Performance of Multimedia Session Setup Scheme in UMTS Wireless Networks

---

A thesis
submitted in partial fulfilment
of the requirements for the Degree
of
Master of Science
at the
University of Canterbury
by
Sung Jun Yi

University of Canterbury
2004

Examining Committee:

Prof. Dr Krzysztof Pawlikowski, University of Canterbury	(supervisor)
Prof. Dr Harsha Sirisena , University of Canterbury	(co-supervisor)
Prof. Dr Kimmo Raatikainen, University of Helsinki, Finland	(external examiner)

Abstract

The third generation (3G) of mobile network systems is seen as the technology that will bring the full scope of Internet services to the cellular world. In order to fulfill such ambitious goals, 3G specifications have been evolved into cellular networks with full capabilities of understanding the Internet protocols. The two major protocols proposed, Session Initiation Protocol (SIP) and Session Description Protocol (SDP), are the driving forces of such a merger. They are capable of initiating, modifying and terminating multimedia sessions currently available on the Internet.

This project investigates the performances of the multimedia session setup scheme, based on SIP and SDP, over Universal Mobile Telecommunications System (UMTS). Particular attention was paid on the wireless portion of the UMTS, and we developed SIP layer virtual link, namely *User Equipment (UE) to Proxy-Call State Control Function (P-CSCF) link*. The link model used in this project reflects all the lower layer mechanisms specified in the UMTS air interface such as channel codings and interleaving schemes.

A major concern with SIP in 3G networks is to ensure the efficient use of the bandwidth constraints inherent in the wireless medium. Because of the significant size of the SIP messages, 3G terminals can suffer from substantial set-up delays. We investigated the implication of using the compression scheme of SIP messages proposed by Internet Engineering Task Force (IETF), called SigComp. The evaluation was carried out to obtain the savings in terms of delay and multimedia session initiation attempt failures. A novel scheme that can replace the plain text based application protocols for increasing time savings and reducing the load of the SigComp layer is also proposed. The scheme, referred to as Binary Hybrid paradigm of protocol design, is based on binary sequences and plain texts. This paper evaluates the performance of this scheme with contrast to the normal SigComp/SIP. The major focuses of the investigation are how much time delay the schemes are likely to cause and how many session attempt failures are likely to occur over the UMTS wireless link at the

given channel conditions.

Acknowledgments

I would like to thank my supervisors, Professor Krzysztof Pawlikowski and Associated Professor Harsha Sirisena, and Prasan De Silva for their patience and endless supports on the research. I also would like to thank Telecom New Zealand and Foundation for Research Science and Technology (FRST) for the financial support of this project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The third generation (3G) of mobile network systems is seen as the technology that will bring the full scope of Internet services to the cellular world. It is claimed that 3G will provide ubiquitous access to all the successful services provided by the Internet.

In a significant move to merge Internet technologies, recently, both the Third Generation Partnership Project (3GPP, an European collaborative agreement of a number of telecommunication standard bodies) and Third Generation Partnership Project 2 (3GPP2, a North-American and Asian collaborative agreement of a number of telecommunication standard bodies) have decided to adopt and deploy Internet based protocols with their network infrastructures [31],[32]. Adapting the Internet based protocols directly onto the cellular core networks makes 3G networks more compatible with IP-based networks and capable of providing sophisticated services only the Internet is currently able to offer.

The convergence of the Internet (packet switching) and the telephony (circuit switching technology) has been seen as the ultimate goal of network researchers. The two technologies have been following different paths since their births. Telephone network has evolved from the notion of, so called, *"smart network and simple terminal"*, whereas Internet has dictated the notion of *"smart terminal and simple network"* design. The merger of the two technologies means a compromise between both network design paradigms. The current methodology of 3G network design, which has a rather different aim from the normal telephone network, can be regarded as *"smart all around"* approach. The network should retain the complexity in order to provide legacy telephony service (circuit switching) and accommodate the Internet type of traffic (packet switching). In addition, the terminals are required to understand the language (protocols) of the Internet as well as telephone networks, hence

the the smart terminals are inevitable. 3G network, in fact, is the relaying point of the ever evolving network technology. In the near future, next generation networks (known as *Beyond 3G* or *4G* networks) will completely outrun the circuit switching service and there will only be IP-based (packet switching) services available. It is believed that even 3G networks can solely rely on IP if a Voice over IP (VoIP) scheme is successfully implemented [62].

The benefits of using an Internet-like (IP based) structure is that it can bring Internet-type services to cellular network users, such as video conferencing, multi-player interactive games, multimedia messaging, location aware services, etc. Moreover, the evolved form of E-commerce, namely M-commerce (*M* for mobile) will become accessible to cellular network users. According to a British investment bank, Durlacher, the estimated market for of M-commerce in 2003 was 23 billion Euros [40]. The deployment of 3G provides opportunities to broaden the market even further. 3G also explicitly defines the network structure, so that international roaming of the terminals can be done without hassles. This means better services to users and cost reduction to the network operators because they can share the physical resources, such as base stations etc. Because of the revenue generating services of 3G networks, the attraction of these networks to service providers, network operators and users has grown enormously during the past few years. Despite all the benefits 3G can bring, the development of 3G network technology has been relatively slow.

In order to provide all the services that the Internet currently offers, a 3G network must be equipped with the resources the Internet has. These days, wire-line networks can achieve data transfer rates in the order of Gigabits per second. With the aid of such a speed, the services of the Internet have become more sophisticated and complex. Protocols designed for such services follow the basic assumption that the datagrams will be transmitted over such high speed links. Adaptation of such protocols to 3G environment might lead to unexpected and unpleasant results in terms of service qualities because the speeds of wireless links are currently far behind those of wired links. Recently, Japanese telecommunication network operator, *NTT DoCoMo* managed to achieve the speed of few hundreds of kilobits per second, however only for the down-link channel (base station to the terminal) and was the peak rate rather than the average rate; see [59] for details. Because of the restrictions on the signal transmission power of mobile terminals, the up-link (terminal to base station) channel can theoretically provide only half of the data transfer rate of the down-link and often in practice even lower [43].

During multimedia sessions, data can be compactly “trimmed”, in order to give a user

the illusion that the wireless user is using Internet-like broadband services. For example, reducing the voice data samples to some degree and reducing the number of frames per second can reduce overall loadings of the network while minimizing the degradation of Quality of Service (QoS) expected. Since most cellular network devices are equipped with relatively small display areas, fewer frames per second and a smaller size image may cause still acceptable degradation of QoS. A more serious problem occurs during the initiation phase of the multimedia session, rather than during the session itself. This phase, defined in this thesis as the *post-dialing period*[1], is the elapsed time from when user finishes dialing (or presses *send* or *enter* button on the cellular terminal) to the time when the user receives the final feedback from the callee (the ring-tone sound in the case of normal telephony service), which confirms that all necessary resources for the session have been reserved. An excessive length of period can cause serious problems to users who are used to the fast responding land-line telephony and broadband Internet services.

Work by Curcio and Lundan [15] revealed how the 3G multimedia setup time can be significantly affected by the size of the data transfer rate. However their work is far from complete. The control (session setup) messages they used are much smaller than what is suggested by 3GPP [2], and their work does not involved a performance study of the proposed compression scheme called SigComp. The problematic data transfer speed of 3G networks, the multimedia control protocols that are essentially designed for broadband links, and incomplete investigation of the multimedia session setup delay have motivated this project to investigate the performance of the currently proposed multimedia session setup scheme in more detail. The goal of the project is to formulate the network requirements for satisfying Quality of Service (QoS) requirements for users.

The rest of the thesis is organized as follows.. Chapter 2 introduces the major issues of the 3G mobile network with an emphasizes on the error correction and detection mechanisms defined for UMTS networks. Moreover, how Session Initiation Protocol (SIP) is used to initiate the multimedia sessions over 3G network is throughly introduced. Chapter 3 surveys various wireless channel modeling techniques widely used in modern wireless telecommunications. Since the major goal of this project is evaluate the performance of multimedia session setup scheme[2] when it is deployed onto UMTS network, the control

[1]The term, "post-dialing period" used in [15] for the same purpose

[2]This scheme is currently based on the control message exchanges among terminals and the network entities using SIP and SDP protocols.

message[3] level link model is required. The main contents of Chapter 3 is to discuss and describe the selected model for the control message level link model. Chapter 4 introduces the compression scheme that is employed by two network nodes that share the narrow bandwidth link (i.e., low data transfer rate link). The scheme called SigComp has been proposed by Internet Engineering Task Force (IETF) [22], [45] to reduce the size of control messages which are expected to travel through slow wireless link[4] of UMTS. Since the time to complete a data transfer depends heavily on the size of datagram, this scheme is expected to significantly reduce the overall control message exchange delay. This chapter addresses the functionalities and implementation issues of the scheme. The performance of the scheme, in terms of the size reduction, is also addressed in this chapter. Chapter 5 introduces a novel approach to enhance the control protocols used during multimedia session setup phase by reducing the size of their datagrams. This approach of redefining protocols is expected to reduce the workload of the network nodes, which share low data transfer rate links. The comparison of sizes between the original protocols and the redefined protocols are include in this chapter. The chapter concludes with the summary of the benefits of the novel approach and its disadvantages. Chapter 6 describes the experimental setup used in this project for investigating the performance of the multimedia session setup scheme used within 3G mobile networks. The setup contains the virtual link defined in Chapter 3 and the network entities which are programmed to act as entities addressed in Chapter 2. The control messages used for this project with this experimental setup is one that is shown as a example in [2][5] and the redefined version of control messages discussed in Chapter 5. The control messages were compressed using the compression scheme discussed in Chapter 4. All the outcomes of the experiments are included and discussed in Chapter 7. The final chapter, Chapter 8, summarizes the findings and proposes the future works. In this thesis, unless otherwise specified, the term *3G mobile network* refers to as *UMTS network*.

[3]The control messages initiate, modify and terminate multimedia session. SIP is a example of such control message

[4]slow link = link with low data transfer rate

[5]SIP and SDP based control message exchange

Chapter 2

Introduction of 3G Mobile Network Technology

This chapter provides a brief background on the 3G network infrastructure, especially focusing on multimedia support and error correction/detection schemes.. More details about such IP based multimedia supporting infrastructures can be found in [2], [3], [4], [7], [8] and [62].

The two standardization bodies, 3GPP and 3GPP2, have slightly different specifications on their infrastructures. However, since the specifications of 3GPP2 are not yet fully matured, compared to those of 3GPP, especially for the core network structures that support IP-based multimedia sessions, this project closely follows the description of IP Multimedia Subsystem (IMS) from Release 5 (*R5*) of 3GPP specification, in which IMS has been firstly introduced and defined. The wireless channel specification also follows the most recent standards of 3GPP.

Since radio access over wireless link plays the vital role in the performance evaluation of the multimedia initiation scheme, this chapter first briefly discusses the wireless interface of UMTS then the description of IMS is included.

Figure 2.1 shows the simplified model of *R5* UMTS networks according to the specification of 3GPP. It is referenced throughout this chapter to give physical view of the network infrastructure.

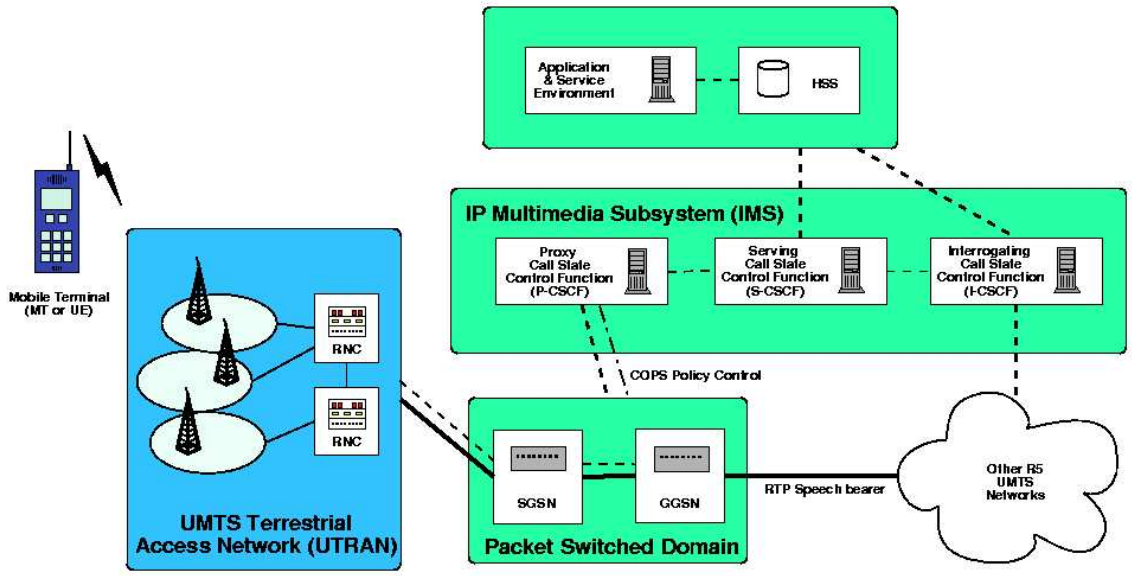

Figure 2.1: UMTS Release 5 structure

2.1 Radio Access Network of 3G

For detailed information on 3G wireless channel, reader is referred to the specifications such as *TS 25.213* [5] and *TS 25.223* [6]. This section only includes the characteristics, which are essential to model the wireless link for the purpose of this investigation.

The radio access portion of the infrastructure, UMTS Terrestrial Radio Access Network (UTRAN), consists of two major entities; they are Node-B (base station) and RNC (Radio Network Controller); see Figure 2.1. The term *Node-B* is a formal name for the base station used for UMTS network. RNC manages the physical layer functions for proper data transfer and controls Node-Bs. During the communications from a mobile terminal to UTRAN (the link of such direction is referred to as *up-link* and the opposite direction link is referred to as *down-link*), Node-Bs pick up the electromagnetic wave transmitted from the mobile terminal and RNC reassembles the binary interpretation of the electromagnetic wave into a proper bit stream then forwards the stream to the core network structures. The signal received by Node-Bs is generally severely distorted by back-ground noises and channel fading. In order to avoid the loss due to such distortions, RNC performs physical layer routines such as error detection and correction during the up-link communication. During the down-link communication (from UTRAN to a mobile terminal), RNC segments the data into a number

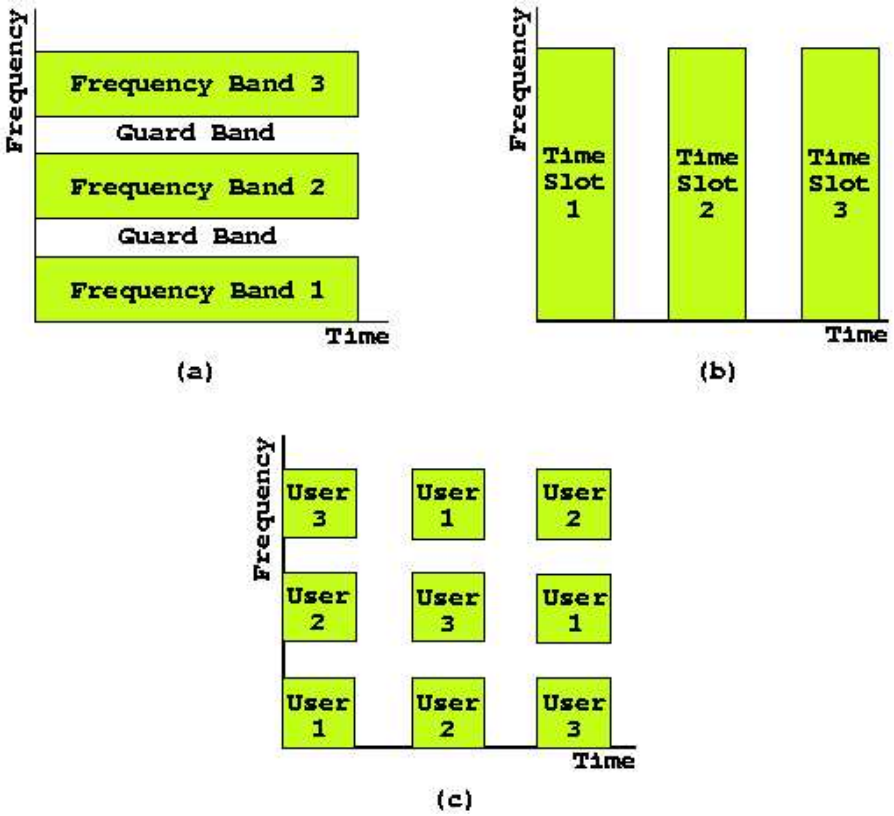

Figure 2.2: Multiple access techniques (a)FDMA (b)TDMA (c)CDMA (adopted from [28])

of transport blocks and at the MAC (Medium Access Control) layer, proper error detection and correction overheads are added. Finally, the Node-B forwards the binary data to the terminal using digital modulation schemes by the unit of radio frames.

The implication of wireless environment and the 3G specific modulation and coding schemes are further discussed in Chapter 3, when the wireless link model used in the experiments is defined.

2.1.1 Multiple Access Techniques

The two standardization bodies (3GPP and 3GPP2) have considered slightly different multiple access specifications for their air interfaces. 3GPP uses W-CDMA (Wideband-Code Division Multiple Access), whereas 3GPP2 uses CDMA-2000 (Code Division Multiple Access 2000). CDMA 1xEv-DO and CDMA 1xEv-DV are the variations of CDMA2000.

There are three types of major multiple access schemes widely used in the wireless com-

munications. Firstly, in FDMA (Frequency Division Multiple Access), the bandwidth is shared by dividing it along the frequency coordinate into disjoint frequency bands, as illustrated in Figure 2.2(a). This approach enables a user to use the channel anytime while the available bandwidth is divided by the number of users. In TDMA (Time Division Multiple Access), the bandwidth is shared by dividing it along the time coordinate into disjoint time slots, as shown in Figure 2.2(b). By assigning a short time duration with the full bandwidth, it can give a user the illusion that the channel is always available. CDMA, shown in Figure 2.2(c) is the hybrid of the two mentioned. It is relatively invulnerable to the distortion of signal occurred during the wireless communication, since the signal is spread like noise and it is only recoverable if a proper code is added to the noise-like signal. Moreover, CDMA schemes employ Frequency and Time Hopping (FH/TH) mechanisms which prevent the interception and the interpretation of the data traveling air interface by intruders [44]. The hopping is based on *pseudo-noise* (PN) sequence, so consecutive data sequence can hardly be decoded by intruders. Because of such reliability and the faster data transfer rate, CDMA based schemes are chosen to be used in the 3G networks. Note that W-CDMA, the multiple access technique adapted by UMTS, does not use FH and TH. W-CDMA, an ITU standard derived from Code-Division Multiple Access (CDMA), is officially known as IMT-2000 direct spread. W-CDMA is a third-generation (3G) mobile wireless technology that promises much higher data speeds to mobile and portable wireless devices than commonly offered in today's market. W-CDMA can support mobile/portable voice, images, data, and video communications. The input signals are digitized and transmitted in coded, spread-spectrum mode over a broad range of frequencies.

2.1.2 Digital Modulation

The standardization bodies, 3GPP and 3GPP2, have decided to use the same modulation schemes that are used in IS-95 (original CDMA) system. For the up-link communications, Binary Phase Shift Keying (BPSK) is used. In BPSK modulation, the phase of the radio frequency carrier is shifted 180 degrees in accordance with a digital bit stream. The digital coding scheme used is called NRZ-M. In NRZ-M, a ”one” causes a phase transition, and a ”zero” does not produce a transition. The receiver performs a differentially coherent detection process, in which the phase of each bit is compared to the phase of the preceding bit. This type of modulation is less efficient but also less susceptible to noise than similar

modulation techniques [1]. The major benefit of BPSK is that it can transmit the data with a relatively small error rate, yet consuming the small amount of power to transmit bit streams. However, BPSK does not utilize the bandwidth of the air channel effectively and efficiently, hence the overall data transfer rate is far from the optimal. Despite its relatively slow data transfer rate, BPSK was chosen to save one of the most scarce sources of a mobile terminal, the electrical power.

For the down-link communications, Quadrature Phase Shift Keying (QPSK) is used. In QPSK, as with BPSK, information carried by the the transmitted signal is contained the phase. However, unlike BPSK it uses a pair of quadrature carriers to utilize the bandwidth [28]. This scheme theoretically has the same average error rate while doubling the data transfer rate if data undergoes a proper encoding such as Gray encoding [28] prior to the modulator. A disadvantage, with contrast to BPSK, is that it uses more power to transmit. Since a Node-B is powered by land-line power supply, power shortage is not an issue. Moreover, in most Internet services, the size of downloading traffic is usually larger than that of the uploading traffic, hence, QPSK seems to be the ideal choice for the down-link communications

Mathematical derivation of bit error rate of BPSK and QPSK can be found in [28] and [44]. Such mathematical representations of the bit error rates for both modulation schemes are used in Chapter 3 to model the wireless link of UMTS, especially the *User Equipment(UE) to Proxy-Call Session Control Function (P-CSCF)link* used in this project. The full description of this link is included in Chapter 3.

2.1.3 Error Detection and Correction

Even though BPSK and QPSK have relatively low average bit error rates, even a single bit error may not be tolerable in certain data communication sessions. 3GPP employed three major mechanisms in order to avoid and correct the error caused during wireless transmission. They are the Cyclic Redundancy Code (CRC), the channel coding and the interleaving.

