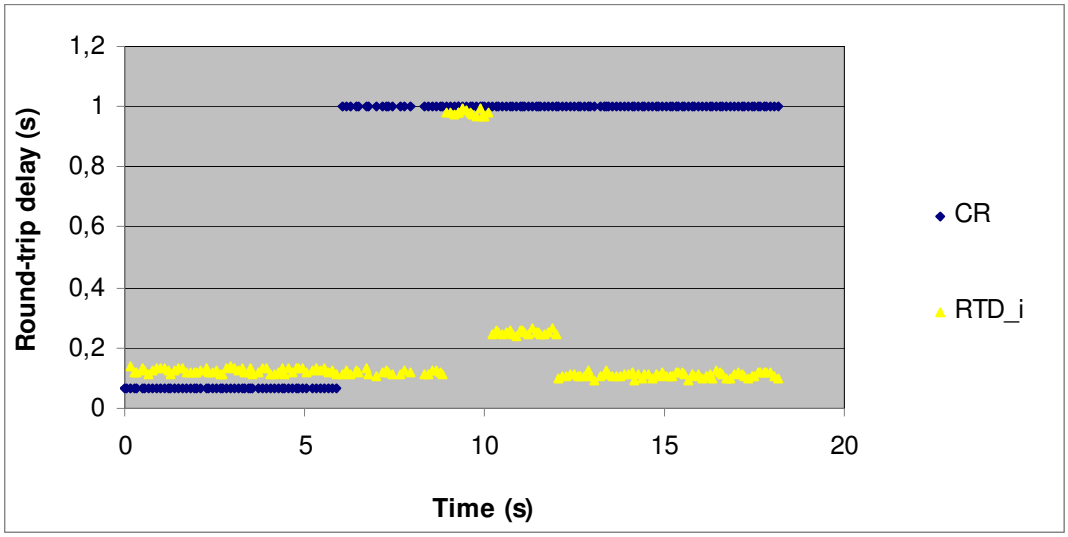

Figure 6.3: RTD peak caused by RAB reconfiguration.

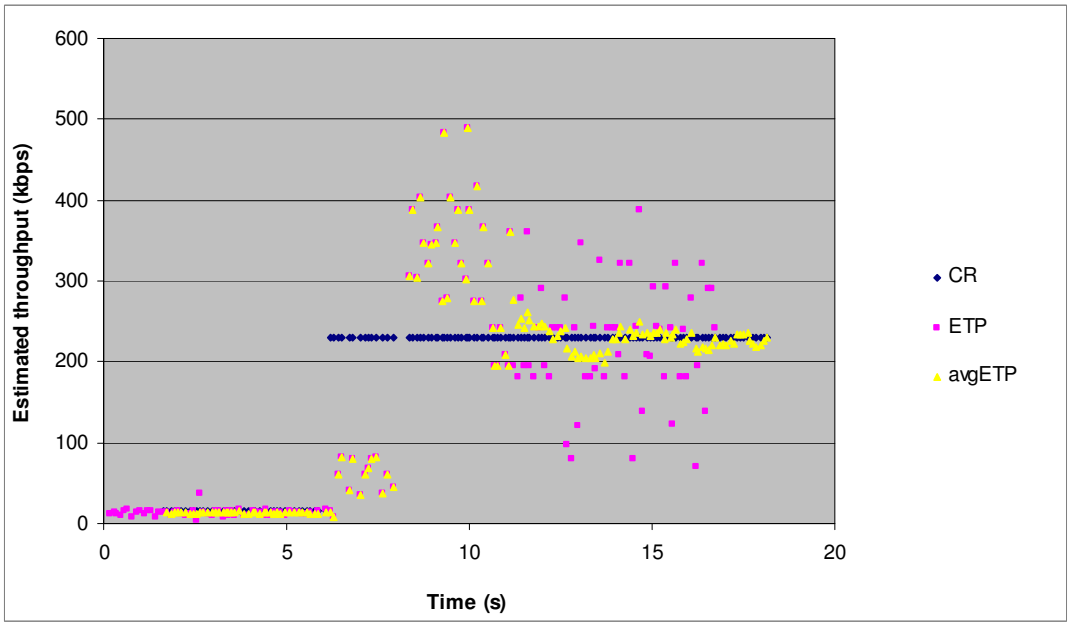

Figure 6.4: ETP during RAB reconfiguration

It is important that the algorithm does not misinterpret these peaks as a sign of network buffers filling up and thus switch down the bit rate again. Currently, the algorithm is instructed to ignore RTD peaks for five seconds after making a large upswitch.

As mentioned, the algorithm currently jumps to the highest content rate below 384 kbps. This might be a too aggressive approach since it is not guaranteed that a 384 kbps RAB is available (however, this has never happened to the author during any of the tests). It might therefore be preferable to avoid switching as aggressively and instead switch to a rate below 200 kbps (which should be supported by EDGE as well).

If the initial bit rate is about 100 kbps in UMTS, then an initial delay of about one second is incurred on the first packets as described in the previous chapter. When the delay drops back to normal after this, there is a huge delay variation. This will fool the algorithm that it is connected via a 2G technology. Of course, this large initial delay is a clear indication that the RAN used is UMTS, and therefore a future improvement would be to implement detection of this.

The tests performed in this chapter also shows that the algorithm mistakes 3G technologies for 2G - if there are a lot of retransmissions. These two problems are related; round-trip delay spikes causes spikes in the variance, thus skewing it. A way to circumvent this problem might be to only consider the variance between receiver reports referring to the same sender report. This of course requires that RR frequency is higher than the SR frequency; at least three RR per SR. Extreme values caused by retransmitted RRs also needs to be accounted for. Another way to compensate for the retransmission might be to instead look at the long-term delay, which will conceal most of the retransmission. This is of course not possible to use in a fast upswitch mechanism, since the period to calculate the long-term delay will negate any speed gain; but might be useful if the RAN type detection is just for some other purpose (such as adapting the settings of the algorithm).

Chapter 7: Implementing and Evaluating the Upswitch Algorithm

As discussed in the chapter 5, nothing indicates that changes in the link data rate that may be detected by passively monitoring feedback from the receiver. Furthermore, in 3G networks, the user will not be allocated a higher data rate channel unless needed. Due to these facts the server must increase its sending rate in order to find the maximum throughput available as well as to force the assignment of a higher data rate carrier (in the case of UMTS).

For pre-encoded content it is trivial to increase the transmission rate. This can be done for instance by mimicking the TCP transmission control algorithm, i.e. slowly increasing the transmission rate until packet loss occurs (however, some packet loss is acceptable for streaming media) or until the round-trip delay starts to rapidly increase, indicating that a buffer along the path is starting to fill up (which should occur before packet loss).When the feedback indicates that the maximum bandwidth has been reached, the highest media bit rate below this bandwidth may be selected. If the client is 3GPP Release 6 compliant, the RTCP NADU APP-packet might also be used to indicate the client buffer's fullness.

As mentioned in chapter 1, it is not as easy to perform rate control for live content, since there is no media content (i.e. useful data) available to increase the transmission rate with (without actually increasing the media bit rate). There is of course always the possibility of not performing any content rate switching for live content (like the Helix server). If no switching is performed, care must be taken when choosing the video bit rate; if it is set too low the end-user will suffer from unnecessarily poor quality and if set too high the user might not be able to view the content. Another possibility might be to offer the media in several different bit rates, where you select one rate at start-up, then never switch during the session. In that case the problem is how to decide which rate to choose for each user. Can the end-user be trusted to make an intelligent decision regarding his or her available bandwidth? Basing the decision upon information received from the client during the RTSP setup may not always provide the optimal rate, since the information received from the client is not always correct (the K800i, for instance, always sets the RTSP Bandwidth header to 64000, even though it is connected via GPRS).