The transport block, RNC level data unit, is firstly added with CRC, in order to verify the integrity of the received data. The size of CRC overhead can be 24, 16, 12, 8 or 0 bits. The upper layer is solely responsible for choosing the size of CRC depending on the type of data. CRC is used in conjunction with ARQ (Automatic Repeat reQuest) scheme. That is, when erroneous data are detected in the received transport block, ARQ requests the sender to

retransmit that particular transport block. Upon a certain number of consecutive erroneous receptions or losses of the same transport block, ARQ fails. CRC only detects existence of the erroneous bits within the transport bock. If more bit is added to the transport block, it is more likely to find the error within the data while increasing the overall data size. For most delicate data, such as SIP/SDP datagrams for controlling multimedia sessions, a longer CRC overhead is expected to be used, whereas for streaming real time data, using 0 bit CRC is more realistic since there is no time for the retransmission and the user is more tolerable to the errors on voice or video distortions.

After CRC attachments, the data undergoes a channel coding. For control channels, the system uses a convolutional encoder of coding rate $1/2$ and for conversational channels, the coding system has the choice of one of three different encoders. They are two convolutional coders with coding rates of $1/2$ and $1/3$ and one turbo coder of rate $1/3$ [3],[4]. The 3GPP specification recommends the turbo coder to be used if the data transfer rate is guaranteed to be relatively high. Since the multimedia session establishment phase, which this project is focused on, commences prior to the resource reservations, a guaranteed high data transfer rate is unlikely the case. The performance issues of the two convolutional encoders are discussed in Chapter 3.

When the CRC attached transport block is passed into the encoder, it is further segmented into code blocks, and then the encoding begins. For the convolutional encoders the size of each code block can be up to 504 bits and for the turbo coder it can be up to 5114 bits. Figure 2.3 is the graphical illustration of two convolutional coders defined in the 3GPP specifications [3],[4]. Both encoders have constraint length of 9 (1 + the number of delay(memory) entities). At the receiving end, a Viterbi soft-decision decoder is used to correct the error occurred in the air channel, implementing the FEC (Forward Error Correction) scheme.

The last mechanism to provide a reliable data exchange via the air interface is an interleaving scheme. The purpose of this scheme is to put successive bits produced by the encoder in time slots that are widely apart to avoid “burst” of data loss over the wireless environment. If the channel experiences the fading, it lasts for a certain amount of time corrupting the data transmitted during that period. By artificially placing “gaps” between consecutive bits by placing them in several time slots which are not transmitted sequentially, the interleaving avoids the “busts” of data losses or corruptions. The physical layer applies a two-stage (inter frame and intra-frame) bit-level interleaving to mitigate the effect transitory bad channel conditions. Inter-frame interleaving happens on the scales of a Transmission

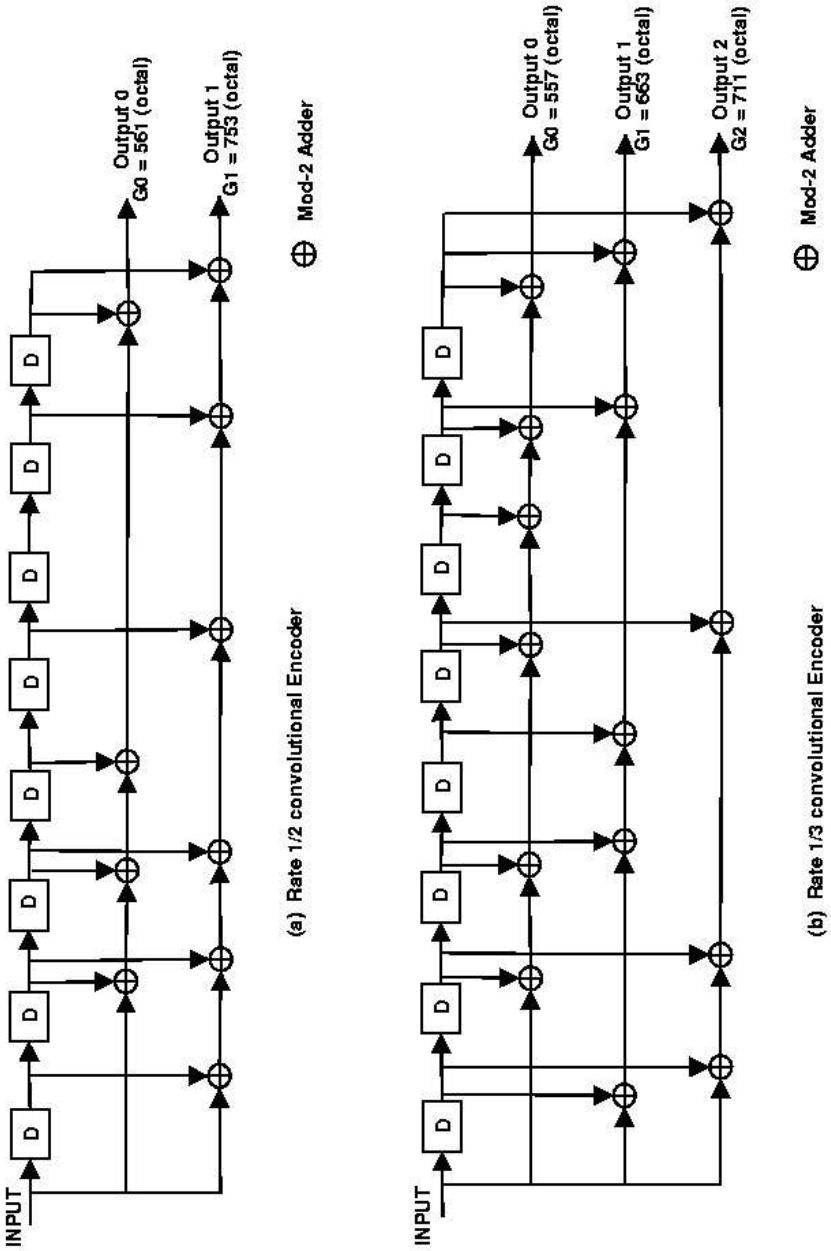

Figure 2.3: Convolutional encoders of rate 1/2 and 1/3

Time Interval (TTI), which has a fixed duration equals to 1, 2, 4, or 8 radio frames (10, 20, 40 or 80 ms respectively). The successive bits in the channel encoder output bit stream that arrives during the same TTI are mapped to non-successive radio frames within the following TTI. If no inter-frame interleaving is performed, that is the span of TTI is one radio frame, the intra-frame interleaving still provides some additional protections against error bursts, by mapping the successive bits in the channel encoder output bit stream to the non-successive time slots within the same radio frame. The choice of proper TTI is a trade-off between packet loss and packet delay. A longer TTI means more delay, but also decreases the probability of a burst of bit errors on the transport channel. Hence the packet loss probability is lower and the effect of transitory bad channel conditions decreases [43]. How large the TTI can be depends on the available delay budget, and how small it can be depends on the tolerance toward the loss of transmitting data. For instance, a voice data stream have much high tolerance of error than SIP/SDP data while the delay budget is strictly limited. The receiver decodes the incoming data. During this phase a few corruptions of the incoming bits are corrected, however it erroneous bit persists, CRC mechanism detects the error after decoding and invokes the ARQ. Ideally, this ARQ repeats until the data received after decoding is completely error free, however in practice due to the time constraints, only a few attempts of the retransmission by ARQ is performed.

2.2 IP Multimedia Subsystem

The IP Multimedia Subsystem (IMS) was firstly introduced and defined in the release 5 (*R5*) of 3GPP specification. Since the release 4 (*R4*) of the specifications, the organization has focused on defining a core structure, which supports IP datagrams without designing and installing expensive and complex gateways, which translate IP datagrams into the cellular network specific languages or vice versa. The approach of R5 is to define a new core system, called IMS, within the 3G network, which accompanies all the necessary IP based network functionalities. In addition, R5 defines the link between UTRAN (UMTS Radio Access Network) with other core network with AAL5 based ATM network. Usage of 16 time-slots per one radio frame allows fast and effective lower level data transfer from the radio frame into the ATM cell. Major considerations have been taken place to design UTRAN and IMS, so that the operation of IMS, basically IP level packet operations, are completely independent of the functions of UTRAN [62].

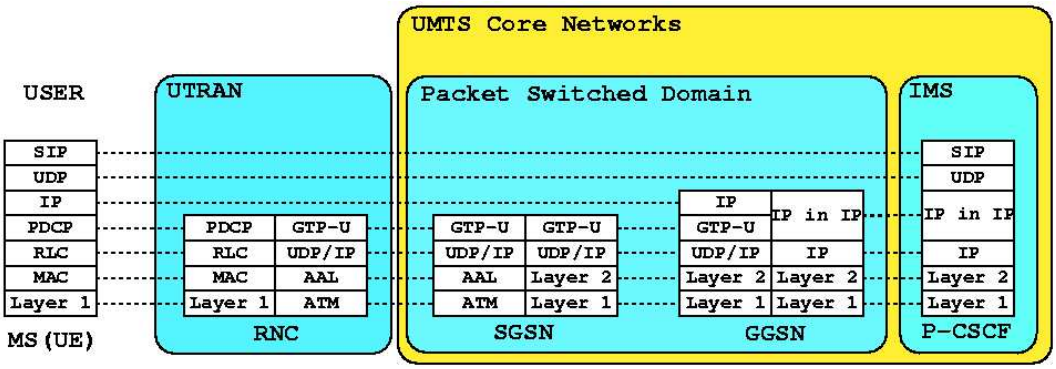

Figure 2.4: Protocol Stack for User Plane

The real purpose of R5 is to enable network operators to offer new services, such as multimedia conferences, multi-player interaction games etc. As R5 is known to support all the services IP network supports, normal voice calls can be dealt as VoIP sessions. However, the voice data can still be carried by the circuit switched domain since UMTS supports the circuit switching as well.

In order to introduce the IP-based network structure to the cellular network with the full functionalities, 3GPP needed to adapt the protocol stacks designed for Internet [31], [32]. Usage of the Internet based protocol stacks enables the 3G cellular network to offer more sophisticated services, such as location aware services. The main protocols included in the specification to offer such services and the compatibility with Internet, are Session Initiation Protocol (SIP) and Session Description Protocol (SDP). SIP is an application layer protocol (c.f., SIP is sometimes regarded as a presentation layer protocol since it usually does not have direct involvement with a user and usually caries other form of datagrams, for instance images and SDP messages), which is capable of initiating, modifying and terminating any types of multimedia sessions [49]. When Internet Engineering Task Force (IETF) was developing the protocol, its aim was to develop a IP based protocol which can replace the H.323 [33] based multimedia system proposed by International Telecommunication Union (ITU). H.323 defines the overall structures of the system, which offers packetized multimedia data exchanges between users. However it tends to be complicated and a lot of consideration must be taken place if the data traverses IP based networks since compromises at the gateway can happen. The approach taken by IETF was simply to define an application level protocol which carries all the relevant information of the multimedia session in a plain text

form. SIP generally carries the information of the user including authentication information and network information, whereas SDP (Session Description Protocol) carries the session related information, such as a session name and usable codecs. The approaches of the two major organizations, ITU and IETF, implementing the multimedia system vary significantly. It was also created the issue of comparability of networks using H.323 and SIP (see [24] and [60] for instance). Despite that 3GPP resides under the umbrella of ITU, it decided to use the SIP-based multimedia session control structure [31] rather than H.323-based, in order to be in harmony with Internet.

The structure of IMS is to follow the structure of SIP-based network, hence all the network entities resemble the entities that the SIP specification defines.

2.2.1 Session Initiation Protocol (SIPv2)

The name SIP firstly came from the protocol called Session Invitation Protocol (SIPv1) proposed in 1996. The basic functionalities of SIPv1 are the same as those of SIPv2. However, the role of SIPv1 stops once the session is established, whereas the roles of SIPv2 includes modifying and terminating the ongoing sessions. The major contribution of SIPv1 was the concept of a registration server for the multimedia conferences. The idea evolved into a registration server, a proxy server and a redirect server of current SIPv2 specification. In the similar time frame, another protocol Simple Conference Invitation Protocol (SCIP) was proposed. The contribution of the protocol was that it used an e-mail address to identify the user, which later evolved into SIPv2 addressing (e.g., SIP URI, e-mail address etc). In 1996, at the 35th IETF meeting both SIPv1 and SCIP were presented and resultantly both protocols had merged together to form SIPv2. SIPv2 has all the good ingredients from the two protocols. It can run on any types of transport protocols (UDP or TCP) and can be called anytime during the multimedia session to modify the parameters of the session and even terminate the session. Thereafter, in this thesis, the term *SIP* refers to *SIPv2*.

In order to use SIP, a user must be registered into the system. During the registration phase, the registration server, *Registrar*, stores all the information about a user including the user's location, especially the terminal the user is currently logged onto. The registration provides a personal mobility. The personal mobility allows a user to use multiple terminals placed in different locations with only one identity (SIP identifier). The advantage of such an approach is that the caller does not need to spend any effort for locating the callee. As

long as the caller initiates multimedia session to the callee with the callee's unique identifier, SIP URI, SIP server finds the current location where the callee is currently logged on and then delivers the session invitation to that location.

The network, which employs SIP as a multimedia control protocol, requires several logical entities, which control the SIP message flows and store appropriate data within the messages. According to the SIP specification [49], a caller (session initiating end) is defined as UAC (User Agent Client) and a callee is referred to as UAS (User Agent Server). There is no restrictions on the number of callees, that is, an invitation can be sent to multiple callees. The terms, UAC and UAS, emphasize that the SIP message flows are similar to the server-client communication. In 3G networks, UAC and UAS can represent as application software installed within the users' mobile terminals or applications running on some kind of service servers. The specification also defines three types of SIP specific servers, namely *registrar*, *proxy server* and *redirect server*. The *registrar*, which supports a personal mobility, stores all the registration information of users and keeps the tracks of the users' availabilities. In order words, the *registrar* is a huge database, which stores a bunch of SIP addresses (SIP URI) indicating where specific users can be contacted. The *redirect server* informs UAC which alternative address should be tried if it receives a request and attempts to reach an address currently unavailable. The availability of the address is confirmed by consulting the *registrar*. The *proxy server*, as the name implies, sends and receives the SIP messages on behalf of UAS and UAC. It isolates UAC and UAS from the outer networks so that some form of security scheme can be implemented, and it can send messages to the multiple number of addresses which are currently registered as available. Hence, it can reduce the load of UAC and possibly results in a more reliable session establishment. The roles of the *redirect server* and the *proxy server* are directly related to the information from the *registrar*. That is why, in practice, the *registrar* is implemented within the same physical structure of the redirect server and/or proxy server.

2.2.2 SIP and IMS

3GPP defines three types of Call Session Control Functions (CSCF) residing within the IP Multimedia Subsystem (IMS). They control all the multimedia sessions of a 3G network [7][62] (see Figure 2.1 and 2.4). CSCFs are regarded as SIP proxies and/or redirect servers, according to their roles in the particular circumstances. Proxy CSCF (P-CSCF) is the point

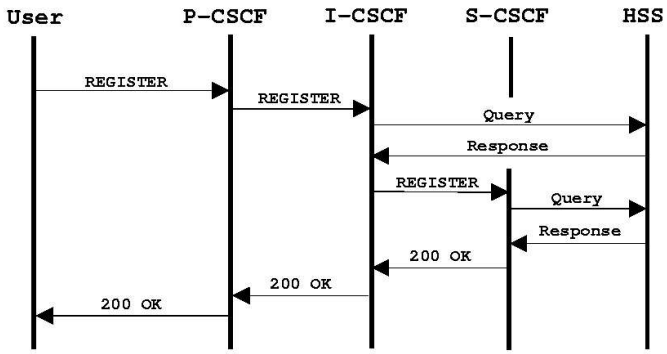

Figure 2.5: Normal registration in 3G network

of contact of a terminal to the network at the application layer level and it is regarded as an outbound proxy (a proxy server attached to UAC side). Serving CSCF (S-CSCF) provides services to users. When a terminal REGISTERs, it is associated with an S-CSCF, which provides the terminal with services that the user is subscribed to. Both incoming and outgoing sessions will traverse the S-CSCF associated with the terminal. This way, the assigned S-CSCF can provide services on both types of sessions. The role of Interrogating CSCF (I-CSCF) is to find the proper S-CSCF for the particular user. For incoming sessions, I-CSCF is the point of contact within the provider's network; it receives requests for the user within its domain and routes them to the proper S-CSCF. Most security measures for incoming data toward the network are to be dealt with this node. For outgoing sessions, I-CSCF receives requests from the user and routes them to the associated S-CSCF. I-CSCF is also regarded as a stateful SIP proxy server. The roles of CSCFs in session setup phase are discussed in more detail in [7], [25] and [52]. During message exchanges between CSCFs, the network must follow appropriate resource reservation mechanisms to ensure the session meets Quality of Service (QoS) parameters specified.

Figure 2.5 and 2.6 illustrate the message exchange sequences during the registration. The figures are deliberately simplified in order to show how IMS entities behave as SIP entities. The aim of such a registration is to assign an S-CSCF to the user. The S-CSCF will be in charge of providing services to the user. Figure 2.5 illustrates the case when the user initiates the registration within the home domain, and Figure 2.6 shows the case when the user initiates the registration while roaming to a foreign or visited domain. The sequences of the message flows are the same in both cases. The REGISTER message is firstly generated

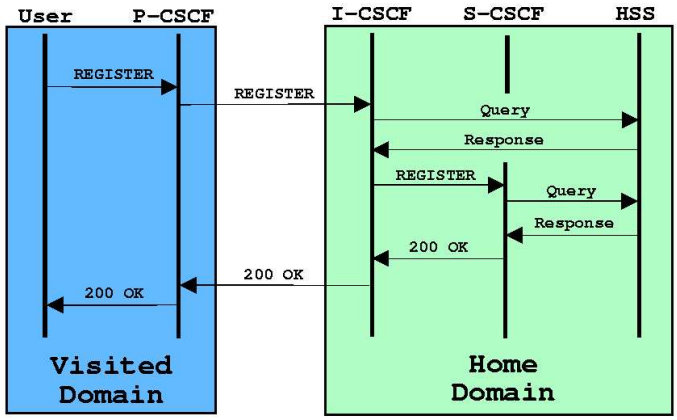

Figure 2.6: Registration in 3G network when user is roaming

by user's terminal and transmitted to UTRAN and RNC forwards the assembled message to attached P-CSCF. Note that the operation of UTRAN is completely invisible to SIP and SDP. The P-CSCF forwards the SIP message on behalf of the mobile terminal to the I-CSCF of the home network of the user. Note that the P-CSCF is the first point of contact at the SIP layer level, hence it can be either P-CSCF of the home network (Figure 2.5) or visited network (Figure 2.6). Upon the reception of the request, the I-CSCF consults HSS (Home Subscriber Server), which is not a SIP entity although its role is similar to SIP registrar: to store the information of the user's subscriptions of the services provided by the service providers or the network operators and the location information. Since HSS is not a SIP entity, the communication between HSS and I-CSCF is not necessary to be based on SIP. By consulting HSS, I-CSCF acquires information about the services, which the user subscribes. Based on the response from HSS, I-CSCF selects an appropriate P-CSCF and forwards REGISTER message. P-CSCF then downloads the information of the user from HSS and tells HSS that it is currently responsible for the user. If every operation goes without error, 200 OK message is generated and it is sent to the user via the same route REGISTER but traversed in the reverse direction. The usage of the same route is a special feature of SIP. It allows any relaying entities to store the information of the session request and the response, which may be required in some circumstances (see [10] and [49] for more detail). The roaming case (see Figure 2.6) follows the same message exchanges procedures. There are mainly two reasons why forwarding REGISTER message to the home network is important. Firstly, it allows a user to access services, which the visited network does not support but the home

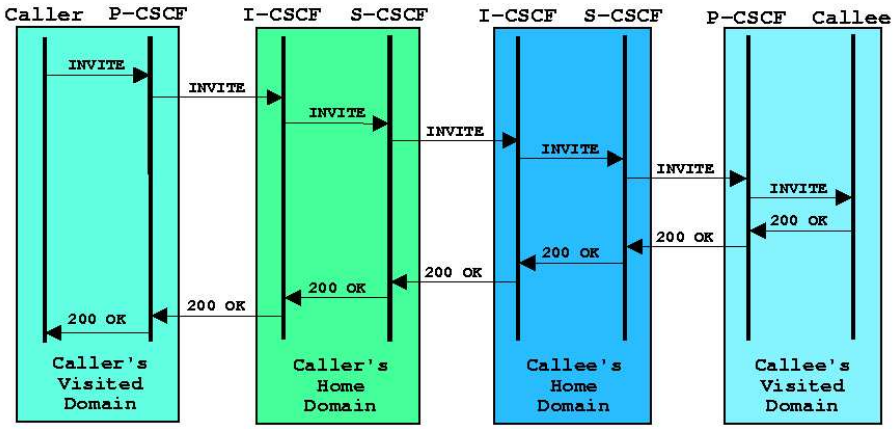

Figure 2.7: Session establishment between roaming users

network does. The S-CSCF of the user's home network is solely responsible for providing the services, which the user subscribes, hence storing the registration information in the HSS of the home network is the obvious solution. Secondly, it provides a personal mobility. Since the HSS of the home network stores all current information about the user, it can provide the right location of the user to the incoming session invitation.

Figure 2.7 is a simplified illustration of how the SIP messages travel when a roaming caller initiates the session with a roaming callee. It is assumed in the figure that the registrations of the roaming users are stored in the HSS of their home domain. Firstly, the P-CSCF of the caller's visited network receives the INVITE request and forwards it to the caller's home network. Such operation is possible because the SIP message carries the information of the user's home domain information. The I-CSCF of caller's home network receives the INVITE message and by consulting HSS, it finds the S-CSCF the caller is assigned to. By looking at the destination address of the SIP message, S-CSCF notices that the callee belongs to the foreign network and sends out the INVITE message to the callee's home network. The I-CSCF receives the INVITE message and consults HSS of this network in order to find the proper S-CSCF, which the callee is assigned to. Since the HSS of the callee's home network knows the location of the callee from the prior registration, it informs the S-CSCF about the address,under which the callee can be contacted. The INVITE message is finally delivered to the callee by the P-CSCF of the callee's visited network. The response of the request follows the same route, which the request traveled.