There are some methods that might achieve transmission rate control for live content. This could for instance be done by sending the content at an uneven rate but keeping the average transmission rate the same as the content rate (e.g. a period when transmission rate is lower than the content rate to buffer packets at the server, followed by a period of sending the buffered packets at a rate higher than the content rate). Another possibility might be to perform bit stuffing in order to increase the transmission rate. Upswitching might also be done by simply switching to a higher content rate periodically (if the radio environment appears favorable, e.g. low packet loss rate, low round-trip delay).

This chapter will discuss several ways to achieve this transmission rate control and describes the implementation and evaluation of an algorithm for doing this. The requirements for such an algorithm are:

- The playback in the client should not be interrupted due to lack of media content (i.e. it should avoid buffer underrun).
- The bit rate of the media should be as high as possible

- The server should avoid flipping up and down between two media bit rates, as this is noticeable in the quality observed by the user and will be annoying for the user. This behavior may occur naturally e.g. when the server only has access to a number of pre-defined bitrates, and one or more of these bit rates is just above the currently available bandwidth.
- Finding and switching to the optimal bit rate should be as fast as possible

7.1 Increasing the Transmission Rate for Live Media

To increase the transmission rate for pre-encoded content is trivial as there is always additional media to be sent. For live media it becomes increasingly difficult since data is produced in real-time and can of course not be sent before it is produced. In this section, a number of ways to increase the transmission rate for live contents are examined.

7.1.1 Periodic Increase of Content Rate

The most straightforward approach to content rate switching is to periodically increase the content rate, then monitor the feedback in order to decide whether the upswitch where successful (as currently done by Ericsson Research's streaming server). This approach will always switch up to the optimal rate eventually. However, if the highest media bit rate is higher than the currently available bandwidth such an algorithm will switch up to a transmission rate higher than offered by the network. If the difference between two offered media bit rates is large, there is a risk that the transmission rate might be increased significantly above the available data rate. This increases the risk of both overflowing the radio network buffers and the risk of client buffer underrun. For instance, assume that the media is encoded in 50, 100, and 250 kbps and the available bandwidth is 128 kbps. When the media bit rate is increased from 100 to 250, the transmission rate will be 132 kbps more than the available throughput. If it takes the server three seconds to discover this, the buffers in the network will be filled with 132*3=396 kilobits, which is more than 1.5 seconds of media. This will also cause new data to be forced to wait in the buffer which will increase the one-way delay, possibly so much so that the client buffer runs empty. As measurement 3 in the chapter 5 illustrates, if the transmission rate is just a few kilobits over the available data rate, the delay might increase up to several seconds severely risking a client buffer underrun. Taking all of this into account, it is clear that a more conservative approach is required.

7.1.2 Probing by Using Bit Rate Modulation

A method that was tested was to vary the transmission rate in square wave pattern, as done in [3]. This is implemented by first transmitting media data at a slower rate than received from the encoder in order to buffer data at server. This buffered data is subsequently sent at a higher rate than the content rate. There are two motivations for this; first to force a bearer rate upgrade in UMTS networks and second to evaluate if it was feasible to transmit at the higher rate. The implementation was tested using REDWINE/PortwineLite. The idea is that if there is sufficient bandwidth to provide the higher rate, then the measured ETP should roughly correspond to the sent square wave, while the wave would be distorted if there is insufficient bandwidth. However, when the algorithm was tested in the emulator, it proved that it was not possible to detect additional bandwidth in this way (as depicted in Figure 7.1).

There are also a number of problems associated with this approach. First of all, if the period of low transmission rate is too long, the client buffer may be starved. Furthermore, if the lower transmission rate is too low, it may do the opposite of what the algorithm is supposed to achieve; it may trigger a downswitch in bearer rate.

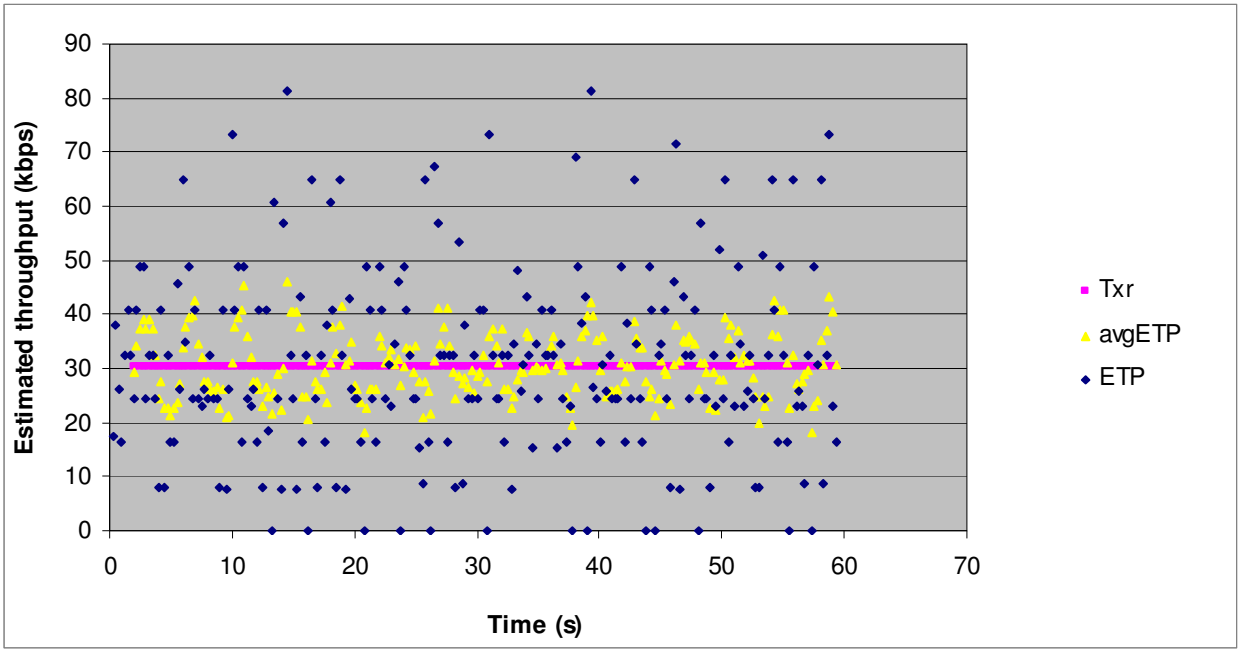

Figure 7.1: Transmission rate varied between 12.5 kbps and 62.5 kbps every 2 seconds. Bearer rate changed from 128 kbps to 64 kbps after 20 seconds and changed back to 128 after 40 seconds.

7.1.3 Utilizing Existing Connections for Stuffing Data

The idea behind this approach is to utilize one or several of the five existing connections (audio RTP, audio RTCP, video RTP, video RTCP, and RTSP) to send probing data. For a live source, there is no additional media data available to use for probing, thus some other kind of data will need to be used to increase the transmission rate.

One possibility is to use the RTCP channel to send additional RTCP messages, for instance by increasing the rate of the sender reports or sending extra RTCP SDES packets.

Another possibility is to insert filler data into the media stream. This could for instance be done by inserting empty P-frames (which signal no change from the previous picture and stuffed to an appropriate size) into the media stream or using the ordinary media frames, but inserting extra bit stuffing patterns.

The RTSP-connection could of course also be utilized for probing by sending additional RTSP messages. However, since TCP has its own transmission control algorithm, it might be more difficult to know for certain exactly how much more data is being inserted onto the connection during a given time period.

Inserting stuffing data provides the server with fine-grained control over the transmission rate. This enables the server to increase the transmission rates in smaller steps, compared to simply

switching to the next higher media bit rate. This enables the server to detect that it is exceeding the maximum rate at a lower transmission rate, which avoids filling up the network buffers as much. Furthermore, since the content rate of the media is not actually increased, it will avoid the problem of the content quality flipping up and down (which happens when there is not enough bandwidth to support the next higher rate/quality), which might be annoying to the user. The obvious disadvantage of this approach is of course that "unnecessary" (i.e. that will not be used by the receiver) data is being sent over the channel. The upswitching time will also be longer compared to speculative upswitch to higher media rate/quality.

For the purpose of the algorithm, the exact way in which way stuffing data is inserted is not important, but rather at what rate. Because of this, the algorithm will make no assumptions on how the stuffing data is generated. Since the server has complete control of the encoder in the lab environment, the extra data will be generated simply by increasing the encoding rate during the development of the algorithm.

7.2 Basic Algorithm Concept

The main idea behind the algorithm is to slowly increase the transmission rate in order to find out how much traffic the network can handle without filling up the network buffers. Using bit stuffing, as described in section 7.1.3, the transmission rate is increased in steps until the next media rate is reached. The purpose of this is to detect if the available bandwidth is insufficient for the higher rate *before* the actual increase in video bit rate. This will avoid flipping between two different bit rates, which may provide poor user experience. Furthermore increasing round-trip delays will be detected earlier, and thus the source will not fill the network buffers as much. The behavior of the algorithm is illustrated in Figure 7.2.

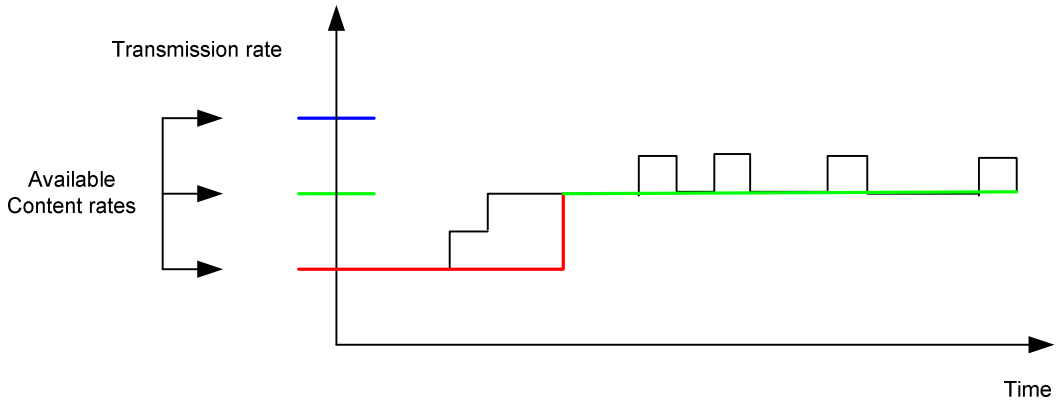

Figure 7.2: Basic algorithm behavior. The black curve shows the transmission rate. The coloured curves show the content rate.

7.2.1 State Machine

The algorithm is implemented as a state machine, as depicted in Figure 7.3. The following sections describe each state.

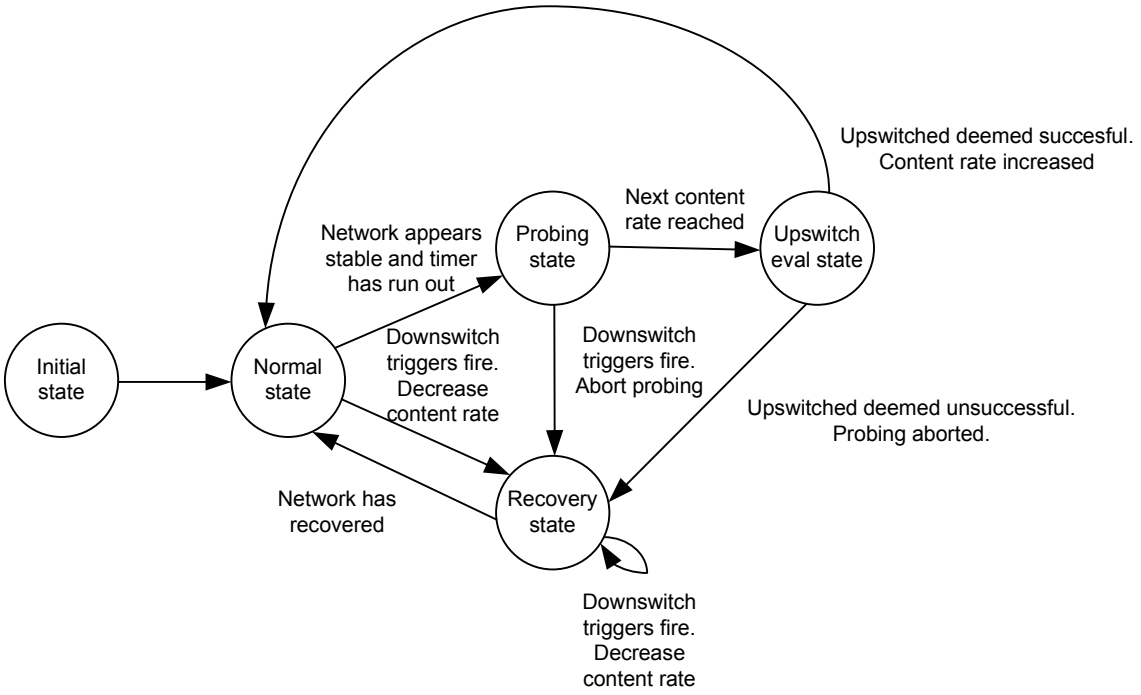

Figure 7.3: Algorithm state machine.

7.2.1.1 Initial State

As the name implies, the algorithm starts in this state. The purpose of this state is to gather information about the network such as average the round-trip delay and NBMT (Network Buffer Media Time, see chapter 4). These values are useful for the recovery state, so that it can determine when the network has "cooled off". After a configurable amount of time, the algorithm moves into the normal state.

7.2.1.2 Normal State

The algorithm should spend most of its time in this state. In this state, the server streams the media at a constant rate, while monitoring feedback from the client. Should the feedback indicate that the network can not sustain the current media bit rate, a downswitch occurs and the algorithm enters the recovery state. If the transmission currently appears stable, the algorithm remains in this state until a timer runs out. When the timer runs out, the algorithm estimates how much media time is currently stored in the network buffers (NBMT, see chapter 4) and if this is below a certain threshold, the algorithm moves into the probing state. Before the algorithm enter the probing state, the NBMT and the average round-trip delay are sampled and stored. These values are later used in the recovery state, should the probing fail.

7.2.1.3 Probing State

The algorithm divides the difference between the current media bit rate and the next higher media bit rate into a configurable number of steps according to the formula below:

$$increase = (nextMediaBitrate - currentMediaBitrate) / NUMBER\_OF\_STEPS$$

$$probingSteps = list(currentMediaBitrate + i * increase),\ i = 1,2,..,NUMBER\_OF\_STEPS$$

Formula 7.1: Calculation of probing steps

At a given interval (currently set to two seconds), the transmission rate is increased to the next step. During this process the algorithm monitors the round-trip delay, estimated throughput, loss fraction (which always should be 0), and the NBMT. If any of these indicates that the network is unable to sustain this data rate (for instance by a rapidly increasing RTD), probing is aborted and the algorithm enters the recovery state. If the transmission rate is successfully increased to the next media rate, the algorithm switches into the upswitch evaluation state.