Chapter 3

Modeling the Link between UE and P-CSCF

As far as SIP and SDP layers are concerned, the first point of contact from the mobile terminal to the currently attached network is the P-CSCF of that network (see Figures 2.1 and 2.4 of the previous chapter). Hence, in order to investigate the performance of the SIP based multimedia session initiation, which is heavily dependent on the lower layer functionalities, the precise modeling of the link between UE and P-CSCF is essential. The link between UE and P-CSCF only exists at the logical level. Despite that the operations of low layers (layers below SIP/SDP) are invisible from the perspective of SIP/SDP layer, their performance severely influences the performance at the SIP/SDP layer level multimedia session control. Since the direct connection between UE and P-CSCF does not exist but only at conceptual level, we call the link model between UE and P-CSCF the virtual link. Upon modeling the virtual link between UE and P-CSCF, the mobile radio channel places a fundamental limitation on the overall performance of the multimedia session initiation scheme. The transmission path between the transmitter and the receiver varies from rather simple line-of-sight (LoS) to one that is obstructed severely by surroundings such as buildings and mountains. Unlike a wire-line link, which is rather predictable and stationary, modeling a wireless channel has been a great research challenge. Because of its extreme randomness, traditionally, statistical models based on the actual measurements of wireless communication systems, have been widely used.

The link model, this chapter is focused on, can be used to evaluates the SIP-based multimedia session initiation scheme and reflects all the characteristics of underlying layers,

which are heavily dependent on wireless channel conditions. The contents of this chapter are closely related to the contents of Chapter 2. This chapter uses bottom-up approach to describe the proposed link model. That is because the link model is heavily affected by the characteristics of the operation and configurations of the lower layers. The proposed link model between UE and P-CSCF is a result from stacking up the characteristics of lower layers which are specific to UMTS links significant to the virtual link between UE and P-CSCF.

3.1 Objectives of the Model

The virtual link model describes the condition of the link between the mobile terminal, UE, and the P-CSCF of the currently attached network. The model state varies depending on the conditions of underlaying layers. For example, ARQ failure leads to an IP packet drop and eventually to a SIP message loss. The delays caused by the physical and data link layers result in postponement of the SIP message delivery. A reasonably accurate model of the underling layers should be adequately translated into the virtual link model.

The requirements for the model follow

1. The model should be described by the numerical variables which represent the characteristics of the link conditions.

2. The model should be implementable on the available computers.

A controllable model is important, since it allows the experiment to be repeated to obtain reproducible results, which is an essential part of the scientific exercise.

Although the virtual link should exhibit both wire-line and wireless characteristics (wireless link between UE and UTRAN and wire-line link between UTRAN and P-CSCF; see Figure 2.1), the loss or corruption of data is mainly happens in the wireless portion of the link. Hence, the loss from the wire-line portion is assumed to be negligible in this project. This assumption is often used in other studies, for instance see [43], since such a event is very rare and seldom introduces the loss of the upper level layer datagram. Moreover, the message exchange delay incurred within wire-line portion is reported to be relatively small (5–10ms) [43], therefore the delay does not significantly contributes to the overall SIP packet delivery delay.

3.2 Physical Layer Model

This section investigates the characteristics of the wireless link in order to identify the distortion level of an electromagnetic signal traveling through the UMTS wireless channel. Traditionally, the wireless model is modeled with Additive White Gaussian Noise (AWGN) as a fundamental noise source, and research has been focused on deriving analytical models of Bit Error Ratio (BER) using various modulation schemes and channel codings. Basically, those studies disregarded the distortion or sudden fluctuation of signal level. Typically, the wireless propagation channel contains objects (particles) which randomly scatter the energy of the transmitted signal. These objects (particles) are referred as scatterers. Scatterers introduce a variety of channel impairments including fading, multi-path delay spread, Doppler spread, attenuation etc., and the inherent background noise (AWGN in general). Hence, the varieties of channel impairments must be taken into account while modeling the wireless channel portion of the physical layer model.

3.2.1 Causes of Loss

The properties of the impairments of a wireless mobile channel are described in statistical forms, and generally they are classified into two categories. Firstly, the propagation model that predicts the mean-signal strength for an arbitrary T-R (transmitter to receiver) distance (typically, several hundreds or thousands of meters) is called the *Large Scale Path Loss* model. The second category, which characterizes rapid fluctuations of the received signal strength over short travel distances (a few wavelengths long) or short time durations (order of seconds), is called the *Small Scale* or *fading* model. In other words, the *large scale Path Loss* model describes the effects due to the T-R distance and the surroundings, whereas the *Small Scale* models the effects caused by the multi-path of signal from the transmitter to the receiver. The overall degradation of the signal as it traverses through wireless channel is caused by the combination of those two [47].

In the modern cellular networks, where the T-R separation is usually less than $1km$ in urban area networks, the *Large Scale Path Loss* is usually assumed to be negligible, because the *Large Scale Path Loss* model assumes the T-R separation of at least $1km$. Moreover, even if the degradation due to this occurs, a sophisticated transmission power control mechanism equipped in modern cellular terminals can compensate for this loss [43]. More severe effects come from the *fading* or *Small Scale fading* model. Depending on the relation be-

tween the signal parameters (such as carrier frequency, bandwidth, symbol period, etc.) and the channel parameters (such as RMS delay spread and Doppler spread), different signals undergo different types of fading. It is the crucial factor of modeling the virtual link, which reflects the properties of the physical layer, to identify which type of fading will dominate the degradation of the signals transmitted by 3G terminals.

According to [3], [4], one radio frame is transmitted over $10ms$ period and it contains 16 time-slots. For the up-link communications (UE to Node-B), each time-slot contains 10×2^k bits and for the down-link communications (Node-B to UE) each time-slot carries $10 \times 2^{k+1}$ bits, where k is an integer between 0 and 6. With these information, the duration (period) of one symbol[1] can be calculated and compared with the RMS delay spread value. Note that the down-link provides twice the data transfer rate for the up-link due to the different digital modulation schemes (see Chapter 2).

3.2.2 Wireless Link Fading Model

Since we assume that the effect of the *Large Scale Path Loss* is negligible, only *Small Scale fading* model was considered. The *Small Scale fading* is further categorized into four different types of fading. The multi-path spread of the signal leads to *flat fading* and *frequency selective fading*, while Doppler spread leads to *fast fading* and *slow fading*. Figure 3.1 shows a tree of the different types of fading [47].

It is a primary concern of this subsection as to which of the different types of the fading the 3G wireless carrier signal is most likely experienced in normal operating conditions.

Multipath Time Delay Spread based Classification

As mentioned earlier, the scatters which traverse different paths introduce the impairment of the signal, when they are received by the receiver. Since each of them are distorted in terms of power and the phase differently, the overall received signal is the collection of those distortions due to the multipath. The key parameter, which determines which type of multipath time delay based fading dominates (see Figure 3.1 (a)), is the RMS delay spread for a given channel condition. The RMS delay spread is the square root of the second central moment

[1] Each digitally modulated symbol of signal can carry one or more bits per symbol depending on the modulation scheme. For example, a BPSK symbol only carries one bit of a radio frame whereas a QPSK symbol carries two bits of a radio frame.

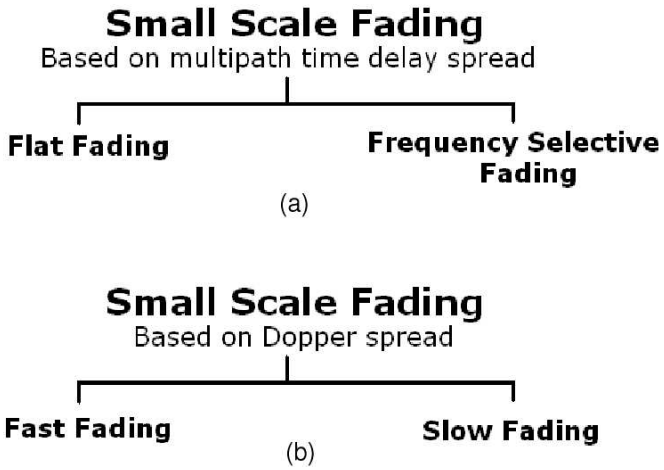

Figure 3.1: Types of small-scale fading depending on (a) multipath time delay spread and (b) Doppler spread

of the power delay profile[2]. In practice, it is measured by temporal or spatial average of consecutive impulse response measurements[3] collected and averaged. Many measurements of RMS delay spread have been recorded and published, varying the carrier frequencies and the surrounding environment (see [12], [13], [17], [48], [50] and [54]) It is experimentally measured by previous researches and found that the values of RMS delay spread are on the order of nanoseconds for indoors and on the order of microseconds for outdoors for typical frequency ranges used for cellular communications.

If the RMS delay spread for a given channel condition is less than the symbol period, it is generally regarded that the signal undergoes the *flat fading*, otherwise it undergoes *frequency selective fading* [47]. In other words, if Equation 3.1 is true, then *flat fading* dominates. Otherwise *frequency selective fading* dominates.

$$RMS\_delay\_spread < Symbol\_period \tag{3.1}$$

In *flat fading*, the multipath structure of the channel is such that the spectral characteristics of the transmitted signal are preserved at the receiver. However, the strength of the

[2]The Derivation of mathematical model of RMS delay spread is out of scope of the thesis and many measurements of RMS delay spread are reported and the measurements were referenced instead of using mathematical model of RMS delay spread when dominating fading is determined.

[3]As the name, impulse response measurement, implies it is done by sending impulse to a particular destination from a source with particular frequency and distance, and measures the received power of the impulse.

received signal changes with time, due to fluctuations in the gain of the channel caused by multipath. When *frequency selective fading* occurs, the received signal includes multiple versions of the transmitted waveform which are attenuated and delayed in time, and hence the received signal is distorted. If this happens the channel induces *inter symbol interference (ISI)*. Due to ISI, if the received signal is viewed from the frequency domain, for instance using spectrum analyzer, certain frequency components have greater gain than others, hence the name *frequency selective fading*.

As mentioned, each radio frame contains 16 time-slots and each contains 10×2^k bits for the up-link and $10 \times 2^{k+1}$ bits for the down-link, where k is an integer between 0 and 6. With this information, we can deduce that each radio frame carries 160×2^k bits for the up-link and $160 \times 2^{k+1}$ bits for the down-link. Since, BPSK transmits one symbol for one bit of radio frame, and each radio frame takes $10ms$ to transmit, the symbol period for up-link channel can be calculated by

$$T_{up-link} = \frac{10 \times 10^{-3}}{160 \times 2^k} \tag{3.2}$$

For the down-link channel, QPSK transmits two bits per one symbol. Hence, the number of symbols transmitted over $10ms$ (time required for one radio frame transmission) is $160 \times 2^{k+1}/2 = 160 \times 2^k$. Therefore, the symbol period for both up-link and the down-link channels can be calculated by Equation 3.2 (i.e., $T_{up-link} = T_{down-link}$).

Varying the value for k, the symbol period ranges from $62.5\mu s$ to $0.977\mu s$. The value of the symbol period are greater than the typical values for the RMS delay spread for indoors and outdoors. Hence, we conclude that the 3G wireless signal carrier exhibits *flat fading* characteristics rather than *frequency selective fading* and assume the virtual link suffers from the *flat fading*.

Doppler Spread based Classification

Since the moving signal source is a subject to the Doppler effect, the impairment due to the speed of the source should be concerned as well. The key parameter which distinguishes the type of Doppler spread based fading is the coherent time. The coherent time is a statistical measure of the time duration over which the channel impulse response is essentially invariant, and quantifies the similarity of the channel response at different time. In other words, coherent time is the time duration over which two received signals have a strong potential for amplitude correlation. The coherent time, T_c can be calculated by [47]

$$T_c = \frac{0.423}{v/\lambda} \tag{3.3}$$

where v is the speed of the signal source in m/s and λ is the wave length of the carrier signal.

If the following inequality holds, then the signal undergoes *fast fading* otherwise it undergoes *slow fading* (see Figure 3.1 (b)).

$$Coherent\_time < Symbol\_period \tag{3.4}$$

Depending on how rapidly the transmitted signal changes as compared to the rate of change of the channel. The rate of the change of the channel is measured by the impulse response of the channel. In other words, if the impulse response changes rapidly within the symbol period of the signal, it is said that the signal undergoes *fast fading* otherwise it undergoes *slow fading*.

With a 2.1 GHz carrier frequency and assuming the speed of electromagnetic signal is $300 \times 10^6 m/s$, the wavelength, λ, is approximately $0.143m$. Note that the value of the carrier frequency is not the exact value for the UMTS carrier frequency. Rather the UMTS specification defines the range of the frequencies can be used. However, since the values of the wave lengths of the such frequencies do not vary significantly, we use 2.1GHz as the carrier frequency for both directions of the communications for computational convenience. If the source is traveling at $4km/h$, normal walking speed of humans, the coherent time is 0.05 second, and while moving at the speed of a vehicle cruising within an urban area, $50km/h$, the coherent time is 0.0044 second. Those coherent time durations are definitely greater than the symbol period calculated previously using Equation 3.2, hence the inequality given in Equation 3.4 is false. Therefore we can conclude that the signal undergoes *slow fading*. In fact, the coherent time is inversely proportional to the speed of the signal source, hence any speed slower than $50km/h$ would experience *slow fading*. In other words, as long as the source of the signal, for example 3G mobile terminal, does not move significantly fast, the signal is affected by *slow fading*.

Overall, if the speed of the mobile terminal is kept below $50km/h$, which is what this project is focused on, the wireless signal undergoes *flat* and *slow* fading. Since the difference between the coherent time and the symbol period is significantly large for the cases this project is focused on, we can assume that the signal mostly suffers from flat fading. The

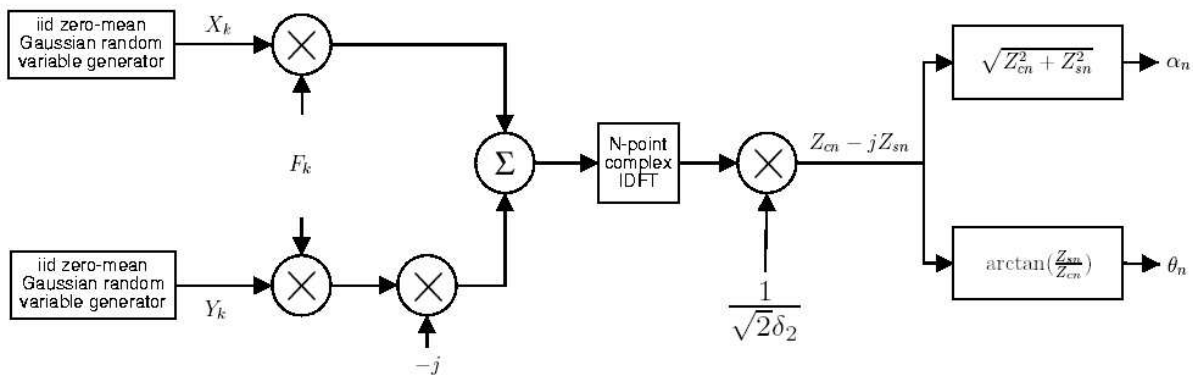

Figure 3.2: Block diagram of a Rayleigh fading channel simulator

next subsection introduces the method to generate the signal envelop[4] which is affected by *flat* fading.

3.2.3 Fading Signal Envelop Generation

This subsection discusses the method to generate the signal envelop, which is affected by flat fading. Traditionally in mobile radio channels, the Rayleigh distribution is commonly used to describe the statistical time varying nature of the received envelop of a flat fading signal. It is also well known that the envelope of the sum of two quadrature Gaussian noise signals obeys a Rayleigh distribution [47]. Using such property of the Rayleigh distribution, a computer simulation model for the *flat fading* channel was proposed by *Smith, J. I.* [56]. Later, *Young* and *Beaulieu* proposed the method of generating correlated Rayleigh random variates by inverse discrete Fourier transform [65]. In this project, we used the fading signal generator described in [37], which is basically the simplified form of the fading signal generator defined by *Young* and *Beaulieu* in [65]. Figure 3.2 shows the block diagram of the flat fading signal generator[5] governed by Rayleigh distribution.

Based on the characteristic that the envelope of the sum of two quadrature Gaussian noise signals obeys a Rayleigh distribution, two independent and identically distributed (iid) random sequences of length N, X_k and Y_k, each sample having the following characteristics,

[4]Appearance of the signal amplitude over certain period of time

[5]We use terms “Rayleigh fading channel simulator” and “fading signal generator” interchangeably. They have the same meanings.

are used to generate the fading channel envelope.

$$E[X_k] = E[Y_k] = 0 \tag{3.5}$$

$$E[X_k^2] = E[Y_k^2] = \delta_1^2 \tag{3.6}$$

$$E[X_k X_l] = E[Y_k Y l] = 0, k \neq l \tag{3.7}$$

$$E[X_k Y_l] = 0, \forall k, l \tag{3.8}$$

The sequence Y_k is converted to the imaginary component and then the random sequences of X_k and Y_k are multiplied by the weighting factor F_k. The role of F_k is to ensure that the resultant amplitude sequence, α_n, has the desired autocorrelation characteristics of the flat fading signal. The definition of the weighting factors, F_k is shown in Equation 3.9 [65].

$$F_k = \begin{cases} 0 & , \quad k = 0 \\ \sqrt{\frac{1}{2\sqrt{1-\left(\frac{k}{Nf_m}\right)^2}}} & , \quad k = 1, 2, ..., k_m - 1 \\ \sqrt{\frac{k_m}{2}\left[\frac{\pi}{2} - \arctan\left(\frac{k_m - 1}{\sqrt{2k_m - 1}}\right)\right]} & , \quad k = k_m \\ 0 & , \quad k = k_m + 1, ..., N - k_m - 1 \\ \sqrt{\frac{k_m}{2}\left[\frac{\pi}{2} - \arctan\left(\frac{k_m - 1}{\sqrt{2k_m - 1}}\right)\right]} & , \quad k = N - k_m \\ \sqrt{\frac{1}{2\sqrt{1-\left(\frac{k}{Nf_m}\right)^2}}} & , \quad k = N - k_m + 1, ..., N - 2, N - 1 \end{cases} \tag{3.9}$$

The value of $f_m = f_D T_s$ where f_D is the maximum Doppler frequency in Hz and T_S is the sampling period in second, and $k_m = \lfloor f_m N \rfloor$. An inverse discrete Fourier Transform (IDFT) is taken of this filtered complex sequence to form the complex time sequence of length N. Note that the original sequences X_k and Y_k, and the weighting factor F_k are defined in the frequency domain, hence IDFT is required to generated the sequence in the time domain[6].. Let us define the inverse Fourier transformed sequence of length N as $z_{cn} - jz_{sn}$ where n ranges from 0 to N. z_{cn} and z_{sn} have zero mean and the same variance δ_2^2 given by

$$\delta_2^2 = \delta_1^2 \sum_{k=0}^{N-1} \left(\frac{F_k}{N}\right)^2 \tag{3.10}$$

[6]For the readers who are not familiar with time/frequency domain concept can refer to the elementary electrical engineering texts such as [28]

Hence, if we multiply the sequence, $z_{cn} - jz_{sn}$, with $1/\sqrt{2}\delta_2$ we have a new sequence, $Z_{cn} - jZ_{sn}$. In this case, Z_{cn} and Z_{sn} have zero mean and variance of $1/2$. The second moment of the amplitude fading is normalized to unity so that $E[\alpha_n^2] = 1$, where α_n is the resultant amplitude sequence of length N, $\alpha_n = \sqrt{Z_{cn}^2 + Z_{sn}^2}$ [37].

Using this generator, we can examine the fading trends of the signal depending on the speed of the signal source, the carrier frequency, sampling period and the number of samples. Figures from 3.3 to 3.5 shown later in this chapter are the resultant amplitude envelope generated by this channel simulator. The discussions of the resultant amplitude envelope are included in the later section.

3.3 Link level Models

The previous section focused on the physical level characteristics of an electromagnetic signal traveling over a wireless channel. During recent decades, lots of research has been conducted in order to understand and characterize fading at the physical layer level. However, when the link level or transport level performance is more of a concern as in this project, the characteristics of physical level should be translated to upper layers. In this section, some main models of the logical level link reflecting the lower layer characteristics, are discussed. Since the detailed description of proposed link model are far too complex to be included in this thesis, only brief introduction of models are included.

3.3.1 *Wu* and *Negi*'s model

Wu and *Negi* [63] tried to model the data link layer generalizing the approach proposed earlier by *Chang* and *Thomas* [11]. Their model combines the physical characteristics with data layer specific parameters. The study was motivated by studies of wire-line networks such as the ATM and Internet models. By adjusting the parameters of the model to resemble the characteristics of the physical characteristics of the wireless link, the link is modeled with two fundamental functions. The advantage is that it takes account of the effect of the queue at the data link layer level, and the several statistical models of the link, such as Rayleigh, Rician, Level Crossing Rate (LCR) and Average Fading Duration (AFD) were used to complete the model. However it does not take into account the effect of the error detection and correction adapted in the 3G wireless link.

3.3.2 *Guasi-Agyei* and *Coutts*'s model

The model proposed by *Guasi-Agyei* and *Coutts* tries to model the wireless IP link [26]. The model follows the wireless model called log-distance path loss with shadowing [7], which is categorized as *large scale model*. Since the effects of *large scale loss* is negligible if a terminal is equipped with a power control system, it is not the suitable model for this project. Its queuing model is directly related to the traffic classification of the UMTS standard [8]. The UMTS standard [8] classifies four specific levels of data within it core network structure depending on the priority and the functionality of data. Each class of data is queued in different prioritized queues for more efficient data flow. However, currently the SIP and SDP messages are not formally classified into any of these four levels of classes defined in [8]. Moreover, such queue is implemented within the wired network infrastructure, hence it is unlikely that this queue.contributes significantly on the SIP layer level delays and the message losses. Hence their mathematical model is of no use for the project.