7.2.1.4 Upswitch Evaluation State

In this state the algorithm evaluates if the network is able to sustain the current transmission rate for a period of time. If the network feedback indicates that the network can not, the transmission rate is reset to the content rate and the algorithm changes to the recovery state. When the evaluation period has passed, the actual upswitch in content bit rate is performed and the algorithm moves into the normal state again.

7.2.1.5 Recovery State

If the algorithm has entered the recovery state, it means that either the media bit rate has just been decreased or that probing has just failed. The purpose of this state is to allow the network buffers to be emptied. If the network feedback indicates that the amount of data in the network buffers is not decreasing, the media bit rate is decreased even more. The algorithm also uses linear regression to estimate the slope of the round-trip delay curve. The formula for estimating the slope is as follows:

$$k = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

Formula 7.2: Calculating best fit using linear regression

Where k is the slope of the round-trip delay curve, x the reception time of the RR, $\bar{x}$ the average value of x for all samples, y the current round-trip delay, and $\bar{y}$ the average round-trip delay. Should the slope indicate that the round-trip delay is still rapidly increasing; the media rate is once again downshifted.

If the round-trip delay and NMBT is nearing the values sampled and stored in the normal state, the network is considered to have cooled down and consequently the algorithm moves into the normal state. Since either a downswitch took place or a probing attempt failed, the timer value used to indicate when the probing state should be entered is increased. This back off is used to avoid a ping-ponging behavior.

7.2.2 The Algorithm in Operation

Figure 7.4 shows how the algorithm defers an upswitch of the media bit rate until the network is able to sustain this rate. The session was run over GPRS and the available media available were 20, 40, and 60 kbps. The audio was encoded at a fixed rate, while the bit rate of the video was variable (this is the reason for the very small initial video bit rate, since almost all of the available bit rate was allocated to audio).

Figure 7.4 shows how the transmission rate is gradually increased and how probing is aborted when the round-trip delay is escalating (indicating that the network buffers are filling up).

Probing is then again tried when the network has cooled off. After about 100 seconds, the algorithm successfully upswitches the media bit rate. The number of probing steps was set to five and the evaluation period set to five seconds.

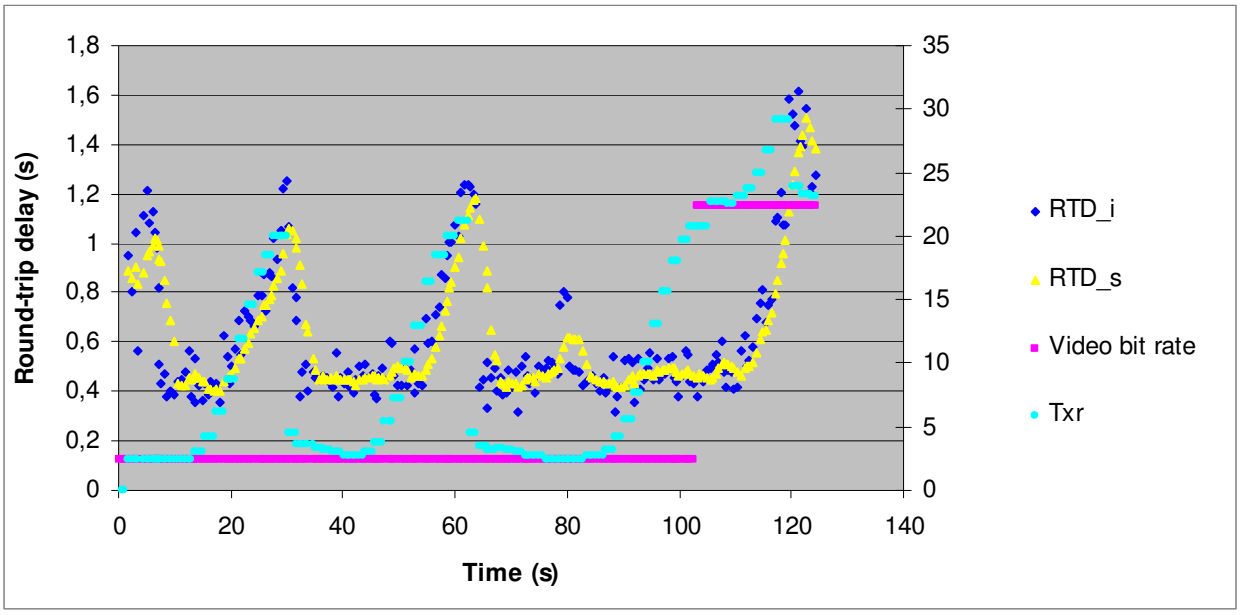

Figure 7.4: Algorithm run over GPRS

Figure 7.5 shows an example of the algorithm running over the REDWINE/PortwineLite network emulator. The bandwidth was initially set to 384 kbps, then reduced to 128 kbps after 60 seconds and restored back to 384 kbps after 120 seconds. The number of probing steps was set to five and the evaluation period was set to five seconds. As shown in the figure, the decrease in bandwidth causes a huge spike in the round-trip delay (about 2.5 seconds). Because of this huge spike, a lot of media is now stored in the network buffers (also about 2.5 seconds). This means that there is a real danger of client buffer underrun and the media bit rate is therefore reduced to the lowest rate in order to drain the network buffer faster. The algorithm then slowly increases the media bit rate until the highest quality is reached at the end.

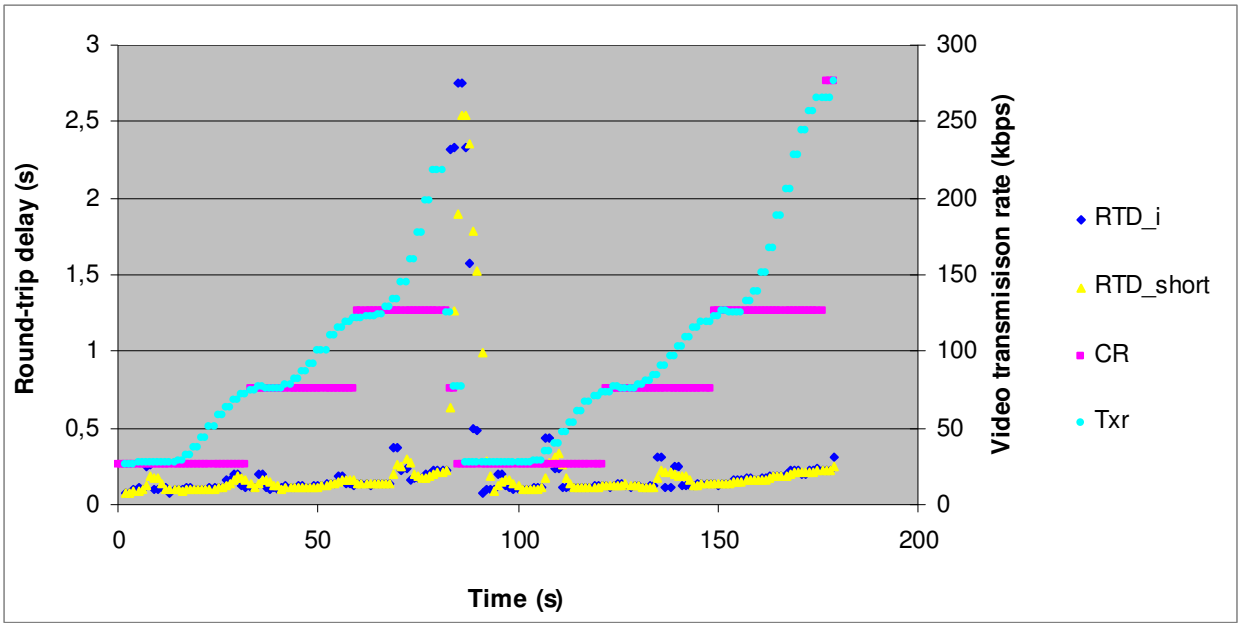

Figure 7.5: Algorithm run over REDWINE. Bandwidth reduced from 384 kbps to 128 after 80 seconds and restored after 100 seconds.