3.3.3 *Ho*'s model

Ho [30] attempted to profile 3G signals by obtaining live data from the real environment. However, the work has no implications for a fading channel, with which this project is more concerned.

3.3.4 *Poppe, De Vleeschauwer* and *Petit*'s model

The modeling of the virtual link in this project closely follows the work by *Poppe, De Vleeschauwer* and *Petit* [43]. It incorporates all the necessary aspects of the physical layer and the link layer, especially focused on UMTS. The model by *Poppe, et. al.* is focused on voice data which is at the top end of the OSI protocol stack. It allows to study the implication of the wireless link characteristics on voice packets. Additionally, it models the *UE to P-CSCF* virtual link in terms of delay and loss, which is precisely how the network emulator, NISTnet, used in this project can be configured. Since this model has all the aspects of the UE to P-CSCF link requires, the model was chosen to implement the link model for this

[7] The model basically states that as the user moves away from the base station, signal attenuates with log of distance between terminal and the base station. Moreover, the mathematical model uses coeffi cient to describe how the signal is distorted due to the surroundings such as buildings and hills etc. This phenomenon is called shadowing.

project. The following section addresses the details of the model based on effective E_b/N_0 [8], which is the essential concept of this model.

3.4 Effective E_b/N_0 based Virtual Link Model

The concept of effective E_b/N_0 was proposed by *Nanda* and *Rege* in 1998 [41]. The major implication of the method is that it allows a mathematical description of the error properties of modulation schemes, which are derived using average AWGN-based E_b/N_0, for computing the error probabilities in a fading channel. A study of the wireless link performance using this concept was published in 2000 by *Poppe, De Vleeschauwer* and *Petit* [43]. This subsection describes the basis of the effective E_b/N_0 and how it is used to model the 3G wireless links used in this project in detail.

By the aid of the effective E_b/N_0 concept, we define the virtual link, which incorporates the characteristics of the 3G specific low layer functionalities and the degradations of signal during transmission. The link defined for the project is describable with two factors; the transmission delay and the data unit[9] loss rate.

3.4.1 Effective E_b/N_0

There is a rich body of literature dealing with the error performance of convolutional codes in an additive white Gaussian noise (AWGN) channel. The channel model in these analysis assumes an average signal level and AWGN with a fixed variance over a given time interval. In addition, most mathematical formulas, that predict the performance of the modulation schemes and the coding schemes, use the average E_b/N_0 level, which does not reflect any information of fading of the signal. The effective E_b/N_0 is another type of the estimate of the instantaneous E_b/N_0s over some time period. Unlike average E_b/N_0, it takes into account the effect of fading. This estimate, effective E_b/N_0 or $(E_b/N_0)_{eff}$ in short, allows to use the error probability formulas derived using the average E_b/N_0 to obtain the results, which reflect the performance of the digital modulation and convolutional coding over a fading

[8] E_b/N_0 is defined as the ratio of Energy per Bit (E_b) to the Spectral Noise Density (N_o). It is the measure of signal to noise ratio for a digital communication system.

[9] Data unit is a measure of data. For instance at IP layer, the data unit is IP packet, and at the channel coding level the data unit is a code block.

channel (see section 3.4.3 for how the estimate is used for predicting the error rate of the system).

Two algorithms were proposed to calculate the effective E_b/N_0 for given signal amplitude fading characteristics. The results obtained by two algorithms, compared with the results from simulation studies shown in [41], have an acceptable level of accuracy. The algorithm selected for the project is mathematically represented by

$$(E_b/N_0)_{eff} = \min_k (E_b/N_0)_{J(k)} + (E_b/N_0)_{J(k+1)} + \cdots + (E_b/N_0)_{J(k+D-1)}/D \quad (3.11)$$

where $J(k)$ is a function to identify the value of instantaneous E_b/N_0 at a specific time. $(E_b/N_0)_{eff}$ is calculated by the examining the level of E_b/N_0 when kth bit is transmitted. The function $J(k)$ is a dummy function which returns the value of instantaneous E_b/N_0 when kth bit is transmitted. The main reason for favoring this algorithm over other introduced in [41] is that it tends to slightly overestimate the error rate of the modulation scheme with a convolutional coder, if the results are compared with the simulation results [41]. The other algorithm usually underestimates the actual error rate obtained from the simulation study. Moreover, if fluctuations of the signal amplitude level are more radical in the given time interval, the estimates obtained from Equation 3.11 follows the simulation results more closely than the estimates obtained from the other algorithm. Since relatively rapid and drastic changes of the signal envelope in a wireless channel is likely to occur in reality (for instance, see Figures 3.3, 3.4 and 3.5), we have chosen this estimates obtainable by Equation 3.11.

3.4.2 Derivation of Transmission Time

The time needed for transmission of bit streams over the communication channel is lower-bounded (fastest time) by the data transfer rate the channel is offering. In lossless wireless communications (none of transmitted bits are corrupted or erroneous bits are all corrected by channel coding[10] successfully without causing any extra delay, hence no retransmission occurs), the upper bound can be deduced by the interleaving of the bits (see Chapter 2. In such cases, disregarding any degradation factors occurring in wireless channels, the delay can be upper-bounded theoretically by

[10]In the case of SIP/SDP over UMTS wireless channel, channel coding is provided by two types convolutional codes; see Chapter2 for the details of the encoders.

$$T = \frac{B_F}{R_T} + 2N_i T_R \tag{3.12}$$

where N_i is the Transmission Time Interval (TTI) (from 1 to 8) discussed in Chapter2. T_R is the time required to transmit one radio frame (i.e., $10ms$ typically), B_F is the size of the data unit and R_T is the data transfer rate of the channel. The first term of Equation 3.12 is obviously the lower-bound constrained by the data unit size and the data transfer rate. The second term comes from the fact that the bits arriving when TTI has just began will only be received at the end of the next TTI [43] (see Chapter 2 for details on interleaving schemes) This implies that the interleaving scheme contributes the delay of the data transfer over the wireless channel.

3.4.3 Derivation of Data Unit Loss Rate

The theoretical data unit loss rate[11] is closely related to an error correction scheme that the communication system adapts. Recommended channel coding for the UMTS air channel based on two convolutional coders and a turbo coder; see Chapter2. The turbo coder is recommended for use when enough bandwidth is arranged to the mobile terminal, whereas the convolutional coders are recommended to be used for other cases. In other words, bandwidth required for the session is needed to reserved before using the turbo coder. In this project we concerns multimedia sessions are being established by the SIP/SDP messages, hence the reservation of the resources such as bandwidth is expected to be finished once the message exchange is successfully completed. Hence, for more realistic scenarios, data unit loss should be considered assuming that the two convolutional coders, defined in [4] and [3] are used (see Figure 2.3). The choice of the type of convolutional encoder is up to th network operators. Two encoders shown in Figure 2.3 have the coding rate of $1/2$ and $1/3$. If we use $1/3$ rate encoder, then it gives the higher chance of recovering the corrupted data, however, it approximately triples the size of the original data stream to provide such protection. On the other hand, $1/2$ rate encoder is still capable of correcting data with less chance than $1/3$ rate encoders, while only approximately doubling the size of the original data. The readers who are not familiar with a convolutional encoder and its technical terms, should refer to Appendix A throughout this section.

[11] We use the terms *rate* and *probability* interchangeably. Those two terms have the same meaning

Bit Error Rate

Typically, the bit error rate is bounded by (see [46] for instance)

The probability of having a corrupted bit (or bit error rate) is bounded by

$$P_b \leq \sum_{d=d_{free}}^{\infty} \beta_d P_2(d) \tag{3.13}$$

where β_d is the total number of information errors produced by the wrong paths of Hamming weight[12], d, that diverge from the correct path and re-merge with it at some later stage. $P_2(d)$ is a probability that the decoder makes an error in the pairwise comparison of two paths through its state space differing in d bits. $P_2(d)$ is known to be

$$P_2(d) = \frac{1}{2}\text{erfc}\left(\sqrt{R_c d \frac{E_b}{N_0}}\right) \tag{3.14}$$

where $erfc(u) = 2/(\sqrt{\pi}\int_u^{\infty} exp(-z^2)dz)$ is the complementary error function, and R_c is the coding rate [46]. The theoretical values of $P_2(d)$ for BPSK and QPSK are the same if QPSK employs prior coding schemes such as Gray coding. In order to use the formula the value of d_{free}, the free Hamming distance, should be known and the value for β_d for each distance should be computed as well.

Note that as the argument of the complementary error function gets bigger the resultant value becomes smaller. For example, the value of $erfc(10)$ is equal to $2.0.885 \times 10^{-45}$ which is very close to 0 and does not affect the result of summation of Equation 3.13. Hence, there is no need to execute the summation of Equation 3.13 up to infinity. However, it is required to find out the free distance, d_{free}, of a convolutional code and the β_d value for each coding distance. A simple C based code, which returns the value of β_d depending on the value of d is implemented. Table 3.1 shows the relevant values of d and β_d obtained by feeding the binary sequences which are equivalent to the decimal numbers range from 0 to 2^{9}[13].

The value, denoted by d_{free}, is a measure of a convolutional code's ability to combat channel noise and it is called the free distance. The free distance of a convolutional code is defined as the minimum Hamming distance between any two code words in the code. A

[12]Hamming weight or Hamming distance is the number of bits which differ between two binary strings

[13]The constraint lengths of convolutional encoders used in this project are 9. Hence to generate all the possible code word according to all the possible input bit stream, binary sequences from 000000000 to 111111111, which is 2_9 in decimal, are used.

Rate 1/2 encoder		**Rate 1/3 encoder**	
d	β_d	d	β_d
12 (d_{free})	33	18 (d_{free})	11
14	281	20	32
16	2094	22	195
18	12267	24	564
20	59935	26	1473
22	221176	28	5047
24	599406	30	16285
26	1215341	32	46478

Table 3.1: Values of β_d for convolutional encoders

convolutional code with free distance d_{free} can correct t error if and only if d_{free} is greater than $2t$ [28]. Due to such a property, 1/2 rate code can correct up to 6 bit errors in a single code word and 1/3 rate code can correct up to 9 bit errors.

Since the convolutional encoders defined in the UMTS specification, have the coding rates of 1/3 and 1/2, the value for R_c can either be 1/3 or 1/2. . Equation 3.14 is derived using average value of E_b/N_0. As mentioned, the average value of E_b/N_0 does not reflect the fading effect of the wireless channel. Hence, as it is proposed in [41], we substitute average E_b/N_0 with the effective E_b/N_0 to evaluate the performance of the channel in fading conditions.

Instantaneous E_b/N_0

Since $(E_b/N_0)_{eff}$ is an estimate of the instantaneous E_b/N_0s over some time period, it is necessary to calculate the values of instantaneous E_b/N_0s. Since, in UMTS wireless transmission, the interleaving of two level is used. the bits from the encoded bit stream are transmitted in the order that is not the same as the order they appear in the original encoded data. For instance, each consecutive bit from the encoded bit stream are transmitted in non-consecutive time-slots. Spreading the consecutive bits over non-consecutive time-slots has benefit of minimizing consecutive bit losses. Such spreading of bits over time-slots in a radio frame is called the first-level or intra-frame interleaving. To have better protection against

the "burst" of consecutive bit losses higher level of interleaving called the second-level or inter-frame interleaving is used with intra-frame interleaving. The parameter TTI defines how far the inter-frame interleaving is allowed. For instance if TTI is 1, interleaving within one radio frame is permitted, hence it is equivalent of using only first-level interleaving. If TTI is set to 8, then the consecutive bits can be spread over up to 8 radio frames Hence, in order to estimate the value of $(E_b/N_0)_{eff}$, the instantaneous E_b/N_0s of the time-slots which carry the original message are only needed to be used. [43] derived a set of formulas to compute which time slot of which radio frame carries nth bit of the convolutionally encoded data stream.

$$s(n) = (N_I Span(n) + Frame(n))\, N_s + Slot(n) \tag{3.15}$$

$$Span(n) = \left\lfloor \frac{n}{N_I B_R} \right\rfloor \tag{3.16}$$

$$Frame(n) = \sum_{i=0}^{2} \left(\left\lfloor \frac{Offset_{frame}(n)}{2^i} \right\rfloor \mod 2 \right) \left\lfloor \frac{N_I}{2^{i+1}} \right\rfloor \tag{3.17}$$

$$Slot(n) = \left\lfloor \frac{1}{2} \sum_{i=0}^{4} \left(\left\lfloor \frac{Offset_{slot}(n)}{2^i} \right\rfloor \mod 2 \right) \left\lfloor \frac{2N_S}{2^{i+1}} \right\rfloor \right\rfloor \tag{3.18}$$

$$Offset_{frame}(n) = n - N_I \cdot B_R \cdot Span(n) \tag{3.19}$$

$$Offset_{slot}(n) = \left\lfloor \frac{Offset_{frame}(n)}{N_I} \right\rfloor \tag{3.20}$$

B_R is the number of bits that each radio frame contains. N_I represents the value of TTI and N_S represents the number of time-slots in one radio frame ($N_s = 16$). The number of time slot which carries nth bit is computed by $s(n)$. If the resultant value of $s(n)$ is 0, then it represent the nth bit is transmitted by the first time-slot of the first radio frame. If $s(n)$ returns 18, it means that the nth bit is transmitted by 3rd time-slot of the second radio frame, since each radio frame has 16 time-slots and the numbering starts from 0. Note that the value of n also starts from 0. For example, $n = 1$ means the second bit.

The subroutines such as $Span(n)$, $Frame(n)$ and $Slot(n)$ delivers the intermediate results for the final computation. $Span(n)$ is the number of interleaving span in which bit n is transmitted. For instance, if the size of radio frame, B_R, is 16 bits long and the TTI, N_I is 2, it can only place 32 bits within one span of the interleaving. Hence is the 33rd bit comes in for interleaving it places the bit into the next span of interleaving and the function returns the

value of 1 representing the next span of interleaving. $Frame(n)$ computes the number of radio frame (relative to the beginning of the span) and $Slot(n)$ gives the number of time-slot (relative to the radio frame) in which bit n is transmitted.

Assuming the value of instantaneous E_b/N_0 within one time-slot is constant, the Equation 3.11 can be rewritten as

$$(E_b/N_0)_{eff} = min_{n=0}^{N-D} \left(\sum_{i=0}^{D-1} \frac{(E_b/N_0)_{S(n+i)}}{D} \right) \tag{3.21}$$

The justification of the assumption is discussed in a later section. The value of D can be found by dividing the constraint length of the encoder by the coding rate [43]. Both convolutional encoders have a constraint length of 9, hence the value of D is equal to 4.5 for the code rate of 1/2 and 3 for the code rate of 1/3.

E_b/N_0 during each time-slot depends on the energy of AWGN (noise) and the signal level fluctuated by the fading. The methods to generate time related faded signal amplitude were discussed previously. Using the filter function given in Equation 3.9 and ***Rayleigh's distribution***, instantaneous E_b/N_0, which corresponds to one time-slot can be calculated.

As discussed in section 3.2.3, the resultant fading signal amplitude α_n has $E[\alpha_n^2] = 1$ (or 0dB). Each value of α_n represents the fluctuation level of the signal which is sampled at a given sampling period. Since we are assuming the signal amplitude stays the same within one time-slot length of time ($0.625ms = 10ms/16$), the α_ns, which are sampled at $0.625ms$ were obtained to compute the instantaneous E_b/N_0. Using the values of α_ns and Equation 3.15, we can find out the signal fluctuation level at a given bit. For example, if nth bit of encoded data stream is places in second time-slot of the second radio frame, then the signal fluctuation level corresponds to the 18th sample of α_n. The obtained signal fluctuation level, α_n, of the time slot is then used to calculate the instantaneous E_b/N_0 of the time-slot which is carrying nth bit as

$$\left(\frac{E_b}{N_0}\right)_{s(n)} = \left(\frac{E_b}{N_0}\right)_{avg} \times \alpha_n^2 \tag{3.22}$$

$(E_b/N_0)_{s(n)}$ represents the E_b/N_0 level of the particular time-slot which carries nth bit and α_n is the signal fluctuation level at the time that time-slot is traveling through wireless channel. $(E_b/N_0)_{avg}$ is the average of E_b/N_0 level over relatively long period (e.g., orders of tens of seconds), which is a fixed variable. Figures 3.3 to 3.5 shows the α_ns sampled at $0.625ms$.

The computed values of instantaneous E_b/N_0s for each time-slot are fed into Equation 3.21 to compute the resultant value of effective E_b/N_0 for a given fading channel.

The values for $(E_b/N_0)_{avg}$ and $(E_b/N_0)_{eff}$ are both constant over a given period of time. However, $(E_b/N_0)_{avg}$ is an estimate that represents the mean value of the instantaneous E_b/N_0s, whereas $(E_b/N_0)_{eff}$ is another form of estimate which is controlled by fading and interleaving schemes. What we discussed here is that since the fading signal envelop does not change the value of $(E_b/N_0)_{avg}$ [14] if relatively long time period is considered, we can set the value of $(E_b/N_0)_{avg}$ as a controllable parameter and compute the value of $(E_b/N_0)_{eff}$ according to $(E_b/N_0)_{avg}$, and fading signal envelop and interleaving scheme. Fixing the value of $(E_b/N_0)_{avg}$ means that over a long time period, a signal transmitter retains the energy of the transmitting information relative to the background noise. It is mathematically convenient and true. That is why in telecommunication researches evaluating the channel coding or modulation scheme by fixing the value of $(E_b/N_0)_{avg}$ at some level is a general practice. The usage of smaller value of $(E_b/N_0)_{avg}$ means that the transmitter uses less power and it may not require high-cost equipment, but typically it is more vulnerable to the losses due to the severe channel condition. Moreover, the larger value of $(E_b/N_0)_{avg}$ (i.e., an expensive equipment and higher signal power) usually is more resistive to the losses due to the severe channel condition.

Probability of Data Unit Loss

With the computed value of effective E_b/N_0, Equation 3.14 and 3.13 are used to obtain the bit error rate of a given fading channel. The probability of the data unit drop can easily be calculated as

$$P_{loss} = 1 - (1 - P_b)^{B_F} \tag{3.23}$$

where B_F is the size of the data unit in bits.

[14] The expected value of the signal fluctuation due to fading over relatively long period is 1 (see section3.2.3), thus the values of $(E_b/N_0)_{avg}$ for non-fading channel and the fading channel are the same. Hence usage of $(E_b/N_0)_{eff}$ instead of $(E_b/N_0)_{avg}$ is needed.

3.5 Project Specific Model Description

This section defines three scenarios which resemble the realistic cases where user initiates or receives the multimedia session invitation in terms of the terminal speed. These three scenarios were considered when we obtained the experimental data.

3.5.1 Scenarios of Terminal Movement

In order to investigate the performance of the multimedia session initiation procedure of 3G networks, three basic scenarios were used. They are modeled in terms of moving speed of terminal which receives or initiates session invitation. They are

- **Slow moving mobile terminal (1km/h)**: It resembles the case where a user initiates the session while moving slowly. People tends to walk slower when they are concentrating on other things.
- **Walking speed terminal (4km/h)**: If some sort of "speed-dialing" functionality is implemented in the terminal, the user does not need to slow down in order to glance at the display of the mobile terminal.
- **Urban vehicular speed (50km/h)**: It resembles the case when a user initiates the session while in a car moving at 50 km/h, the legal urban speed limit in New Zealand.

Basic assumptions are; 1) there is no LoS[15] existing during the whole period of session initiation or post-dialing phase and 2) the base station is located at a fixed position. 3) The factors introduced by the climate condition are ignored and the only sources of signal fluctuations are due to fading discussed. Every assumption mentioned in modeling UE to P-CSCF virtual link still holds for these scenarios.

3.5.2 Scenario Specific Signal Fluctuation

Each scenario exhibits different fading characteristics due to different speeds of mobile terminals. The Doppler frequency defined in terms of speed is given by

[15]LoS, Line of Sight is a physical direct path between mobile terminal and the base station, typically in mobile communications

$$f_D = \frac{v}{\lambda} \tag{3.24}$$

where λ is the wavelength of the carrier signal and v is the speed of the signal source (e.g., terminal) in m/s. Assuming that the speed of electromagnetic wave is $3 \times 10^8 m/s$ and the carrier frequency is 2.1GHz, the wavelength of the carrier signal is approximately $0.1429m$. Table 3.2 shows the relevant values of the Doppler frequencies for our three scenarios.

Terminal Speed	**Doppler Frequency**
1 km/h	1.9439 Hz
4 km/h	7.7754 Hz
50 km/h	97.193 Hz

Table 3.2: Doppler frequencies for three scenarios of terminal movement

For a given Doppler frequency, the time related sequence of fading can be obtained by the method discussed in section 3.2.3. The selection of the sampling rate of such signal amplitude is based on the assumption that E_b/N_0 does not change during one time-slot of the radio frame. That is, E_b/N_0 does not changes during the time of 0.625 ms ($= 10ms/16$), since each radio frame is transmitted over 10 ms of period and the each radio frame has 16 time-slots. This assumption is important when we compute the estimate of effective E_b/N_0.

The assumption can be justified by the Level Crossing Rate (LCR) .

Level Crossing Rate (LCR) gives the rate that the signal crosses certain level per one second duration, and it is defined as

$$N_R = \sqrt{2\pi} f_D \rho e^{-\rho^2} \tag{3.25}$$

where f_D is the Doppler frequency and ρ is the normalized threshold[16] which is the value of the specific signal level of R, normalized to the local RMS amplitude of the fading envelope [35].

LCR is the expected number of times that the signal, which has amplitude of R_{rms}, crosses a threshold level R over one second duration. If the normalized threshold ρ is 0.01,

[16] ρ is defined as R/R_{rms}, where R is the signal level you are considering and the value of R_{rms} is the local rms amplitude of the actual fading signal. Local rms can be calculated by 1) collect signal sample continuously for some amount of time 2) square the amplitude, 3) average the amplitudes and 4) square root the result.

LCR computes the expected number of times that the signal crosses the threshold that is 1/100 of RMS value of the actual signal. If the normalized threshold ρ is 1, LCR computes the expected number of times that the signal crosses the threshold that is the same as RMS value of the actual signal

The values of LCR for three scenarios are shown in Table 3.3 for varying normalized threshold.

Doppler frequency	**Normalized Threshold**	**Crossing/sec**	**Crossing/time-slot**
1.9439 Hz	0.01	0.0487	3.0438^{-5}
1.9439 Hz	0.1	0.4824	3.0150^{-4}
1.9439 Hz	1.0	1.7925	0.0011
7.7754 Hz	0.01	0.1949	1.2181^{-4}
7.7754 Hz	0.1	1.9296	0.0012
7.7754 Hz	1.0	7.1700	0.0045
97.193 Hz	0.01	2.4360	0.0015
97.193 Hz	0.1	24.1203	0.0151
97.193 Hz	1.0	89.6254	0.0560

Table 3.3: Level Crossing Rates (LCR) for varying normalized threshold

Even in the cases when the normalized threshold ρ is 1, it is unlikely that the signal fluctuation will occur in a time period equivalent to the one time-slot, $0.625ms$. Hence, we assume the constant E_b/N_0 over a single time-slot. Based on the assumptions and the fluctuated signal envelope generation method discussed in the previous chapter, we generated the time sequences of signal envelop according to the terminal's traveling speeds.

Figures 3.3, 3.4 and 3.5 illustrate 3000 samples of signal amplitude with previously calculated Doppler frequencies and the sampling rate for illustrative purpose only. The first sample of the graph corresponds to signal level of the first time-slot of the first radio frame, and the sixteenth sample corresponds to the last time-slot of the first frame, and so on. The sample values were used to compute actual $(E_b/N_0)_{s(n)}$ to be used in Equation 3.22 (i.e., E_b/N_0 for each time-slot).

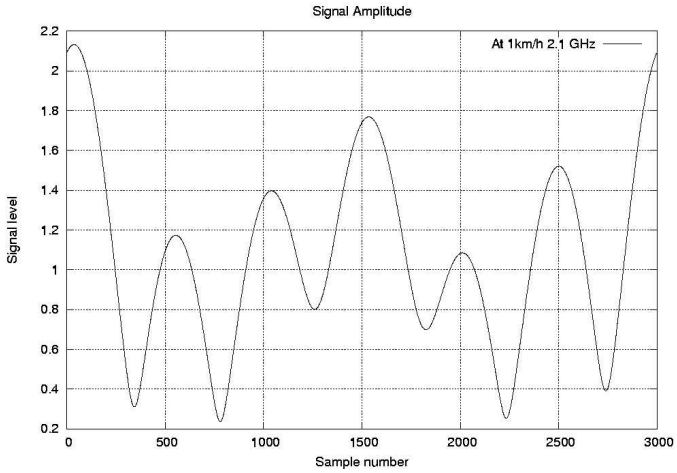

Figure 3.3: An example of amplitude fading level when the signal source moves at 1 km/h

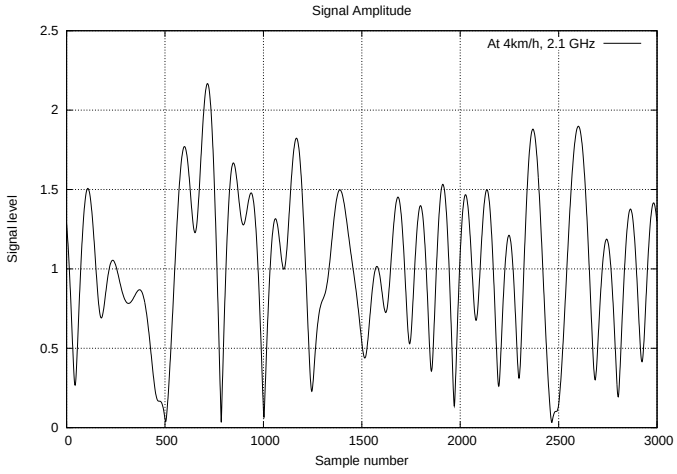

Figure 3.4: An example of amplitude fading level when the signal source moves at 4 km/h

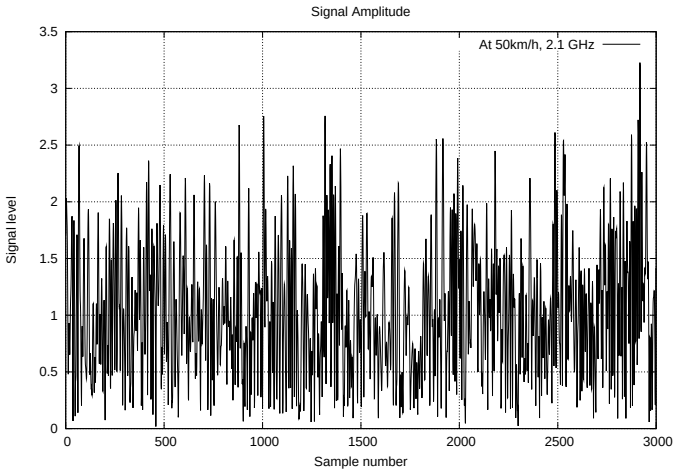

Figure 3.5: An example of amplitude fading level when the signal source moves at 50 km/h

3.5.3 Computed Values of Effective E_b/N_0

We have discussed all the relevant information and assumptions to compute effective E_b/N_0. This subsection discuses numerical results obtained for effective E_b/N_0 for the three scenarios of terminal movement.

Since [43] and [41] suggest relatively long samples in order to obtain more reliable effective E_b/N_0, 150000 samples of signal level (equivalent to 93.75 sec) were obtained and then Equation 3.21 was used to compute the effective E_b/N_0 for the three scenarios. The effective E_b/N_0 is, as shown in Equation 3.21, dependent on the interleaving scheme, since its computation follows the numbering of the time-slot in which the consecutive bits are transmitted. Using all the information obtained, the computed values of effective E_b/N_0 are shown in Appendix C. The values show that in the case of slower moving source, the effective E_b/N_0 is much lower. That is because, as shown in Figure 3.3, 3.4 and 3.5, the fading lasts longer at the lower speed than at the higher speed. Hence, multiple numbers of consecutive bits, although transmitted in non-consecutive time-slots, can suffer from fading. Note that the algorithm for estimating the effective E_b/N_0 chosen in this project tends to underestimate the value of real E_b/N_0, hence in the actual case the signal may suffer slightly less severely. In addition, as expected, more separation of consecutive bits by interleaving results in higher effective E_b/N_0, implying that the higher TTI is more resistant to bursts of errors due to fading.

Theoretically, a lower channel coding rate should provide better protection over an er-

roneous channel condition. If we consider the case where the traveling speed of terminal is $1km/h$ and a TTI of 1, both convolutional encoders perform similarly, however at a higher speed or a higher TTI, the lower rate (i.e., 1/3 rate encoder) outperforms the higher rate one. Such a performance of channel encoding is not detectable for other scenarios. This happens because at lower speed and lower TTI too many consecutive bits suffer fading and the decoder fails to recover the original values.

The overall error rates are bounded values provided by Equation 3.13. The implications of loss due to error are discussed in Chapter 7, together with the overall post-dialing delay of multimedia session.

Chapter 4

SigComp Scheme

Many application layer protocols used for multimedia communications are text-based and engineered for broadband links. It is obvious since multimedia sessions usually require high rate of data exchanges of streaming images and the sound data and the data transfer rate of the link is rather fixed in the normal Internet. Typical examples of such protocols used for multimedia sessions in the Internet and 3G mobile networks are SIP and SDP. The size of SIP messages including SDP varies from a few hundred bytes up to a few thousand bytes. The data transfer rate in the Internet is somewhat fixed before, during and after the multimedia sessions. However, for 3G mobile networks, since the wireless bandwidth is a scarce resource, only a minimum data transfer rate is expected to be available to a user before session initiation. The session initiation or post-dialing phase should reserve the resources for the multimedia sessions, while the messages which seek the multimedia session requirements are transmitted over a limited bandwidth, hence limited data transfer speed.

For 3G networks, usage of the Internet based protocols, SIP and SDP, is necessary in order to preserve compatibility between two networks. However, since SIP and SDP are designed for broadband links, modification of datagrams of SIP and SDP had to be done for reducing the size of datagram, so Signal Compression (SigComp) scheme proposed by IETF to reduce the size of SIP and SDP messages is a solution to that problem [22], [45].

This chapter briefly introduces the structure of SigComp as defined in [45] and investigates the performance of the scheme in terms of size reduction. In Chapter 7 the transmission delay due to the size of datagrams and data transfer speed is discussed. Additionally, a simplified static SigComp structure is proposed and compared with the original SigComp scheme. In this chapter, the term *"dynamic SigComp"* refers to the original SigComp de-

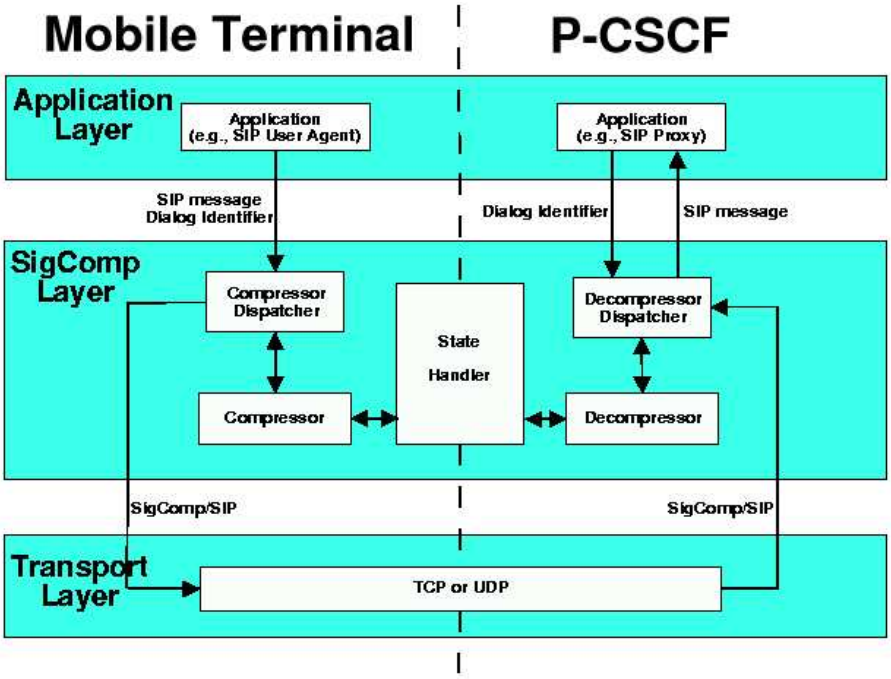

Figure 4.1: Structure of SigComp layer

fined in [45] and the term *"static SigComp"* refers to the proposed simplified version of SigComp.

4.1 SigComp Structure

SigComp, in logical terms, resides between the application layer and the transport layer (see Figure 4.1). In order words, SIP/SDP message is passed to SigComp layer and then to UDP or TCP in IP networks.

In 3G mobile networks, the bandwidth of the radio link, whether it is CDMA2000 or W-CDMA based, is the bottleneck from the perspective of overall data transfer speed. Hence, the SIP message exchange between UE and the P-CSCF must utilize all the possible bandwidth available. Network entities on the wire-line portion do not need to employ the SigComp layer, because they have no bandwidth shortage problems. In order words they are connected with broadband links.

SigComp is a compression scheme which is not restricted to a particular compression algorithm, and its sole purpose is to reduce the size of datagram, hence reducing the data

transfer delay. The results from [61] confirm that compression is capable to significantly reduce SIP/SD datagram in size. The compression ratio reaches its maximum, if compressor and decompressor perform dynamic updates on the predefined dictionaries defined in [22]. As shown in Figure 4.1, SigComp layer uses a state handler and SIP dialog identifier to uniquely compress and decompress the messages. The state handler reassembles the dictionary for a given dialog value, hence achieving the maximum compression ratio after a few message exchanges. The compression dictionary [22] corresponds to the initial state of the SigComp layer.

The decompression employs a novel notion of Universal Decompressor Virtual Machine (UDVM), which is optimized for the task of running decompression algorithms. The UDVM can be configured to understand the output of many well-known compressors such as DEFLATE [16]. The major benefits of using UDVM is that it decompresses any type of compressed messages with a given set of algorithms. Hence, the implementation of the entities which adapts SigComp layer, for instance UE and P-CSCF, is a major issue. Since the evaluation of compression algorithms is not within the scope of the project, and since their preliminary evaluation has been completed, see [61], no more discussion of SigComp is given.

4.2 Performance of SigComp

As mentioned in the Chapter 2, multimedia session in 3G network should be accomplished upon the successful SIP/SDP message exchanges among multiple participants. A typical message exchange sequence illustrated in the 3GPP specification involves 8 messages [2]. In order to investigate the reduction of size due to SigComp, we have implemented experimental SigComp layer with standard Linux library, *zlib 1.1.4* and the SIP/SDP messages were compressed with the LZ77 based DEFLATE algorithm. The initial dictionary and the dictionary updates followed what is required in [22] and [45].

4.2.1 Dynamic SigComp

IETF suggests that SigComp should use the pre-defined dictionary in order to maximize the initial compression ratio, and later, the dictionary should be manipulated by the contents of the next SIP messages, so that the compression ratio of the trailing messages can reach

its maximum. For example, when a caller transmits an *INVITE* message to the callee, it compresses the message with the initial dictionary and then the dictionary itself is updated with the contents of the *INVITE* message. The callee receives the *INVITE* message by decompressing the message with the initial dictionary. The response of the *INVITE* message, *Trying*, is later compressed with the dictionary that is updated with the *INVITE* message it has received. Since the contents of the SIP/SDP stay similar during the same dialog of the SIP/SDP message exchanges, it is expected that the contents of the dictionary could hold most of relevant information to maximize the compression after few message exchanges of the same dialog.

Type	**Original Size**	**Dynamic Compression**	**Compression Ratio**
INVITE	1694	574	66.12 %
Trying	243	43	82.30 %
Session Progress	1177	276	76.55 %
PRACK	981	90	90.82 %
OK (PRACK)	694	45	93.52 %
UPDATE	1068	61	94.29 %
OK (UPDATE)	676	73	89.20 %
Ringing	466	37	92.06 %
Total Size	6999	1199	82.87 %

Table 4.1: Reduction of SIP/SDP message sizes using dynamic SigComp (values are in bytes)

Table 4.1 shows the sizes of the original SIP/SDP messages and their dynamically compressed sizes. The reduction in size is significant; overall an approximately 83% reduction in size is observed. However, such an updating mechanism is dependent on a close relationship between application layer protocol (SIP/SDP) and SigComp layer (see Figure 4.1). The application layer should provide a proper dialog identifier so that the transmitting and receiving messages are de/compressed with the proper dictionary. The status of the dictionary is governed by the state controller. Although state controllers are physically separated into a multiple number of network entities (in 3G networks, UE and P-CSCF), the status of them should be kept the same, since it provides the proper dictionary to perform de/compression.

Any failures of this mechanism results in a complete loss of the message even though the message is received without any error caused by link problems.

Many researchers blame the statefulness of the network for degrading overall performance of the network in terms of response time, because any small error of state in any network node results in complete loss of the data. Even though dynamic SigComp reduces the size of the overall datagram significantly, the consequences of the state mis-match between UE and P-CSCF can cause many inconveniences. It is expected that the session initiation phase or post-dialing period should be kept to a minimum. However, failure of the SigComp layer may cause additional delay that users may not tolerate; since 3G terminals are essentially the evolved form of cellular phone, the delay expectation of users will be that of normal cellular phone services.

4.2.2 Static SigComp

Since the stateful mechanism can lead to complete failure of the message exchanges in worst-case scenarios, it is of obvious interest to investigate the performance of static compression. Static compression gets rid of the need for the state hander, hence no manipulation of the dictionary is necessary. In order words, if static compression is used, the logical link between UE and P-CSCF at SigComp layer is no longer required. Moreover, it reduces the work load of the SigComp layer significantly.

Every SIP/SDP message is compressed only with the initial dictionary defined in [22]. Table 4.2 shows the sizes of the SIP/SDP messages without dictionary updating by the state handler of the SigComp layer. Note that the size of *INVITE* message is the same as that of a dynamically compressed *INVITE* message. That is because the *INVITE* message is the first message of the dialog, and can only be compressed by the initial dictionary. Overall, a 55% reduction in SIP/SDP size was observed for the particular set of SIP/SDP messages shown in TS 24.228 [2]. Note that the size reduction is highly dependent on the contents of the SIP/SDP. Other sets of the message will results in different ratios.

In terms of the compression ratio compared with that of dynamic SigComp, static SigComp underperforms. However, static SigComp simplifies the layer structure significantly by avoiding state updating and communication between application layer and SigComp layer. In addition, there is no fear of multimedia session failure due to the problem caused by mis-matching of the state of the SigComp layer.

Type	Original Size	Static Compression	Compression Ratio
INVITE	1694	574	66.12 %
Trying	243	163	32.92 %
Session Progress	1177	533	54.72 %
PRACK	981	467	52.40 %
OK (PRACK)	694	331	52.31 %
UPDATE	1068	509	52.34 %
OK (UPDATE)	676	322	52.67 %
Ringing	466	244	47.64 %
Total Size	6999	3143	55.09 %

Table 4.2: Reduction of SIP/SDP message sizes using static SigComp (values are in bytes)

4.3 Conclusion

SigComp is proven to be effective for reducing the size of SIP/SDP datagram hence reducing the message exchange duration. For example, when the message sequences, included in [2], gets compressed dynamically with DEFLATE, approximately 83% of size reduction was observed and such reduction is expected to shorten the message travel duration of control messages. However the faster message exchange comes with the penalty of system complexity. The dynamic SigComp needs a state handling mechanism to continuously update the dictionary. Moreover, the state of the SigComp layer at the transmitter and the receiver should be kept identical and close communication between application layer and SigComp layer should be maintained. The results shown in Tables 4.1 and 4.2, are only for the particular set of SIP/SDP messages used in this project. Other sets will produce different compression ratios, however, given results in Tables 4.1 and 4.2 clearly show the benefit of introducing complexity to the system.

Many network engineers complain that the statefulness of the network entities complicate the system implementation, and in the case of state mis-match, all the network entities which share the state must roll back to a certain state where they can resume the state handling. Such catastrophic events are rare, however, penalties for the failure is rather serious. The static SigComp proposed in this chapter was motivated by the idea of stateless compression, which gets rid of the potential problem of the currently proposed dynamic SigComp.

The static SigComp underperformed compared to dynamic variant in terms of resultant sizes. However, if network operations are able to guarantee a certain amount of bandwidth to idle terminals, the performance of static SigComp can be as acceptable as that of dynamic SigComp.

Chapter 7 presents more discussions on the two variants of SigComp in terms of delay measured, when the compressed messages are transmitted over 3G wireless links.

Chapter 5

Binary Hybrid Protocol Design

Since the beginning of computer networks, design of the communication protocols has significantly influenced the network performance and the applications that use the protocols. The protocols are typically associated with a hierarchy of up to 7 different layers depending on the functionalities they provide. The hierarchical stack of protocol's layers following the guideline called Open System Interconnection (OSI) structure is shown in Figure 5.1. Because of the complexity of the communication task; no single standard suffices. Rather, the functions should be broken down into more manageable parts and organized as a communication architecture. The architecture would then form the framework for standardization. This line of reasoning led International Organization for Standardization in 1977 to establish a subcommittee to develop such an architecture. The result was the OSI reference model.

Although the essential elements of the model were in place quickly, the final ISO standard, ISO 7498, was not published until 1984. A technically compatible version was issued by International Consultative Committee on Telegraphy and Telephony CCITT (currently ITU-T) as X.200 [57].

The IP based structure does not exactly follow the ISO model, because the development of the IP based, or rather specifically TCP/IP based, protocol structure pre-dates the publication of the OSI model. However, it is possible to map the TCP/IP structure into OSI architecture with the omission of a few protocol layers. Internet Protocol (IP) is typically classified as Network layer, because IP provides addressing and routing related information. Transfer Control Protocol (TCP) is usually regarded as a Transport layer protocol since it implements the packet flow control and retransmission of lost or corrupted packet, hence providing error-handling. Since in an IP-based network User Datagram Protocol (UDP) is

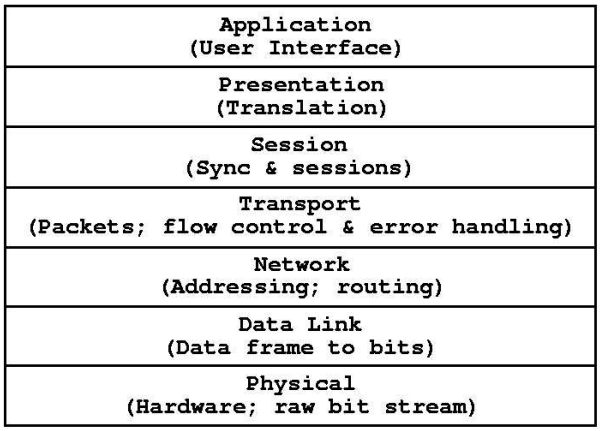

Figure 5.1: Structure of OSI

regarded as an alternative to TCP, it can also be classified as Transport layer protocol, even though it does not have error handling functions.

Such low level protocols are usually defined in bit streams with fixed sizes whereas at higher levels, such as the application layer, protocols are defined with plain text. This chapter discusses the difference between two design paradigms and their advantages and the disadvantages. Later a novel paradigm for designing the protocol is proposed with examples of proposals for new SIP and SDP.