7.3 Testing of the algorithm

The time it takes for the algorithm to switch up to a higher media bit rate, when the bandwidth has increased, depends upon the settings of the algorithm. The time between entering the probing state and switching up the bit rate is the number of probing steps multiplied by time spent in each step plus the evaluation period. In the graphs shown in the previous section, five probing steps were used and the period between each increase was two seconds. The evaluation period was five seconds. These periods means that the media bit rate will be increased 5*2+5=15 seconds after the probing state was entered.

The time to switch up also depends on which state the algorithm is in and how many times it has failed to switch up. The algorithm only enters the probing state through the normal state, so before probing may commence, the algorithm must enter the normal state. The algorithm will enter the normal state from the initial state and upswitch after configurable time periods. From the recovery state, the algorithm will enter the normal state once the network has cooled off. The algorithm will enter the probing state from the normal state after a given time period (assuming that network is stable, but since we are considering the case for increased bandwidth, by definition the network is stable). This time period is increased every time the algorithm leaves the recovery state and is reset on a successful upswitch. Thus the time period depends upon how many downswitches and failed probing attempts have taken place since the last upswitch. In order to avoid this value growing too large, a maximum value is defined. Since all of these parameters, except the time for the network to cool down (which as shown in Figure 7.4 and Figure 7.5 is just a few seconds), are known, the maximum time to switch up the bit rate n levels can be expressed as:

$$t_{\max\_upswitch\_time} = \max(t_{recovery}, T_{eval}, T_{inital}) + T_{\max\_period\_until\_probing} + n \cdot (NUMBER\_OF\_PROBING\_STEPS \cdot T_{time\_between\_increases} + T_{eval})$$

In the same way the minimum time[2] to switch up the bit rate n steps can be expressed as:

$$t_{\min\_upswitch\_time} = n \cdot (NUMBER\_OF\_PROBING\_STEPS \cdot T_{time\_between\_increases} + T_{eval})$$

Setting the number of probing steps to one and the evaluation period to zero is equivalent of immediately switching the bit rate.

Another parameter that affects the speed of the upswitch is the fact that the state machine is only called upon the receipt of a receiver report. This means that even though the algorithm should change state (for instance due to a timer expiring), it will not happen until the next receiver report.

By modifying these parameters, the time to switch up the bit rate can be controlled. It is a trade-off between the speed of the upswitch and the risk of "false" upswitches (i.e. switching up to a rate higher than supported by the network). In this section the algorithm is tested with different settings over different RAN types and different receiver report frequencies.

In test cases 1, 2, and 3 the algorithm is tested over the REDWINE/PortwineLite network emulator. PortwineLite was configured to emulate a 128/64 kbps bearer with 1% BLER. The media bit rates available were 50, 100, and 200 kbps and receiver reports were sent every 4 seconds. The server logged the time for each bit rate switch. Since the bandwidth is 128 kbps, the best bit rate available is 100 kbps. In these tests, the number of probing steps was varied to see which number of steps causes the algorithm to spend the most time at 100 kbps and how many switches were performed. Having one probing state means that the transmission rate is immediately increased (by the means of bit stuffing) to equal the next media rate. When the evaluation period has passed, the actual media bit rate will be increased. The algorithm started at the lowest bit rate and in the initial state. For each test case, ten measurements of three minutes duration were conducted.

7.3.1 Test Case 1

Media bitrates:	50, 100, 200 kbps
Network:	REDWINE
Client:	Ericsson Research's rtspclient
RR interval:	1 RR every 4 seconds
Probing steps:	*1*
Evaluation period:	5
Time between increases:	2
Duration:	3 minutes

Table 7.1: Test case 1

	50 kbps	100 kbps	200 kbps	Upswitches	Downswitches
Run 1	120.00 s	60.00 s	0 s	3	3
Run 2	128.00 s	52.00 s	0 s	3	3
Run 3	131.98 s	48.02 s	0 s	3	3

[2] The minimum time for upswitch will occur when the increase of bandwidth happens at the same that the algorithm enters the probing state.

Run 4	127.98 s	52.02 s	0 s	3	3
Run 5	124.00 s	56.00 s	0 s	3	3
Run 6	123.99 s	56.01 s	0 s	3	3
Run 7	124.05 s	55.95 s	0 s	3	3
Run 8	120.02 s	59.98 s	0 s	3	3
Run 9	119.97 s	60.03 s	0 s	3	3
Run 10	128.02 s	51.98 s	0 s	3	3
Average	***124.80 s***	***55.20 s***	***0 s***	***3***	***3***

7.3.2 Test Case 2

Media bitrates:	50, 100, 200 kbps
Network:	REDWINE, 128/64 kbps, 1% BLER
Client:	Ericsson Research's rtspclient
RR interval:	1 RR every 4 seconds
Probing steps:	*2*
Evaluation period:	5
Time between increases:	2
Duration:	3 minutes

Table 7.2: Test case 2

	50 kbps	**100 kbps**	**200 kbps**	**Upswitches**	**Downswitches**
Run 1	76.00 s	104.00 s	0 s	2	1
Run 2	76.00 s	104.00 s	0 s	1	1
Run 3	107.98 s	72 .02 s	0 s	3	3
Run 4	104.00 s	76.00 s	0 s	2	2
Run 5	104.02 s	73.98 s	0 s	3	2
Run 6	32.03 s	147.97 s	0 s	1	0
Run 7	96.00 s	84.00 s	0 s	1	1
Run 8	35.99 s	144.01 s	0 s	1	1
Run 9	68.00 s	112.00 s	0 s	2	1
Run 10	95.98 s	84.02 s	0 s	1	1
Average	***79.60 s***	***100.02 s***	***0 s***	***1.7***	***1.3***

7.3.3 Test Case 3

Media bitrates:	50, 100, 200 kbps
Network:	REDWINE, 128/64 kbps, 1% BLER
Client:	Ericsson Research's rtspclient
RR interval:	1 RR every 4 seconds
Probing steps:	*5*
Evaluation period:	5
Time between increases:	2
Duration:	3 minutes

Table 7.3: Test case 3

	50 kbps	100 kbps	200 kbps	Upswitches	Downswitches
Run 1	40.00 s	140.00 s	0 s	1	0
Run 2	43.99 s	136.01 s	0 s	1	0
Run 3	44.02 s	135.98 s	0 s	1	0
Run 4	40.00 s	140.00 s	0 s	1	0
Run 5	44.00 s	136.00 s	0 s	1	0
Run 6	40.00 s	140.00 s	0 s	1	0
Run 7	40.02 s	139.98 s	0 s	1	0
Run 8	40.02 s	139.98 s	0 s	1	0
Run 9	43.99 s	136.01 s	0 s	1	0
Run 10	40.03 s	139.97 s	0 s	1	0
Average	***41.61 s***	***138.39 s***	***0 s***	***1***	***0***

These tests clearly show that increasing the number of probing steps provides a big performance boost. With a small number of probing steps, the algorithm will more quickly make the initial upswitch from 50 kbps to 100 kbps. However, a transmission rate of 200 kbps is reached more quickly. This causes a problem, because by the time the algorithm realizes that 200 kbps is unsuitable, so much data is stored in the network buffer that the algorithm is forced to switch down to an even lower rate in order to avoid packet losses.

When the transmission rate is increased more slowly, the algorithm is able to detect that the bandwidth is insufficient earlier and will thus neither switch up the bit rate nor be forced to make an additional downswitch in bit rate. The downside of this is of course that it takes longer to make the initial upswitch.

Test cases 4, 5, and 6 investigates how the results are affected if the receiver report frequency is increased. In these tests, a receiver report is sent every second.

7.3.4 Test Case 4

Media bitrates:	50, 100, 200 kbps
Network:	REDWINE, 128/64 kbps, 1% BLER
Client:	Ericsson Research's rtspclient
RR interval:	1 RR per second.
Probing steps:	*1*
Evaluation period:	5
Time between increases:	2
Duration:	3 minutes

Table 7.4: Test case 4

	50 kbps	100 kbps	200 kbps	Upswitches	Downswitches
Run 1	14.00 s	166.00 s	0 s	1	0
Run 2	15.00 s	165.00 s	0 s	1	0
Run 3	45.00 s	135.00 s	0 s	2	1
Run 4	54.97 s	125.03 s	0 s	2	1
Run 5	14.02 s	165.98 s	0 s	1	0
Run 6	15.98 s	164.02 s	0 s	1	0
Run 7	15.00 s	165.00 s	0 s	1	0
Run 8	67.06 s	112.94 s	0 s	2	1
Run 9	65.00 s	115.00 s	0 s	2	1
Run 10	15.03 s	164.97 s	0 s	1	0
Average	***32.11 s***	***147.98 s***	***0 s***	***1.4***	***0.4***

7.3.5 Test Case 5

Media bitrates:	50, 100, 200 kbps
Network:	REDWINE, 128/64 kbps, 1% BLER
Client:	Ericsson Research's rtspclient
RR interval:	1 RR per second
Probing steps:	*2*
Evaluation period:	5
Time between increases:	2
Duration:	3 minutes

Table 7.5: Test case 5

	50 kbps	100 kbps	200 kbps	Upswitches	Downswitches
Run 1	18.00 s	162.00 s	0 s	1	0
Run 2	50.01 s	129.99 s	0 s	2	1
Run 3	18.00 s	162.00 s	0 s	1	0
Run 4	16.00 s	164.00 s	0 s	1	0
Run 5	48.95 s	131.05 s	0 s	1	1
Run 6	17.00 s	163.00 s	0 s	1	0
Run 7	71.98 s	118.02 s	0 s	2	1
Run 8	51.04 s	128.96 s	0 s	2	1
Run 9	18.00 s	162.00 s	0 s	1	0
Run 10	17.02 s	162.98 s	0 s	1	0
Average	***32.60 s***	***148.40 s***	***0 s***	***1.3***	***0.4***

7.3.6 Test Case 6

Media bitrates:	50, 100, 200 kbps
Network:	REDWINE, 128/64 kbps, 1% BLER
Client:	Ericsson Research's rtspclient
RR interval:	1 RR per second
Probing steps:	*5*
Evaluation period:	5
Time between increases:	2
Duration:	3 minutes

Table 7.6: Test case 5

	50 kbps	100 kbps	200 kbps	Upswitches	Downswitches
Run 1	26.00 s	154.00 s	0 s	1	0
Run 2	26.03 s	153.96 s	0 s	1	0
Run 3	26.02 s	153.98 s	0 s	1	0
Run 4	25.03 s	154.97 s	0 s	1	0
Run 5	26.02 s	153.98 s	0 s	1	0
Run 6	25.98 s	154.02 s	0 s	1	0
Run 7	26.01 s	153.99 s	0 s	1	0
Run 8	25.01 s	154.99 s	0 s	1	0
Run 9	24.03 s	155.97 s	0 s	1	0
Run 10	25.01 s	154.99 s	0 s	1	0
Average	***25.51 s***	***154.49 s***	***0 s***	***1***	***0***

The tests show that when the receiver report frequency is increased, the number of probing steps used does not have as big an impact on the performance as when the frequency is lower. We see that the use of 1 or 2 steps seems to give about the same performance, while five steps provides only a slight improvement. In test 7, the RR frequency was increased even more to see at which frequency the gain of several probing steps diminished.

7.3.7 Test Case 7

Media bitrates:	50, 100, 200 kbps
Network:	REDWINE, 128/64 kbps, 1% BLER
Client:	Ericsson Research's rtspclient
RR interval:	2 RRs per second
Probing steps:	*1*
Evaluation period:	5
Time between increases:	2
Duration:	3 minutes

Table 7.7: Test case 7

	50 kbps	100 kbps	200 kbps	Upswitches	Downswitches
Run 1	12.50 s	167.50 s	0 s	1	0
Run 2	12.50 s	167.50 s	0 s	1	0
Run 3	13.00 s	167.00 s	0 s	1	0
Run 4	12.02 s	167.98 s	0 s	1	0
Run 5	13.00 s	167.00 s	0 s	1	0
Run 6	12.51 s	167.49 s	0 s	1	0
Run 7	12.51 s	167.49 s	0 s	1	0
Run 8	13.01 s	166.99 s	0 s	1	0
Run 9	13.02 s	166.98 s	0 s	1	0
Run 10	12.48 s	167.52 s	0 s	1	0
Average	***12.66 s***	***167.35 s***	***0 s***	***1***	***0***

As the tests show, when the receiver report frequency is about 2 RR's per second, the algorithm is able to detect failed upswitches even with a small number of steps. This indicates that it might be more efficient to let the number of probing steps be dependent upon the receiver report frequency, rather than having a fixed number (as currently used). This will allow the algorithm to avoid filling up the network buffers when the receiver report frequency is low, as well as switching up fast when the reports are sent at higher frequencies.

In the above tests, the media bit rates were chosen based upon the known link bandwidth. In practice, the link bandwidth is unknown (otherwise, it would be unnecessary to perform bit rate adaptation). If one of the bit rate levels is just above the available bandwidth, the algorithm risks performing a "false" upswitch. Test case 8 examines what happens if one of the media rates chosen is just above the available bandwidth.

7.3.8 Test Case 8

Media bitrates:	50, 100, 150 kbps
Network:	REDWINE, 128/64 kbps, 1% BLER
Client:	Ericsson Research's rtspclient
RR interval:	2 RRs per second
Probing steps:	*1*
Evaluation period:	5
Time between increases:	2
Duration:	3 minutes

Table 7.8: Test case 7

	50 kbps	100 kbps	150 kbps	Upswitches	Downswitches
Run 1	12.49 s	167.51 s	0 s	1	0
Run 2	12.50 s	167.50 s	0 s	1	0
Run 3	12.52 s	167.48 s	0 s	1	0
Run 4	30.01 s	149.99 s	0 s	2	1
Run 5	12.52 s	167.48 s	0 s	1	0
Run 6	12.52 s	167.48 s	0 s	1	0
Run 7	12.50 s	167.50 s	0 s	1	0
Run 8	12.50 s	167.50 s	0 s	1	0
Run 9	12.51 s	167.49 s	0 s	1	0
Run 10	12.50 s	167.50 s	0 s	1	0
Average	***14.26 s***	***165.74 s***	***0 s***	***1.1***	***0.1***

As this test shows, the algorithm does not perform an upswitch to 150 kbps, even when only one probing step is used. When the maximum bit rate was set to 130 kbps, the algorithm switched up incorrectly even when ten probing steps was used (the RR frequency was set to 2 RR/s). When the bit rate is this close to the bandwidth (just 2 kbps above), it is probably impossible to avoid a false upswitch. However the number of incorrect upswitches might be decreased by increasing the time until next probing attempt every time the algorithm is unable to sustain a bit rate for longer than a certain threshold value.