5.1 Binary-based Protocols versus Plain Text-based Protocols

The basic notion of protocol design, especially in the Internet, is based on the principle of designing a Internet infrastructure. That is, the notion is of *smart terminal and simple network*. The notion implies that the workloads are usually concentrated at the end nodes, such as terminals users use. The routers or gateways perform the relaying and filtering functions, hence are required to be simple and should not take significant amount of time to process the data. Hence compact forms of protocol headers are favored. On the other hand, at the user end the terminals should give some form of response to the user in readable form, hence the readable format of the protocols are favorable.

One of the typical examples of text-based protocol is Hyper Text Transfer Protocol

(HTTP). HTTP carries content that is directly interpreted by browsers such as Netscape or Internet Explorer. In order to ensure fast and effective interpretation, HTTP is defined to carry plan text with plain text based *"tags"* which manipulate the appearance of the content. As the Internet became more popular, the size of the contents of HTTP got bigger, which might have caused serious problems if the Internet evolution had not been backed up by link speed increases. Despite the speed advantages, the size of HTTP contents is the major cause of delay when user accesses new web pages. For instance, the HTTP contents of the University of Canterbury home page, containing image files, exceeds 70 kilo bytes, whereas all the underlying layer protocols use merely 66 bytes of headers (32bytes for TCP, 20 bytes for IPv4 and 14 for Ethernet). However the major concern in defining HTTP was to develop a protocol which can carry virtually any type of hypertext and ease the working load of browsers which display the contents. The size was a secondary concern due to the high link speed of the modern Internet. For cellular wireless applications, WAP[1] is used in order to compensate for the relatively slow speed of the communication link. Basically, Internet-based application layer protocols follows the same notion of design paradigm, and this chapter refers to it as the text-based design paradigm. Major benefit of using such a paradigm is that the protocols are widely applicable, since there is no restrictions on the contents and the sizes. Hence it gives more flexibility and the scalability.

Relaying network entities, such as routers and gateways, interpret the contents of lower layer protocols (usually below "Session" layer). In order to complete the functions of relaying entities in a relative short duration of time, the contents need to be compact and preferably of a fixed size. That is why the binary based protocol design paradigm dominates in these layers. Internet Protocol (IP) is the core protocol of the Internet, and is defined entirely by binary sequences. Since a binary sequence can be more efficiently scanned by software and even hardware, the relaying entities of network may concentrate on more time consuming processes like routing decisions. Moreover, because the size of the protocol headers are usually much smaller than that of text-based protocols, the protocol headers do not contribute to the degradation of the response time significantly. Despite the advantages, the binary based protocols have strict restrictions on the size. When designing binary-based protocols, the size of each header field is strictly defined. The sizes are chosen based on

[1]WAP (Wireless Application Protocol) is a specifi cation for a set of communication protocols to standardize the way that wireless devices, such as cellular telephones and radio transceivers, can be used for Internet access, including e-mail, the World Wide Web, newsgroups, and Internet Relay Chat.

predictions. For instance, IPv4 uses 4 byte (32 bits) long addresses. Theoretically, it allows a total of approximately 4 billion $= 2^{32}$ addresses. At the time IPv4 was introduced the possible number of addresses IP could seem to be sufficient. Currently in order to solve the IP address exhaustion problems the concept of subnet or subnetwork is used widely. A subnet (short for "subnetwork") is an identifiably separate part of an organization's network. Typically, a subnet may represent all the machines at one geographic location, in one building, or on the same local area network (LAN). Having an organization's network divided into subnets allows it to be connected to the Internet with a single shared network address. However, it is predicted that in the near future the number of IPv4 address will be exhausted. Already now some problems due to the shortage of number of addresses are starting to appear. Another problem of IPv4, due to its lack of flexibility, is the security issues. Since the definition of IPv4 itself does not have functionalities for security, a new extension IPSec had to be introduced. Such problems leaded IETF to introduce IPv6. From the current perspective, IPv6 resolves the problems that IPv4 currently possesses. However, since IPv6 still has the structure of binary sequences with fixed size, it may cause similar problems to that IPv4 has caused. Moreover, in order to increase the number of the possible addresses, the size of the IPv6 header became twice that of IPv4; which somewhat violates the notion of compact sizing of binary-based protocol design paradigm.

The binary paradigm and the plain text-based paradigm were adapted in different areas of protocols and usually they are influenced by the *"smart terminal and simple network"* notion of the Internet and they concentrated on different issues of protocol designs. Although they have distinct advantages and disadvantages, it is not possible to conclude which is better because the purposes of protocols differ.

5.2 Binary Hybrid Protocols

The binary-based and text-based paradigms have their own roles in the protocol stack implementation. Now the question arises what if the content of a protocol is interpreted by both relaying entities and the end nodes, and what if the content of a protocol is not necessarily entirely visible or readable to the user. Usually the protocols at the higher layer are not interpreted by the relaying entities of the network and they contains the data usually directly visible to the user. However, the protocols at the lower layer are not usually required to be visible or directly understandable to the users.

Two protocols, Session Initiation Protocol (SIP) and Session Description Protocol (SDP), are classified as higher layer protocols and the contents of them are needed to be examined by the network entities and the contents are not necessarily visible or readable to the users at all times.. A user's action such as typing in the URI of callee contributes to the generation of the SIP and SDP datagrams. However, the overall contents are seldom looked up by the user, and the network entities such as CSCF[2] needs to read the contents of the messages. Yet, the structure of SIP and SDP is solely based on plain text, because the protocol is required to maintain a certain level of flexibility and scalability in order to cope with any type of multimedia session. The type of multimedia session tends to evolve radically during a short period of time. Moreover, the contents of the protocol varies significantly from one multimedia session to another.

From the perspective, if the entire contents of SIP and SDP is seldom useful to users, the binary-based paradigm could have been the suitable choice for the design. On the other hand, since these protocols contain data which are hardly of fixed size and the contents vary significantly, the text-based paradigm seems to be the appropriate choice. These two perspectives motivated the idea of a paradigm of protocol design which combines both.The resultant protocols are expected to be smaller in size than the corresponding plain text-based protocol while maintaining relative scalability for possible extensions of the protocol, hence combining the advantages of binary and text based protocol design paradigms.

The paradigm is referred as *"Binary Hybrid Paradigm"*. As shown in the subsequent sections, the structure of the protocol consists partially of binary sequence and partially of plain text.

5.2.1 Binary Hybrid SIP (BHSIP)

The size of SIP messages has always been problematic when they are carried over narrow bandwidth links. For instance, let us consider the SIP message exchange illustrated in [2] for 3G specific multimedia session setup phase. The size of this message becomes problematic as it ranges from a few hundred bytes to thousands of bytes, while the available bandwidth (data transfer rate) over the air interface is rather restricted (usually of the order of a few tens of kilobits per second).

One of the primitive methods to reduce the size of a message, as defined in [49], is

[2]Call Session Control Function is a logical entity of IP Multimedia Subsystem of UMTS; see Chapter2

to use abbreviated forms of header field names. For example, the header field name,such as *Call-ID* header, can be replaced with a single character *i*. However, it only applies to the limited number of headers, hence it does not reduce the overall SIP?SDP message size significantly. The names of the header fields need not necessarily be visible to the user, while the contents of the header may be used for user display for feedback. For example, let us consider the line 8 of Example 5.1, the *From* header. The name *From* may be recognized only by the application program and the program may need to display the contents of the header, *¡sip:user1_public1@home1.net¿*, which is the address of the caller In other words, only the application, which scans the SIP message is required to understand the names of the headers. Naturally, adapting binary sequences for the all the possible header names can reduce the overall size. Moreover, if the size of the binary sequence for the header names are long enough, any new extended headers can have their own binary sequences for identification, thus maintaining a certain level of extensibility.

Example 5.1: SIP INVITE message

```
1:  INVITE tel:+1-212-555-2222 SIP/2.0
2:  Via: SIP/2.0/UDP [5555::aaa:bbb:ccc:ddd]:1357;
    comp=sigcomp;branch=z9hG4bKnashds7
3:  Max-Forwards: 70
4:  Route: <sip:pcscf1.visited1.net:7531;lr;
    comp=sigcomp>,<sip:scscf1.home1.net;lr>
5:  P-Preferred-Identity: "John Doe"
    <sip:user1_public1@home1.net>
6:  P-Access-Network-Info: 3GPP-UTRAN-TDD;
    utran-cell-id-3gpp=234151D0FCE11
7:  Privacy: none
8:  From: <sip:user1_public1@home1.net>; tag=171828
9:  To: <tel:+1-212-555-2222>
10: Call-ID: cb03a0s09a2sdfglkj490333
11: Cseq: 127 INVITE
12: Require: precondition, sec-agree
13: Proxy-Require: sec-agree
```

continued on next page

continued from previous page

```
14: Supported: 100rel
15: Security-Verify: ipsec-3gpp; q=0.1;
    alg=hmac-sha-1-96; spi=87654321; port1=7531
16: Contact: <sip:[5555::aaa:bbb:ccc:ddd]:1357;
    comp=sigcomp>
17: Content-Type: application/sdp
18: Content-Length: 941

                    — SDP omitted —
```

Example 5.1 shows the SIP INVITE message depicted from 3GPP TS 24.228. Line numbers are deliberately inserted to aid explanation although they are not used in the real SIP message. Line 1 starts from the type of message, INVITE, followed by the address of the callee. The address can be of any type, e.g., telephone number, e-mail address or SIP addresses etc. The last term *SIP/2.0* identifies the contents of the datagram as SIP version 2. The *via* field records the address of the network entity the message traverses and what kind of protocol is used to carry the information. Each network entity that receives the message, typically relaying nodes (proxy servers), appends its information to this field. This information is mainly used to detect the loop in the routing path of the message and to guide the response from the callee to follow the same route as the request used. Using the same path by the request and the response violates normal rule of the IP routing algorithm, which transmits the IP packet along the best path currently available. However, this feature allows the network operator to monitor the message flow between caller and callee, hence the calling charge can be made easily. The *Max-Forwards* header in line 3 performs a similar role to the *Time to Live* field of the IP header. In this example, after routing the message 70 times the packet gets destroyed. The *Route* field in line 4 forces the message to pass through a specific server so that the network can monitor all the traffics. Line 5 defines the preferred identity of the caller, which can "pop up" on the callee's display [36]. It can be regarded as a more sophisticated form of "caller ID" services because the caller can use several identities. Line 6 shows the information of the network from which the caller is currently transmitting this message [23]. *Privacy* defines the privacy policy that caller wish to use. Lines 8 and 9 are the actual addresses of caller and callee. *To* field can be the same as the address typed in line

1, however, the address in line 1 can be changed by the relaying proxy if the message needs to be redirected to another destination, whereas *To* retains the original value. *Call-ID* field uniquely identifies the dialog of the message. That is, it identifies a particular invitation and all of the subsequent transaction of SIP messages related to that invitation. It is important, since one terminal can initiate multiple sessions at any given time. *Cseq* defines the message type and the number which are used to match the response with the request in the same dialog (or transaction). Line 12 defines a prerequisite that the callee needs to accept this session, whereas line 13 defines the requirements of the relaying proxies. Lines 12 and 13 are mainly used to negotiate the security mechanisms needed to be used. In line 14, the *"Supported"* field enumerates all the extensions supported by the UAC and UAS. It contains a list of option tags, described in Section 19.2 of SIP specification [49]. Line 15 includes the information about the security scheme [9]. *Contact* header show the address where the caller can be directly reached. This information is used upon the completion of the invitation, so that the actual media datagram can be directly sent to the caller. Lines 17 and 18 define the contents of the additional information SIP carries. In this example SIP carries the SDP datagram, which is application layer and the size of SDP content is 941 bytes. Because SIP carries SDP as application layer payload, some researchers classify SIP as the session layer protocol of OSI structure and others classify both as application layer protocols.

The example illustrates how much information SIP can carry with the advantages of flexibility. The contents of each field do not have any restrictions on the size. Each field name uniquely identifies the use of contents. The penalty of such a luxury is the overall size of a SIP message.

SIP and SDP messages must be exchanged before commencing an actual session, which means it only has a small window of time available to complete the exchange. Since a 3G mobile network can only provide a fraction of the data transfer speed compared to the speed of Internet, the size of SIP introduces a serious problem. SigComp discussed in Chapter 4 does manage to reduce the size of each SIP and SDP message, however, SigComp itself has the disadvantage of complicating the system.

Example 5.2: Binary Hybrid SIP INVITE message

```
1:  0x01 0xff 0x03 +1-212-555-2222
2:  0x2a 0xff UDP [5555::aaa:bbb:ccc:ddd]:1357;
```

continued on next page

continued from previous page

```
    comp=sigcomp;branch=z9hG4bKnashds7
3:  0x16 70
4:  0x22 <sip:pcscf1.visited1.net:7531;lr;
    comp=sigcomp>,<sip:scscf1.home1.net;lr>
5:  0x31 "John Doe" <sip:user1_public1@home1.net>
6:  0x35 3GPP-UTRAN-TDD;
    utran-cell-id-3gpp=234151D0FCE11
7:  0x42 none
8:  0x14 <sip:user1_public1@home1.net>; tag=171828
9:  0x27 <tel:+1-212-555-2222>
10: 0x08 cb03a0s09a2sdfglkj490333
11: 0x10 0x0000fe 0x01
12: 0x20 precondition, sec-agree
13: 0x1d sec-agree
14: 0x25 100rel
15: 0x2f ipsec-3gpp; q=0.1; alg=hmac-sha-1-96;
    spi=87654321; port1=7531
16: 0x0a <sip:[5555::aaa:bbb:ccc:ddd]:1357;
    comp=sigcomp>
17: 0x0f application/sdp
18: 0x0e 941

                — SDP omitted —
```

The binary hybrid form of the SIP message from Example 5.1 is shown in Example 5.2. It has exactly the same meaning as the original SIP message. Hexadecimal numbers represent the portion of the header that is written in binary sequence rather than plain text. Binary Hybrid SIP (BHSIP) exhibits the same structure as normal SIP message and the design paradigm is quite simple. It re-defines all the reserved words as binary sequences. In the case of line 1, instead of including the phrase SIP/2.0 to identify SIP messages, it reserves the first byte to represent the type of message (e.g., INVITE, REGISTER, etc.) and the next byte to be padded with ones to uniquely identify BHSIP message. The start-line,

each message-header line, and the empty line are terminated by a carriage-return line-feed sequence (CRLF) in SIP [49]; BHSIP follows the same notation. Each header field starts with a binary identifier of 1 byte, which replaces text based names. In terms of size, it is equivalent to assigning one letter field name for every header of SIP, but colon, ":", which is required in SIP to end the header field name, is not needed, because the SIP interpreter already expects contents of the header after the one byte header identifier. Hence, the size of BHSIP is always smaller than that of SIP.

In particular, headers, such as *Cseq* of line 11 have fixed format. BHSIP redefines these header fields completely with binary sequences. RFC 3261 [49] defines *Cseq* field by starting with a field name followed by sequence number and request type. The field name is substituted by binary sequence that uniquely identifies this field. According to [49], the sequence number can not exceed 2^{31}, hence BHSIP defines uses 32 bit or 4 byte binary sequence. As for request type, every request name has its own binary representation. For example, the phrase INVITE is defined as binary representation of 0x01.

For the response messages, for example *"100 Trying"*, the definition of header fields are the same as that of the request messages. However the first line differs. The first line of response always starts from the type of the response. The following example is the first line of the *"100 Trying"* message.

```
1: SIP/2.0 100 Trying
```

From the first phrase, "SIP/2.0", the parser acknowledges that the message is a SIPv2 response. The portion represents the type of response in numeric format. The number is meant to be 3 digits and the most significant digit (1 in this example) represents the status of the response, and remaining digits are for a more detailed description of the response. As defined in [49], the number *"100"* is a response that tells the UAC that the request is received by some form of UAS and currently is in some of progress. The next text phrase, *"Trying"* is only for illustrative purpose, telling possibly the user that the request is in progress. This format of the first line is always used regardless of the type of response. The approach applied in BHSIP is to redefine the line in the form shown below.

```
1: 0xff 0x01 0x00 Trying
```

As for the request line, the phrase "SIP/2.0" is replaced by 8 bits of ones. The numeric portion is redefined by two byte long sequences. The first byte of the numeric portion shows the type of the response. For example, *"0x01"* for informational responses, *"0x02"* for success, *"0x03"* for redirection, *"0x04"* for client error, *"0x05"* for server error, and *"0x06"*

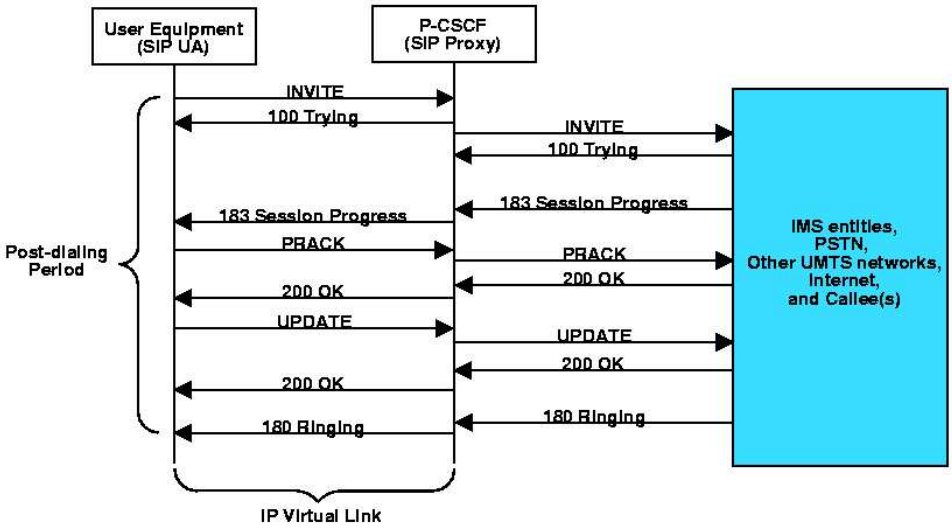

Figure 5.2: Detailed SIP Signal Exchange in 3G networks

for global error. The second byte of the numeric portion, is used to show the detailed description. The plain text *"Trying"* is still added, even though it is arguably of less use.

The message exchange sequences defined by 3GPP are shown in Figure 5.2. They are deliberately abbreviated to emphasize the message exchanges between the mobile terminal and the P-CSCF of the currently attached network (domain) during the post-dialing phase. The sizes of SIP messages shown in Table 5.3 are the sizes of the SIP portion of the message example given in the specification of 3GPP [2], which follows the message exchange sequences shown in Figure 5.2. The corresponding BHSIP messages have exactly the same meaning as the original SIP messages.

For this particular example, approximately 22% of size reduction was observed, without sacrificing the performance of plain text-based SIP. The rate of reduction is highly dependent on the message itself. However, it is obvious that the size of BHSIP is always smaller while having all the privileges of plain text-based protocols.

The software, which interprets the content of BHSIP only needs to scan the first byte to identify the headers. For normal SIP, the interpreter should read all the characters until it encounters colon ":", which marks the end of the header field name, and then a string matching should be invoked in order to identify the header field. However, the header of BHSIP can be identified by the matching of one byte, instead of matching multi-byte long string. Hence, the process time of BHSIP in contrast to that of SIP is expected to be faster

Type	SIP	BHSIP
INVITE	753	551
Trying	243	191
Session Progress	737	613
PRACK	596	473
OK (PRACK)	282	216
UPDATE	679	524
OK (UPDATE)	283	215
Ringing	466	385
Total Size	4039	3168

Table 5.3: Size of SIP and BHSIP messages (in bytes) excluding SDP

although it may not be significant for the faster machines.

Section 25 of the SIP specification [49], defines the SIP structure using Augmented Backus-Naur Form (ABNF) defined in RFC 2234 [14]. The ABNF form is still applicable to BHSIP, since the structure of both protocols are identical, while BHSIP uses binary sequences instead of plain text in the header name and some contents of the header fields.

5.2.2 Binary Hybrid SDP (BHSDP)

SDP is defined to be carried within the payload of SIP during 3G multimedia session setup phase. SDP does not include any transport mechanism or any kind of parameter negotiation. The contents of SDP is simply a chunk of information that the system can use to initiate a multimedia session among multiple users. It includes, for instance, session name, IP address, port numbers, and times and dates when session is active. Even though SDP does not provide a negotiation mechanism, including SDP within SIP datagram can implement session related information negotiation betweens participating parties by the three-way hand shaking mechanism of SIP. SDP is also plain text based protocol, however the structure is much simpler than that of SIP.