Chapter 8: Conclusions and Future Work

The main purpose of this thesis project was to examine if it was possible to improve upswitching mechanism for an adaptive streaming solution by considering the distinct characteristics of the mobile links. The measurements carried out during this thesis project shows that it is not possible to directly conclude the available bandwidth by passively monitoring feedback from the receiver; instead the server must vary its transmission rate.

For pre-encoded content this is not a problem, since there is access to the entire clip and may easily transmit the content faster than the play out rate. For live content, this is more cumbersome, since the server does not have access to the entire stream. This thesis suggests improving this by inserting stuffed data into the media stream, which provides the server with a very fine grained control over the transmission rate. This gives two benefits:

- The next content rate can be checked out before actually increasing the media bit rate and thereby avoiding annoying the user by flip-flopping between different quality levels. Thus failed probing attempts will be hidden from the user and thus improving the user experience.

- Since the transmission rate may be gradually increased, it is possible to detect earlier that the next content rate is unsuitable. Without this, the algorithm might be forced to lower the media bit rate after a failed probing attempt, since so much data has accumulated in the network buffers.

This thesis has also suggested a method of detecting the RAN type by which the client is currently connected. It is shown that 2G technologies are subject to significantly more variance in the round-trip delay. By monitoring the RTD, the algorithm may determine if the client is connected through a 2G or a 3G technology. With this information the algorithm gains information regarding the data rate capabilities of the client and may adjust its thresholds and choice of bit rate (content rate) accordingly.

8.1 Future Work

- The probing algorithm currently uses static parameters. As shown in chapter 7, different conditions require different settings. It would therefore be beneficial to vary the parameters dynamically, depending on, for instance, the difference between two media bit rates, the receiver report frequency, and the RAN technology used.

- Currently, the state machine is only called upon when an RR is received. To make the behavior of the algorithm more predictable, the state machine should also be called upon timer expiry.

- As discussed in chapter 5, it might be possible to detect retransmissions by monitoring the round-trip delay. The number of retransmissions is an indication of the interference level experience over the link and may thus be used to detect changes in radio link quality.

- The accuracy of the RAN detection algorithm may be improved by utilizing the retransmission detection mentioned above

REFERENCES

[1] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson
RTP: A Transport Protocol for Real-Time Applications
Request for Comments (RFC) 3550
Internet Engineering Task Force (IETF)
July 2003
[www] http://www.ietf.org/rfc/rfc3550.
Last access: 2008-07-31

[2] Third Generation Partnership Project (3GPP)
3GPP home page
[www] http://www.3gpp.org
Last access: 2008-10-06

[3] J. Lundberg
A proxy-based application layer rate estimation technique
EAB/TV-03:031
2003-02-11
Ericsson Limited Internal Technical Report

[4] S. Chemiakina
Performance analysis of server-based application level adaption for PS streaming
T/B-02:264
2003-01-26
Ericsson Limited Internal Technical Report

[5] X. Yi
Adaptive Wireless Multimedia Services
Master Thesis
Department of Communication Systems (CoS)
School of Information and Communication Technology (ICT)
Royal Institute of Technology (KTH)
COS/CCS 2006-12
9th May 2006
[www] http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/060511-Xiaokun_Yi-Thesis-with-cover.pdf
Last access: 2008-10-06

[6] A. Tarnowski
An experimental study of algorithms for multimedia streaming – Extending adaptive flow control with content switching.
Master Thesis.
Department of Numerical Analysis and Computer Science
Royal Institute of Technology (KTH)
TRITA-NA-E04nn
2004

[7] J. Rey, D. Leon, A. Miyazaki, V. Varsa, and R. Hakenberg
RTP Retransmission Payload Format
Request for Comments (RFC) 4588
Internet Engineering Task Force (IETF)
July 2006
[www] http://www.ietf.org/rfc/rfc4588
Last access: 2008-10-06

[8] Microsoft
Comparing HTTP Streaming Protocol to RTSP
Microsoft
[www] http://msdn.microsoft.com/en-us/library/bb905764.aspx
Last access: 2008-10-06

[9] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee
Hypertext Transfer Protocol – HTTP/1.1
Request for Comments (RFC) 2616
Internet Engineering Task Force (IETF)
June 1999
[www] http://www.ietf.org/rfc/rfc2616.
Last access: 2008-10-06

[10] N. Anderson
The YouTube effect: HTTP traffic now eclipses P2P
Ars Technica
19th June 2007
[www] http://arstechnica.com/news.ars/post/20070619-the-youtube-effect-http-traffic-now-eclipses-p2p.html
Last access: 2008-10-06

[11] A. Makris and A. Strikos
Daedalus: A media agnostic peer-to-peer architecture for IPTV distribution
Master Thesis
Department of Communication Systems (COS)
School of Information Technology
Royal Institute of Technology (KTH)
COS/CCS 2008-11
June 23, 2008
[www] http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/080624-Makris_and_Strikos-with-cover.pdf
Last access: 2008-11-25

[12] Vidiator
Xenon Streamer Product Sheet
Vidiator
[www] http://www.vidiator.com/PDF_product%20sheets/Xenon%20Streamer.pdf
Last access: 2008-10-06

[13] Mobixell
Mobixell Mobile Webcast – Mobile Video Solutions
Mobixell
[www] http://www.mobixell.com/data/uploads/Mobile%20Webcast%20Brochure%20(Jan%202008).pdf
Last access: 2008-11-11

[14] RealNetworks
Media Servers -> Helix Server v12
Real Networks
[www] http://www.realnetworks.com/info/helixserverv12.html
Last access: 2008-11-11

[15] M. Handley, S. Floyd, J. Padhye, and J. Widmer
TCP Friendly Rate Control (TFRC): Protocol Specification
Request for Comments (RFC) 3448
Internet Engineering Task Force (IETF)
January 2003
[www] http://www.ietf.org/rfc/rfc3448.txt
Last access: 2008-11-11

[16] 3GPP
Transperent end-to-end Packet-switched Streaming Service (PSS): Protocols and codecs (Release 7)
3GPP
TS 26.234 V7.5.0
March 2008
[www] http://www.3gpp.org/ftp/specs/html-info/26234.htm
Last access: 2008-10-06

[17] MacOS Forge
Welcome to Darwin Streaming Server
MacOS Forge
2008-05-16
[www] http://dss.macosforge.org/
Last access: 2008-11-11

[18] J. F. Kurose and K. W. Ross
Computer Networking: A Top-down Approach Featuring the Internet (Third edition)
Addison-Wesley
2005

[19] H. Schulzrinne, A. Rao, and R. Lanphier
Real Time Streaming Protocol (RTSP).
Request for Comments (RFC) 2326
Internet Engineering Task Force (IETF)
April 1998
[www] http://www.ietf.org/rfc/rfc2326.
Last access: 2008-06-31

[20] M. Handley, V. Jacobson, and C. Perkins
SDP: Session Description Protocol.
Request for Comments (RFC) 4566
Internet Engineering Task Force (IETF)
July 2006
[www] http://www.ietf.org/rfc/rfc4566.
Last access: 2008-07-31

[21] C. Perkins
RTP: Audio and Video for the Internet
Addison-Wesley
2003

[22] H. Sinnreich and A. Johnston
Internet Communications Using SIP: Delivering VoIP and Multimedia Services with Session Initiation Protocol, 2nd Edition
Wiley
August 2006

[23] C. Kaufman, R. Perlman, and M. Speciner
Network Security: Private Communication in a Public World (Second edition)
Prentice Hall PTR
2002