Example 5.3: SDP portion of SIP INVITE message

```
1:  v=0
2:  o=- 2987933615 2987933615 IN IP6 5555::aaa:bbb:ccc:ddd
3:  s=-
4:  c=IN IP6 5555::aaa:bbb:ccc:ddd
5:  t=907165275 0
6:  m=video 3400 RTP/AVP 98 99
7:  b=AS:54.6
8:  a=curr:qos local none
9:  a=curr:qos remote none
10: a=des:qos mandatory local sendrecv
11: a=des:qos none remote sendrecv
12: a=rtpmap:98 H261
13: a=rtpmap:99:MPV
14: m=video 3402 RTP/AVP 98 99
15: b=AS:54.6
16: a=curr:qos local none
17: a=curr:qos remote none
18: a=des:qos mandatory local sendrecv
19: a=des:qos none remote sendrecv
20: a=rtpmap:98 H261
21: a=rtpmap:99:MPV
22: m=audio 3456 RTP/AVP 97 96 0 15
23: b=AS:25.4
24: a=curr:qos local none
25: a=curr:qos remote none
26: a=des:qos mandatory local sendrecv
27: a=des:qos none remote sendrecv
28: a=rtpmap:97 AMR
29: a=fmtp:97 mode-set=0,2,5,7; maxframes=2
30: a=rtpmap:96 G726-32/8000
```

continued on next page

continued from previous page

```
31: m=audio 3458 RTP/AVP 97 96 0 15
32: b=AS:25.4
33: a=curr:qos local none
34: a=curr:qos remote none
35: a=des:qos mandatory local sendrecv
36: a=des:qos none remote sendrecv
37: a=rtpmap:97 AMR
38: a=fmtp:97 mode-set=0,2,5,7; maxframes=2
39: a=rtpmap:96 G726-32/8000
```

SDP contents shown in Example 5.3 is obtained from [2] and it is to be appended to the INVITE example shown in the previous subsection. Header *v* specifies the version number of the protocol. The current SDP version number, according to line 1, is *0*. Since the software that interprets this SDP message already has a clue from the SIP message, which carries the SDP message, that the contents is SDP (see line 17 of Example 5.1). The *o* field in line 2 gives the originator of the session (their user name and the address of the user's host) plus a session id and session version number. In this case, the user name is defined with "-" character meaning the the host does not support the concept of user IDs. The first string of numbers represents the session identification and the second sequence does for version of this announcement. The version is needed for proxy announcements to detect which of several announcements for the same session is the most recent. The key word, *IN*, appears at lines 2 and 4, represents the network type of the Internet and the *IP6* identifies the addressing used IP version 6. The *o* field completes with the actual address. Line 3, *s* header, represents the name of the session and "-" is used for no name. *c*, at line 4 field includes connection information and the format of addressing follows the *o* field. Line 5, *t* field, specifies the time when the session starts and terminates. The values are the decimal representation of Network Time Protocol (NTP) time values in seconds [39]. Since the second value, the stop-time, is set to zero, the session is not bounded, though it will not become active until after the start-time. If the start-time is also zero, the session is regarded as permanent. From line 6 to line 13, SDP defines two video codecs that can be used for this multimedia sessions and from line 22 to line 39 defines two audio codecs. The detailed description of this codec specification is beyond the scope of this project, hence omitted, however the reader can refer

to [27] for the description.

The content of SDP is more predictable than that of SIP as most of word phrases used within SDP are reserved words. Following the same notion of protocol design of BHSIP, SDP is redefined as Binary Hybrid HSDP (BHSDP) in Example 5.4.

Example 5.4: BHSDP alternative of Example 5.3

```
1:  0x01 0
2:  0x02 - 2987933615 2987933615 0x01 0x12 5555::aaa:
    bbb:ccc:ddd
3:  0x03 -
4:  0x23 0x01 0x12 5555::aaa:bbb:ccc:ddd
5:  0x11 907165275 0
6:  0x21 0x02 3400 0x06 98 99
7:  0x24 0x02 54.6
8:  0x26 curr:qos local none
9:  0x26 curr:qos remote none
10: 0x26 des:qos mandatory local 0x16
11: 0x26 des:qos none remote 0x16
12: 0x26 0x09 98 H261
13: 0x26 0x09 99:MPV
14: 0x21 0x02 3402 0x06 98 99
15: 0x24 0x02 54.6
16: 0x26 curr:qos local none
17: 0x26 curr:qos remote none
18: 0x26 des:qos mandatory local 0x16
19: 0x26 des:qos none remote 0x16
20: 0x26 0x09 98 H261
21: 0x26 0x09 99:MPV
22: 0x21 0x01 3456 0x06 97 96 0 15
23: 0x24 0x02 25.4
24: 0x26 curr:qos local none
25: 0x26 curr:qos remote none
```

continued on next page

continued from previous page

```
26: 0x26 des:qos mandatory local 0x16
27: 0x26 des:qos none remote 0x16
28: 0x26 0x09 97 AMR
29: 0x26 0x24 97 mode-set=0,2,5,7; maxframes=2
30: 0x26 0x09 96 G726-32/8000
31: 0x21 0x01 3458 0x06 97 96 0 15
32: 0x24 0x02 25.4
33: 0x26 curr:qos local none
34: 0x26 curr:qos remote none
35: 0x26 des:qos mandatory local 0x16
36: 0x26 des:qos none remote 0x16
37: 0x26 0x09 97 AMR
38: 0x26 0x24 97 mode-set=0,2,5,7; maxframes=2
39: 0x26 0x09 96 G726-32/8000
```

BHSDP uses reserved words of binary sequences, instead of plain texts. One important thing to notice is that the same binary sequence can be used for different reserved words, as long as they are not expected to appear in the same position of the header. For example, when the sequence "0x01" appears at the start of line, it represents header name *"v"* (see Line number 1), whereas when the same sequence appears after header "0x23", *"c"* header, it represents the phrase *"IN"*.

Table 5.6 shows the size reduction gained by using BHSDP. In this case, the overall reduction in size of approximately 30% was observed. As for BHSIP, BHSDP is expected to be always smaller than normal SDP, since all the multi-byte reserved phrases are re-defined in binary form of 1 byte.

All the reserved binary sequences for BHSIP and BHSDP are included in Appendix B.

5.3 Binary Hybrid Protocols and SigComp

SigComp introduced in Chapter 4 is a text compression scheme, adapted widely where the link between certain nodes suffer from a relatively slow data transfer rate. As far as the 3G air interface is concerned, the data transfer speed is far behind that of the current wired

Type	SDP	BHSDP
INVITE	941	659
Trying	0	0
Session Progress	440	309
PRACK	385	273
OK (PRACK)	412	290
UPDATE	389	270
OK (UPDATE)	393	267
Ringing	0	0
Total Size	2960	2068

Table 5.6: Size of SDP and BHSDP messages (in bytes) carried within SIP messages

Internet. The results in Chapter 4 illustrate how SigComp effectively reduces the size of the SIP/SDP datagram while sacrificing the CPU and storage resources of communicating entities, such as UE and P-CSCF.

The problems of SigComp come from the fact that it uses an initial dictionary and the updating should be performed basically for each SIP/SDP message exchange. The approach used in the case of SIP and SigComp is that SIP retains all the text structure as it is defined in [49]. The compression uses the initial dictionary, which contains most of keywords defined when the dictionary was proposed. Usage of such dictionary for initial compression ensures the maximum initial compression ratio. Since the initial dictionary strictly prohibits any modification, the initial compression may not achieve the maximum compression ratio if many new extension headers are proposed and widely used. The restriction on the modification of the dictionary is essential to maintain the compatibility of communicating parties. However, it also restricts extension of the protocol. In order words, the initial compression can only become worse as more extensions of SIP are introduced and used.

The compression algorithms, which use dictionary, replace the phrases in the original text with a binary address, which points to the identical phrase in the dictionary, Hence, the compression is achieved. In order to track a large dictionary, the addresses of phrase become larger. Moreover SigComp is not aware of the contents of SIP. BHSIP, on the other hand, uses usually 8 bit (1 byte) binary sequences to identify the keywords, hence these

keywords are in some sense already in compressed form or somewhat optimized in terms of size. Additionally, the same binary sequence can be used to represent different keywords, since the BHSIP interpreter knows that the first binary sequence in the line represents the name of the header field and the other binary sequence represents the contents of the header fields. For example, binary sequence *0x01* presents *"INVITE"* message, *"Accept"* header name and so on. However, they appear in the datagram in different places. Number *0x01* for *"INVITE"* appears in the first line of the message and as a contents of *"Cseq"* header field, whereas *0x01* for *"Accept"* only appears at the start of line, which is not the first line of the message.

Type	BHSIP/BHSDP	Static SigComp	Dynamic SigComp
INVITE	1210 (1694)	510 (574)	510 (574)
Trying	191 (243)	163 (163)	28 (43)
Session Progress	922 (1177)	504 (533)	270 (276)
PRACK	746 (981)	423 (467)	67 (90)
OK (PRACK)	506 (694)	318 (331)	41 (45)
UPDATE	794 (1068)	468 (509)	63 (61)
OK (UPDATE)	482 (676)	306 (322)	33 (73)
Ringing	385 (466)	245 (244)	36 (37)
Total Size	5236 (6999)	2937 (3143)	1048 (1199)

Table 5.7: BHSIP/BHSDP message sizes (in bytes)

As discussed in Chapter 4, BHSIP and BHSDP can be compressed by static and dynamic SigComp. Table 5.7 shows the size of the datagrams. The values in parentheses are corresponding SIP/SDP values. Using binary hybrid protocols obviously has the benefit of reducing the compressed datagram even further. The overall size reduction of dynamic SigComp BHSIP/BHSDP compared with original non-compressed SIP/SDP is 85%, which is a further reduction of approximately 2% if compared to dynamic SigComp SIP/SDP, without sacrificing any data integrity.

5.4 Conclusion

Binary-based protocol and text-based protocol exhibit their own advantages and disadvantages. However, they are well suited to the their roles in the network protocol stacks. For example, designing an IP protocol with plain text would significantly degrade the response time of the network, and binary based HTTP will not be able to display contents easily to a user. Hence, it can be concluded that the lower level protocols should follow the binary based paradigm to have a protocol that is compact and easy to scan. In the case of the application layer, designing an entirely binary protocol restricts the performance of the user-end software and leads to a lack of flexibility that the application layer needs most. However, it is not a feasible solution to define an entire protocol with plain text in some cases, such as SIP and SDP. The contents of SIP/SDP need to be examined by software rather than the user, hence the whole contents do not necessarily need to be readable by a user. Simple redefinitions of such protocols can improve the message exchange time due to the smaller datagram and the processing can be relatively faster. In addition, by retaining some of contents in plain text form, it provides all the required flexibility and the information that may be displayed to a user can easily be extracted from the protocol.

Binary hybrid design paradigm, if it is applied to a 3G network, can reduce the work load of SigComp layer mentioned in Chapter 4. The compression dictionary does not require to have key phrases which are already defined as binary sequences in BHSIP and BHSDP. In addition, the dictionary updating information, which is gathered by the received message, is smaller, since the sizes of BHSIP and BHSDP are smaller than the sizes of SIP and SDP and the compressor and decompressor of the layer need to scan smaller messages. Improvement of the SigComp layer with BHSIP and BHSDP means that a mobile terminal can use additional memory for other applications and power saving can be achieved. However, binary hybrid approach does have its limitation. Since, most of the binary sequences are 8 bits long; only $2^8 = 256$ distinct values can be used. For example only 256 header fields and 256 types of requests can be used.

The Carriage Return Line Feed (CRLF) (16 bit binary sequence, *0x0d0a*) or simply *"0x0a"* is required to mark the end of a single header field. The duplication of the binary sequence between such line termination sequence and the binary sequences defined for of BHSIP and BHSDP is not expected to cause any problem. The parser, which is responsible for interpreting the BHSIP and BHSDP messages, will expect the next header name to appear

only after line terminating sequence (CRLF) is encountered by the parser.

Chapter 6

Testbed Implementation

This chapter present the details of the testbed implementation used to investigate the session failure rate and the control message exchange delay incurred due to the wireless channel conditions using the virtual link defined Chapter 3. The results of the experiments are presented in Chapter 7.

6.1 Network Emulation

For modern scientific experiments, two major paradigms are widely used. One that is most obvious based on real experiments. In order to fully conduct real experiments, the testbed should be equipped with all the necessary physical components. For this project, this would mean an access to a fully functional mobile terminal with proper wireless modulation and error detection/correction schemes implemented, a real base station (Node-B) and radio access controller (RAN) and other network entities (e.g., CSCFs) must be used. Such experiment could be expensive, and access to the network entities are not permitted for the project.

The other paradigm, simulation, is more cost-effective, since real network entities are not required. However, this paradigm itself may introduce faulty results unless proper precautions are taken during the modeling phase of all significant simulation-related factors. In order to fully implement the simulation, researchers must model every necessary entity of the network usually by some sort of statistical estimations. Any mis-modeling or disregarding of certain factors may cause serious error in the results.

In this project, we used computers connected within Local Area Network, hence, in this sense, it is a experimentation. However, those computers are not the 3G specific network

entities. Rather, they are pretending to be the 3G network entities by executing programs which mimic the behaviors of the 3G network entities. Since the imitating the behaviors of network entities and the link between them involves the simulation of the real network entities, this approach can be viewed as a mixture of experimentation and simulation. Such approach is called emulation. In general, the emulation uses real physical entities. An entity can be real network nodes, such as routers and gateways, or the entity, which can be implemented with some special software, which controls the behavior of that entity, resembling its certain role. The advantages of the emulation is that it allows the use of real (or pretending to be real) network entities, and the entities can act as other network entities by resembling the behavior, modeling them in situation. Hence, emulation can reduce the complexity of a simulation model while providing relative cost-effectiveness compared to real experiments.

The virtual link discussed in Chapter 3 is defined with the concept of effective E_b/N_0 of the wireless channel. In order to estimate the values of effective E_b/N_0 according to the scenarios defined in Chapter 3, we implemented the fading channel simulator defined in section 3.2.3 using a commercial package MATLAB Release 13 [38]. The reason for choosing this package was that it has all the necessary mathematical functions pre-implemented and the coding process can be relatively easily done since it has scriptural language supports. The values of the effective E_b/N_0 is then used to compute the code block loss rate[1]. To speed up the process of computing the values of the code block loss rates for various cases (two convolutional encoders, four different interleaving TTI, three terminal moving speeds and various size of code block), we implemented dedicated C programs to computer $(E_b/N_0)_{eff}$.

The results presented in this thesis are obtained from the testbed which uses emulated applications and emulated UE to P-CSCF link discussed in Chapter 3. More information of the testbed configuration and the emulation programs used are presented in later sections. Since, the experiments are to be conducted with an emulated wireless link (UE to P-CSCF link), some available network link emulators were considered as candidates for implementing the virtual link, and the survey of those emulators follows..

[1]Code block is a data unit. It is an input unit for the convolutional encoder, the size can be chosen by the application; see Chapter 2 for more detail.

6.1.1 Seawind

Seawind is a wireless network emulation package implemented by Department of Computer Science, University of Helsinki, Finland [53]. The package is freely available, however, a license must be acquired. The basic operation of Seawind is to intercept the traffic flow [2] between the client and the server transparently to the end-point hosts. The desired link characteristics are emulated by delaying, dropping and modifying packets in the flow. The client and the server can be directly connected to Seawind (e.g., by a serial cable), or they can be located anywhere in the network. The researches involve Seawind were related to the performance of the TCP/IP protocols and the behavior of TCP applications [53].

The basic architecture of Seawind consists of three entities. They are *Mobile Host (MH)*, *Emulation Host (EH)* and *Remote Host (RH)*. The core of the network condition emulation is governed by the *Simulation Process (PS)* residing inside of EH. PS causes delays and packet packet losses to emulate the target link. The status of three entities can be monitored by a simple Graphical User Interface (GUI) and controlled by *Control Tool (CT)*. MH and RH are implemented with actual protocol stacks so that real applications can exchange data via the emulated network environment controlled by EH. Seawind is equipped with a protocol filter, which performs different task for different protocol datagrams. Moreover, it is capable of generating background traffic, and several types of random distributions are implemented to model certain types of link.

Upon the completion of the experiments, Seawind generates two types of log outputs. The first type of log, *Filter Log* is generated by SPs. It contains information about the packets that have traversed through Seawind. Since SP governs protocol filter and protocol filter output is the essential information about the packet, the filter log carries information dependent on the type of protocol protocol filter is responsible for. Another log is called *Seawind Log*. It contains Seawind-specific information of the test run, for instance the time stamp for the packets.

Seawind has attractive features. Firstly, it enables different operations (e.g., delays drops) on different type of data packets. Secondary, it generates logs for further data analysis. However, on March 2003 when this project commenced, Seawind could only support TCP/PPP based connection between network entities. Because in most wireless links the SIP/SDP messages are expected to use UDP and with the current department setup[3] use of PPP would

[2] datagram exchange

[3] The departmental setup refers to the setup of the computer networks and the terminals in the department

require new setup of the departmental setup of machines, it was decided not to use Seawind.

6.1.2 RAMON (Rapid Mobility Network Emulator)

RAMON has been developed in the department of Computer and Information Sciences and Engineering, University of Florida, USA [29]. This package is ideally designed for investigating mobility issues in a multi-cell environment. It is specifically designed mainly to evaluate hand-off schemes.

The package consists of actual antennas, actual signal sources and emulated wired network environments. The configuration is simple. A wireless signal source generates particular signal, and several omni-directional antenna receives the signal. The signal received by the antenna is attenuated by a commercial hardware. If the distances between the signal source and the antennas are relatively similar and assuming that there are no obstacles in between, the strengths of the received signals are expected to be very similar. However, if the attenuators attenuate the signal level lower than one particular signal received by one particular antenna, the system would regard that the signal source is close to that antenna than the others. Basically this is how RAMON works. Even though the signal source is stationary, changing the signal amplitude by attenuators connected to each antenna, emulates the situation that the signal source is located within the coverage of that antenna (or base station, since the antenna imitates the role of base station). With this setup hand-off scheme can also be evaluated. For example, suppose the antenna 'A' only receives a strong signal (i.e., the attenuator of this antenna does not attenuate the received signal as much as the others). By gradually increasing the signal level of antenna 'B', which is neighbor antenna of antenna 'A', as decreasing the signal level of antenna 'A', RAMON emulates the case when the signal source moves one cell, which is covered by antenna 'A', to another cell, which is covered by antenna 'B' The speed of a moving terminal can also be emulated by how rapidly the signal power decreases by the signal attenuator. The wired links are emulated by NISTnet [42] (more discussions on NISTnet later).

Since, the project is not focusing on hand-off issues, simplified RAMON would have be used with an antenna and a signal attenuator. If the attenuator distorts the received signal, according to the fading characteristics of the 3G air interface, real wireless terminal and a P-CSCF-like entity could have been used. However, since RAMON is not freely available

of Computer Science & Software Engineering, University of Canterbury, Christchurch, New Zealand

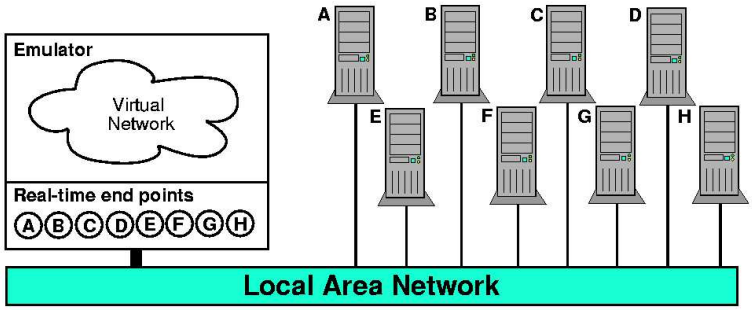

Figure 6.1: Typical connection between IP-TNE and terminals

and without a hand-off scheme investigation RAMON is of less use, it was decided that the emulator was not suitable for the experiments, which we planned.

6.1.3 IP-TNE (IP Traffi c and Network Emulator)

IP-TNE has been developed by the Department of Computer Science, University of Calgary, Canada [55]. The package emulates IP network using discrete event network simulation.

A typical IP-TNE configuration consists of several terminals connected in LAN and all the packets are intercepted by the emulator running on a terminal connected to LAN. Figure 6.1 shows the typical connection of terminals and the emulator package. The hosts, named A–H interact with each other via a virtual network structure simulated by IP-TNE emulator. Each host is connected to the virtual network with real-time endpoint. Hence, the hosts appears to each other as if they are positioned within the virtual network controlled by IP-TNE emulator. The virtual network is controlled by the discrete event network simulation, with the parameters specified to imitate the conditions of the link. The virtual network, as for Seawind, is capable of generating background traffics.. The emulator can be connected to other network entities, such as a web-server located outside the LAN. In this case, it emulates the situation where a terminal or host tries to invoke a procedure in a web-server, which is positioned in a foreign network, located very far (a lot of relaying networks between a web-server and a terminal).

By using real terminals in the department as UE and P-CSCF and IP-TNE emulating the wireless link, the testbed could have been implemented. However, the package was not freely available to use at the time of our project commencement, hence no further information on the package could be obtained and it was unable to use this package.

6.1.4 NISTnet

NISTnet is one of the most well known emulation packages. Literature which discusses previously mentioned emulation packages and others usually refers to NISTnet and its functionalities as the basic reference model of emulators [42]. NISTnet is implemented as a module of the Linux kernel, hence the processing speed is relatively fast if the machine is equipped with relatively fast CPU, so that the additional delay caused by the emulator is usually negligible. It basically works as a router. It reads every incoming IP packet and manipulates (e.g., discards, delays, etc.) the packet if the packet has specific source and destination addresses.

NISTnet provides a simple set of functions, which arguably made this most widely used emulators. Unlike commercially available emulators or previously stated ones, they do not have parameters which are specific for some communications link characteristics. They rather exhibit fundamental characteristics of any data communication links by providing functions that allows for

- holding IP packets within a buffer for a predefined amount of average time with a given variance,
- discarding the IP packet with a predefined probability, and
- emulating a low speed data link by holding incoming packets before relaying them to the application for a certain amount of time, depending on the size of the packet

It is the user's responsibility to decide how those parameters can be translated into the specific link that research is focused on. Many network links have their own characteristics. However, eventually they can be described into very fundamental primitive parameters NISTnet provides. For instance, as discussed in Chapter 3 UE to P-CSCF virtual link are defined by the fundamental functions, therefore, usage of such emulators is possible.

Since the two parameters which control the virtual link defined in Chapter 3 are the data unit loss rate and the data transfer rate of the link, the NISTnet's functionalities exactly match the requirement for implementing the link. Moreover, since it is freely available and the departmental machines are installed with RedHat Linux, NISTnet is chosen for the virtual link implementation needed for this project. NISTnet was installed on two machines, which are specially set up to act as UE and P-CSCF (see Figure 6.4).

6.1.5 DummyNet

DummyNet provides essentially the same functions as NISTnet, but it requires different Operating System (OS) platform. It is currently a part of standard FreeBSD kernel patch in the IP stack [18], [21]. Although the functionalities of the DummyNet enables the configuration, which reflects the virtual link defined in Chapter 3, since it uses the FreeBSD OS rather than Linux [4], we decided not to use this package.