[24] H. Schulzrinne and S. Casner
RTP profile for Audio and Video Conferences with Minimal Control.
Request for Comments (RFC) 3551
Internet Engineering Task Force (IETF)
July 2003
[www] http://www.ietf.org/rfc/rfc3551
Last access: 2008-07-31

[25] J. Ott, S. Wenger, N. Sato, C. Burmeister, and J. Rey
Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF).
Request for Comments (RFC) 4585
Internet Engineering Task Force (IETF)
July 2006
[www] http://www.ietf.org/rfc/rfc4585
Last access: 2008-07-31

[26] I. Johansson and M. Westerlund
Support for Reduced-Size RTCP, Opportunities and Consequences.
[www] http://tools.ietf.org/html/draft-ietf-avt-rtcp-non-compound-06
Last access: 2008-07-31

[27] B. A. Forouzan
TCP/IP Protocol Suite, Third Edition
McGraw-Hill
2006

[28] MPEG
MPEG.ORG – MPEG Home
Moving Picture Experts Group
[www] http://www.mpeg.org
Last access: 2008-10-06

[29] Z. Sarker
A Study on Adaptive Real Time Video over LTE
Master Thesis
Department of Computer Science and Electrical Engineering
Luleå University of Technology
October 11, 2007
[www] http://epubl.luth.se/1402-1617/2007/244/LTU-EX-07244-SE.pdf
Last access: 2008-07-31

[30] T. Friedman, R. Caceres, and A. Clark
RTP Control Protocol Extended Reports (RTCP XR).
Request for Comments (RFC) 3611
Internet Engineering Task Force (IETF)
November 2003
[www] http://www.ietf.org/rfc/rfc3611
Last access: 2008-10-06

[31] A. Yoon and R. Banks.
Mobile versus Internet Streaming: Key Challenges.
[www] http://www.vidiator.com/PDF_product%20sheets/Mobile_vs_Internet_Streaming-Key_Challenges.pdf
Last access: 2008-07-31

[32] GSM World
Subscriber connections – Q2 2008
[www] http://www.gsmworld.com/news/statistics/pdf/gsma_stats_q2_08.pdf
Last access: 2008-09-29

[33] Y.Lin and A. Pang
Wireless and Mobile All-IP Networks
Wiley Publishing
2005

[34] Online chapters of [33]
[www] http://liny.csie.nctu.edu.tw/supplementary
Last access: 2008-07-31

[35] H. Derakhshannon
Voice over IP over GPRS
Master Thesis.
Department of Communication Systems
School of Information and Communication Technology.
Royal Institute of Technology (KTH)
April 30, 2008
[www] http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/080501-Homayoun_Derakhshanno-with-cover.pdf
Last access: 2008-07-31

[36] G. Karagiannis
QoS in GPRS
Ericsson Open Report
21 December 2000
[www] http://doc.utwente.nl/18117/1/00000039.pdf
Last access: 2008-10-06

[37] S. Taylor and L. Hettick
Factors affecting GPRS throughput
Network World Convergence Newsletter
24 April 2002
[www] http://www.networkworld.com/newsletters/converg/2002/01318603.html
Last access: 2008-07-31

[38] G. Q. Maguire Jr
Mobile and Wireless Network Architectures
Lecture Notes.
Department of Communication Systems
Royal Institute of Technology (KTH)
2007
[www] http://www.it.kth.se/courses/2G1330/Lectures-2007/MWA-20070117.pdf
Last access: 2008-07-31

[39] J. Montelius
Developing Mobile Applications: Radio Basics
Lecture Notes
School of Information and Communication Technology
Royal Institute of Technology (KTH)
29 January 2008
[www] http://www.isk.kth.se/~ollek/id2216/pdf/02radio_basics.pdf
Last access: 2008-07-31

[40] T. Köhler
EDGE: A Technical Review
Portable Computer and Communications Association (PCCA) meeting
19-20 August 2003
[www] http://www.pcca.org/standards/architecture/edge.pdf
Last access: 2008-07-31

[41] H. Holma and A. Toskala
WCDMA for UMTS: Radio Access For Third Generation Mobile Communications (Third Edition)
John Wiley & Sons Ltd
2004

[42] UMTS Forum
300 million UMTS subscribers; mobile broadband goes global
UMTS Forum
October 6, 2008
[www] http://www.umts-forum.org/content/view/2522/174/
Last access: 2008-11-25

[43] S. Baudet, C. Besset-Bathias, P. Frêne, and N. Giroux
QoS Implementation in UMTS
Alcatel Telecommunications
1st Quarter 2001
[www] http://www1.alcatel-lucent.com/doctypes/articlepaperlibrary/pdf/ATR2001Q1/gb/09baudetgb.pdf
Last access: 2008-10-06

[44] P. Rysavy
Mobile Broadband: EDGE, HSPA and LTE.
3G Americas
September 2006
[www]http://www.3gamericas.org/PDFs/white_papers/2006_Rysavy_Data_Paper_FINAL_09.15.06.pdf
Last access: 2008-07-31

[45] M. Jurvansuu, J. Prokkola, M. Hanski, and P. Perälä
HSDPA Performance in Live Networks
Institute of Electrical and Electronics Engineers (IEEE) Int. Conference on Communications
Pages: 467-471
24-28 June 2007
Digital Object Identifier: 10.1109/ICC.2007.83
[www] http://ieeexplore.ieee.org/iel5/4288670/4288671/04288754.pdf
Last access: 2008-10-06

[46] Motorola
Long Term Evolution (LTE): A Technical Overview
Motorola
2007
[www] http://www.motorola.com/staticfiles/Business/Solutions/Industry%20Solutions/Service%20Providers/Wireless%20Operators/LTE/_Document/Static%20Files/6834_MotDoc_New.pdf
Last access: 2008-11-11

[47]K. Dovstam
Enhanced rate adaptation for MobileTV
Ericsson AB
2007-07-20
EAB-07:034871 Uen
ERICSSON CONFIDENTIAL REPORT

[48] S. Aarnikoivu and J. Winter
Cellular Radio Networks
Presentation Slides
Telecommunications Software and Multimedia Laboratory
Helsinki University of Technology
2005
[www] http://www.tml.tkk.fi/Opinnot/T-110.5120/2005/slides/01a.Cellu-radio-nets.pdf
Last access: 2008-07-31

[49] B. M. Orstad and E. Reizer
End-to-end key performance indicators in cellular networks.
Master Thesis
Faculty of Engineering and Science
Agder University College, 2006.
[www] http://ikt.hia.no/aml/education/IKT590-H2005-master-thesis/E2E-test-tool/rapport_ikt06_g17.pdf
Last access: 2008-07-31

[50] R. Chakravorty, J. Cartright, and I. Patt
Practical Experiences with TCP over GPRS.
Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE
Volume 2, Pages 1678-1682
17-21 November, 2002
Digital Object Identifier: 10.1109/GLOCOM.2002.1188483
[www] http://ieeexplore.ieee.org/Xplore/login.jsp?url=/stamp/stamp.jsp?arnumber=01188483
Last access: 2008-11-26

[51] K. Gerhardsson
Application and GRPS evaluation tool
Master Thesis
Department of Signals and Sensors
Royal Institute of Technology (KTH)
2002-02-07
[www] http://www.ee.kth.se/php/modules/publications/reports/2002/IR-SB-EX-0203.pdf
Last access: 2008-10-08

[52] F. Glifberg
Optimization of End-to-End TCP Performance in Wireless UMTS Network
Master Thesis
Microelectronics and Information Technology
Royal Institute of Technology (KTH)
5th December 2003

COS/CCS 2008-28

www.kth.se