6.2 Testbed Configuration

The testbed consists of two Intel Pentium based machines connected with normal Ethernet interface (Local Area Network of department of Computer Science & Software Engineering, University of Canterbury, Christchurch, New Zealand). Table 6.1 shows the hardware specification of these two machines used for the experiments. Both machines have identical specifications.

CPU type	Intel Pentium 4	**CPU Clock Speed**	2.4 GHz
RAM Size	512 MB	**Physical link**	10/100 Mbps Ethernet
OS type	RedHat Linux 9	**Kernel Version**	2.4.20

Table 6.1: Hardware specification

The measured data transfer speed between the two machines ranged from 8.7 Mbps to 9.2 Mbps, which was mush faster than the data transfer rate of the 3G wireless channel. Such speed is important, because using an emulator to resemble the virtual link is based on the assumption that the implications of actual physical connection of terminals (an Ethernet connection between two Linux machines) are negligible. With such link, the time required to transfer the data of 2000 bytes ranges from $1.3ms$ to $1.4ms$. Since the biggest size of SIP/SDP datagram appended with UDP/IP headers and other lower overheads used in our experiments is always less than 2000 bytes, the assumption that the implication of physical connection is negligible holds.

[4]The department of Compute Science and Software Engineering of University of Canterbury, uses RedHat Linux as a standard

6.2.1 UE and P-CSCF Emulation

Two Linux machines were setup to act as UE and P-CSCF. These nodes were programmed to generate the sequences of datagram that resemble SIP messages as shown in Figure 5.2. The data exchange delay over the link is measured from the time when transmission of the first *INVITE* message begins to the time when *180 Ringing* response is successfully received by the caller.

The nodes used for the experiments were not fully functional SIP entities, rather, they emulated the behavior of UE and P-CSCF. The behaviors of the emulated UE and P-CSCF are described by Figure 6.2 and 6.3. Both machines run applications which are based on Linux Socket. The client and server example depicted from [58] was used to implement the application which sends and receives messages with minor modifications. The retransmission timer of SIP standard [49] was also implemented using `int alarm(int)` function provided in the Linux distribution.

When the Linux box emulating UE (thereafter, we refer to this Linux box as a *dummy UE* and the other as a *dummy P-CSCF*), is started, the sends out N_{block} code blocks of size N_{size} to the dummy P-CSCF, which emulates the INVITE message. The number of code block, N_{block}, exactly matches the number which makes up the original size of the INVITE message. Since the results are expected to produce the distinctions between two SigComp schemes (dynamic and static SigComp; see Chapter 4) using SIP/SDP and BHSIP/BHSDP (see Chapter 5), the value for N_{block} was adjusted for the particular case. The reason for using the code block rather than actual control message is that in the wireless channel the probability of successful transmission of a message depends on the size of that message. In this case of the multimedia control message exchange, the sizes of requests and responses differ significantly. Hence each response and request has different probability of loss when it is transmitted over wireless channel. However, since the size of code block is the same for all messages, the probability of the code block loss is the same for all code blocks. Since, it simplifies the experiment and the network emulator we used, NISTnet, only allows one value for the probability of datagram dropping, the number of code blocks which resembles the original control message was exchanged, instead of sending actual messages.

According to [49], for unreliable transports (such as UDP), the client transaction[5] re-

[5]SIP transaction is based on HTTP-like request/response transaction model, which includes a request and at least on response.

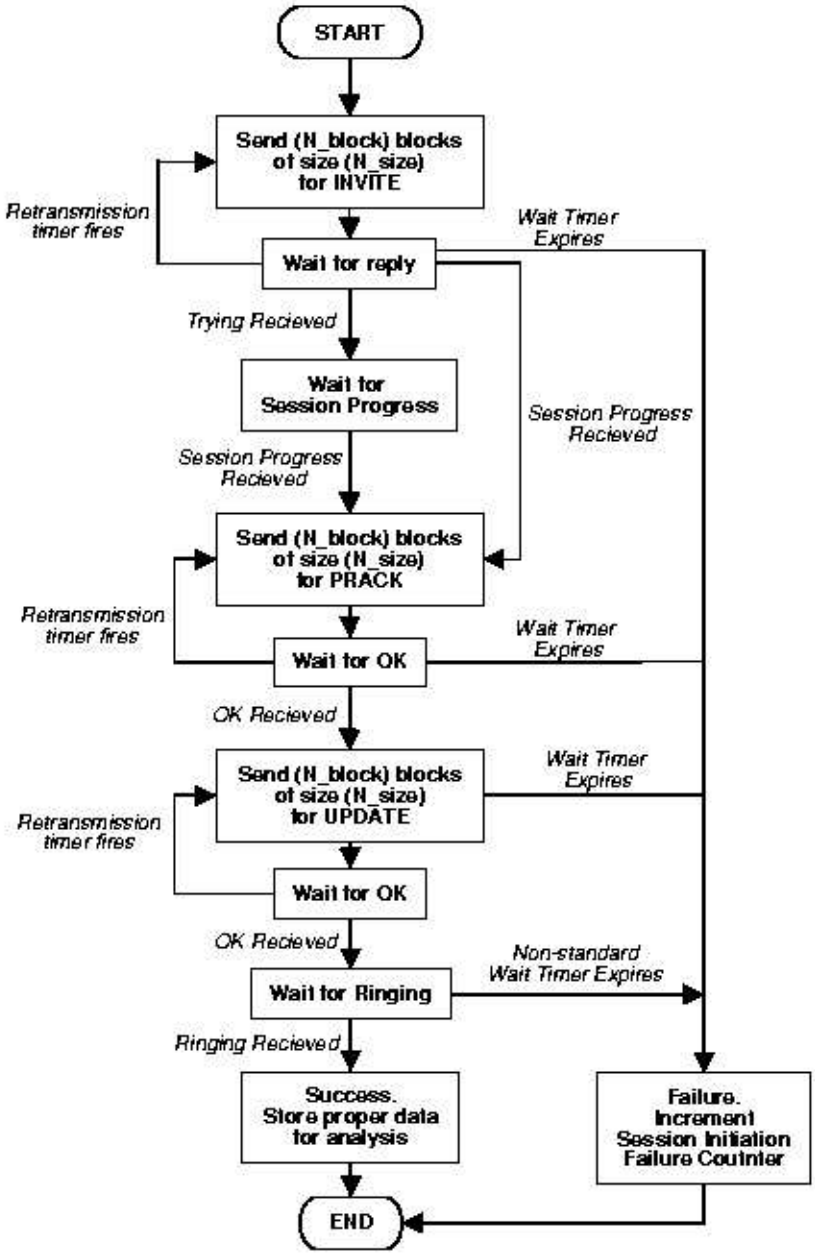

Figure 6.2: State diagram of UE emulation

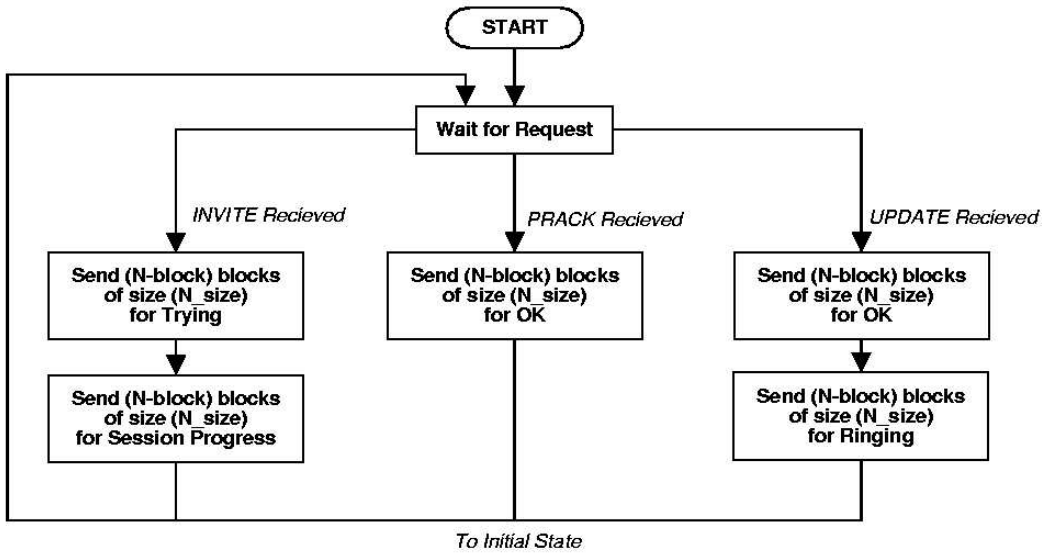

Figure 6.3: State diagram of P-CSCF emulation

transmits INVITE requests at an interval that starts from 0.5 seconds and doubles after every retransmission by default if response is received. After receiving a 1XX response (e.g., *"100 Trying"* or *"183 Session Progress"* message), any retransmissions cease altogether, and the client waits for further responses. Such retransmission is controlled by the *retransmission timer* of a dummy UE. The retransmission terminates when a *1XX* response is not received after 32 seconds from the time when the first INVITE message is generated, and eventually the session initiation attempt fails. This transaction termination timer is referred to as a *wait timer* of the dummy UE. For response messages such as such as *"100 Trying"*, *"183 Session Progress"*, *"180 Ringing"* and *"200 OK"* are sent only once when the request is successfully received. Hence, a dummy P-CSCF does not need any timer mechanisms implemented. Non-INVITE requests, such as PRACK and UPDATE adapts a similar retransmission timer mechanism as INVITE, however the maximum value of the timer is 4 seconds, and the wait timer is also used (32 seconds). This retransmission results in time intervals of 500ms, 1sec, 2sec, 4sec, 4sec, 4sec, etc. If any of client transaction fails, the whole session attempt is assumed to be failed.

The operation of a dummy P-CSCF is somewhat different. It always wait for any requests. Upon receiving any proper requests, it transmits the pre-defined N_{block} code blocks of size N_{size} of appropriate response to the dummy UE.

The actual values for the number of code blocks, N_{block} and the size of each code block N_{size} used in the experiments are shown in Chapter 7 for varying the types of SigComp and the types of control message (i.e., SIP or BHSIP).

6.2.2 Message Exchange Sequence

The SIP/SDP message follows the way of message exchanges depicted in Figure 5.2. In the case of codec negotiation between network entities (between caller and callee or between caller and the network), additional message exchange is inevitable. However, for simpler implementation of experiment, no negotiation with codecs was assumed beyond of what is proposed in the initial INVITE message sent by caller to callee. During the transmission of code blocks if any of code block fails to be received, the whole SIP/SDP message is regarded as lost. In practice, the link between UE and P-CSCF includes several *relay* points, such as NodeB, RNC, SGSN and GGSN (see Figure 2.1). The message loss due to code block loss happens at RNC. However, as far as the UE to P-CSCF virtual link is concerned, a message loss at RNC can be regarded as the loss at the receiving end of the virtual link. Additional delay and message losses while traversing the relaying points are rather negligible, hence no extra attentions was given for such cases. For comparison reasons, the same sequence of message exchanges with BHSIP/BHSDP were also considered under the same assumptions, and the results are shown in Chapter 7.

6.2.3 SigComp Layer

The SigComp layer was implemented using the standard Linux compression library, *zlib 1.1.4*. DEFLATE [16] was used to form compressed messages as discussed in Chapter 4. The implementation of DEFLATE in *zlib 1.1.4* is based on the mixture of static Huffmann coding and LZ77 [20]. DEFLATE is one of the most popular dictionary based compression algorithms. It was originally used in the well-known Zip and Gzip software and has since been adopted by many applications, the most important of which are 1) the HTTP protocol, 2) the PPP compression control protocol, 3) the Portable Network Graphics (PNG) and Multiple-Image Network Graphics (MNG) graphics file formats and 4) Adobe's Portable Document File (PDF) [51]. The reason for selecting this algorithm is that 1) it can handle large dictionary, 2) large look ahead buffer results in the better compression and 3) it uses hash tables for effective and efficient dictionary search. Moreover its implementation, *zlib*

is freely available. Other algorithms may compress differently [61]. However, since the experiment was designed to evaluate the delay during the post-dialing period in general and the compression ratio is more dependent on the contents of the dictionary rather than the capabilities of the compression algorithms, only one type of algorithm was used.

6.2.4 UE to P-CSCF Virtual Link

Radio Access Network (RAN) and P-CSCF are connected by a broadband link, whereas UE is connected to RAN by a narrow band link. Hence, we assumed that the virtual link had a bandwidth equal to that of a wireless radio link. The network emulator from National Institute of Science and Technology, NISTnet [42] was installed and utilized to emulate the characteristics of the virtual link. The virtual link exactly follows the parameters defined and described in Chapter 3.

6.2.5 Testbed Operation

Figure 6.4 shows how the testbed is configured. As a matter of fact, the dummy UE and the dummy P-CSCF do not perform control message (e.g., SIP/SDP or BHSIP/BHSDP) generation and compressions. Only the size of the compressed message to compute N_{block} is used to configure the dummy UE and dummy P-CSCF. Figure 6.4 only emphasizes that the UE and P-CSCF emulations are based on the results from the control message generation and the SigComp layer performance.. The value of N_{size} was varied to identify the implication of the code block size.

Note that the NISTnet embodies the virtual link between UE and P-CSCF and the physical connection between two Linux boxes are of no interest and the influence of this connection is assumed to be negligible

The orders of the operation of testbed are as follows.

1. The control messages were compressed by SigComp dynamically and statically.

2. The compressed size of control message and the size of overheads, and the size of each code block N_{size} decided the number of code blocks N_{block} needed to be transmitted in order to construct realistic message exchanges.

3. Actually, UE and P-CSCF emulators exchanged the number of IP packets which resembled the code blocks in terms of the size. For example is code block is set to be

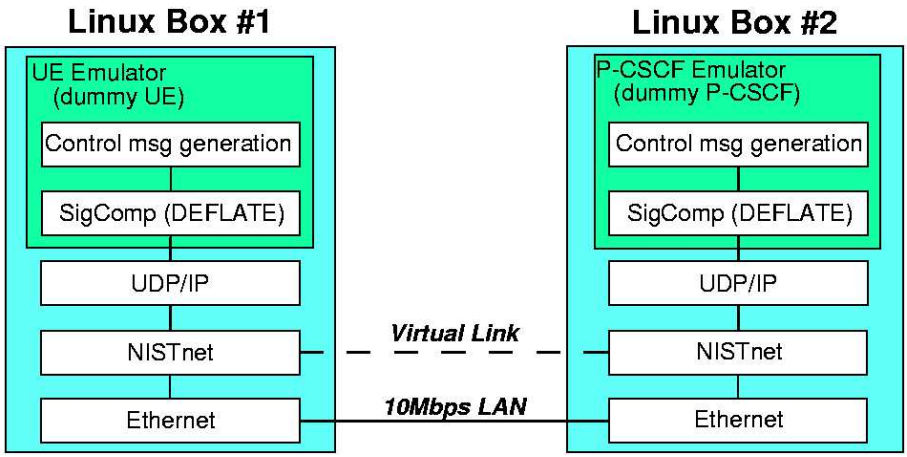

Figure 6.4: Overview of experimental setup

504 bits, then the IP packet including any overheads above and including IP layer must be in that size.

4. Each IP packet, which resembled a code block,is generated and received by a dummy UE and a dummy P-CSCF.

5. Each IP packet, which resembled a code block, was intercepted by NISTnet to delay it by a certain amount of time to emulate a slow link and randomly discarded, to emulate a lossy channel condition.

Chapter 7 presents the experimental results which were obtained by varying the size of the code block N_size and other control message related schemes. The measurements were recorded in terms of the delay measured to complete message exchange and the number of attempt failures over a certain number of trials.

Chapter 7

Performance Evaluation of Multimedia Session Setup Schemes

This chapter presents the results from the performance studies of the multimedia session setup procedure defined by 3GPP specification, TS 24.228 [2] during the post-dialing period. The main concern of the experiments was to investigate the performance of the session initiation scheme over a UMTS wireless channel. The session failure and the control message exchange delays due to the wireless channel condition are the issues discussed in this chapter. All the experimental results shown in this chapter and Appendices C, D and E are obtained using the testbed discussed in Chapter 6.

7.1 Code Block Loss

Multimedia sessions in 3G mobile networks are initiated, modified and terminated by SIP based message exchanges according to TS 24.228 [2]. The sequences of the SIP message exchanges defined in TS 24.228 should be followed unless there is a special circumstance. Because each message of the sequence represents important processes of the session setup phase (e.g., resource reservation, codec negotiation, authentication etc.), all the messages must be received without any error.

As mentioned in Chapters 2 and 3, the datagrams are fragmented before they enter channel encoders and these segments are called *code blocks*. Thus, if we want to use the probability of data unit loss function given in Equation 3.23 with a code block as a data unit, we need to find the number of code blocks needed to be transmitted to resemble the original control

message[1] based on the size of a code block. Note that the reason for using code block is discussed in Chapter 6.

7.1.1 The Number of Code Blocks for Each Message

Assuming SIP/SDP datagrams are carried over UDP and IPv6 according to the 3GPP specification, and SigComp is used for message compression (see Chapter 4), the number of required code blocks for each message can be calculated as

$$N_{block} = \left\lceil \frac{(N_{c.m.} + N_{UDP} + N_{IPv6} + N_{CRC})}{N_{size}} \right\rceil \tag{7.1}$$

where $N_{c.m.}$ is the size of a control message compressed using dynamic or static SigComp. The values for $N_{c.m.}$ are shown in Table 5.7. N_{UDP} is the size of UDP header($= 8bytes$), N_{IPv6} is the size of IP version 6 header ($= 40bytes$), and N_{CRC} is the size of CRC attachment. The value of N_{size} represents the size of each code block, which can vary up to 504 bits (see Chapter 2). The notations of N_{size} and N_{block} follow the notation used in Chapter 6, and Figures 6.2 and 6.3. How these code blocks are transmitted is also discussed in Chapter 6. Since the integrity of the message is important for proper session initiation, the maximum size of CRC attachment (24 bits) was assumed to be used for better error detection (i.e., $N_{CRC} = 24$). Note that in order to use Equation 7.1, all parameters must be converted into bits.

Table 7.1 shows the required numbers of code blocks, N_{block}, for the different sizes of the code blocks N_{size}. The values of different N_{size} is chosen so that the IP packets, which resemble code block can be easily made. Note that in actual experiments IP packets pretending code blocks are exchanged between a dummy UE and a dummy P-CSCF (see Chapter 6). The numbers in the parentheses of Table 7.1 represents corresponding BHSIP/BHSDP alternatives.

Obviously the number of code blocks needed to be transmitted increases as the size of a code block decreases. Using Equations 3.23, the probability of code block loss can be calculated according to the code block sizes (by letting $B_F = N_{size}$). Upon computing the theoretical probability of code block loss, P_{loss}, due to bit corruption using Equations 3.23,

[1]The term control message is widely used in this chapter. It means the message which initiates a multimedia session. For example SIP and BHSIP are the control messages.

Type	104 bits		304 bits		504 bits	
	Dynamic	Static	Dynamic	Static	Dynamic	Static
INVITE	49 (44)	49 (44)	17 (15)	17 (15)	10 (9)	10 (9)
Trying	8 (7)	17 (17)	3 (3)	6 (6)	2 (2)	4 (4)
Progress	26 (25)	45 (43)	9 (9)	16 (15)	6 (6)	10 (9)
PRACK	11 (10)	40 (37)	4 (4)	14 (13)	3 (2)	9 (8)
OK	8 (8)	30 (29)	3 (3)	11 (10)	2 (2)	7 (6)
UPDATE	9 (9)	44 (40)	3 (3)	15 (14)	2 (2)	9 (9)
OK	10 (7)	29 (28)	4 (3)	10 (10)	2 (2)	6 (6)
Ringing	7 (7)	23 (23)	3 (3)	8 (8)	2 (2)	5 (5)
Total	128 (117)	277 (261)	46 (43)	97 (91)	29 (27)	60 (56)

Table 7.1: The required number of code blocks for SIP/SDP and BHSIP/BHSDP for varying code block size (message named *Progress* represents *Session Progress* message and there are two *OK* messages; one for PRACK and the other for UPDATE)

the probability of successful transmission of each message (SIP/SDP or BHSIP/BHSDP) can be calculated as

$$P_{success} = 1 - (1 - (1 - P_{loss})^{N_{block}})^{N_{SIPrtx}} \tag{7.2}$$

where N_{block} is the number of code block required to assemble original message and N_{SIPrtx} is the number of times that SIP level retransmission is permitted. According to [49], for unreliable transports (such as UDP), the client transaction retransmits INVITE requests at an interval that starts from 0.5 seconds and doubles after every retransmission by default. After receiving a 1XX response (e.g., *"100 Trying"* or *"183 Session Progress"* message), any retransmissions cease altogether, and the client waits for further responses. The retransmission terminates when a 1XX response is not received after 32 seconds from the time when the first INVITE message is generated, and eventually the session initiation attempt fails. From such information, it can be deduced that after 31.5 seconds the 7th INVITE message gets generated. After an additional 0.5 second if no 1XX response is received, the request fails. Therefore, for INVITE messages, N_{SIPrtx} is 7. For response messages such as

such as *"100 Trying", "183 Session Progress", "180 Ringing"* and *"200 OK"* are sent only once when the request is successfully received. Hence, the value for N_{SIPrtx} is 1 for any response messages. Non-INVITE requests, such as PRACK and UPDATE adapts a similar timer mechanism as INVITE, however the maximum value of the timer is 4 seconds. This results in time intervals of 500ms, 1sec, 2sec, 4sec, 4sec, 4sec, etc. The 11th non-INVITE message is transmitted after 31.5 seconds from the first request, hence the value of N_{SIPrtx} for PRACK and UPDATE requests are 11.

As shown in Equation 7.2, the number of code blocks required to transmit for each message contributes heavily to the overall probability of successful message exchanges. That is, increasing the number of code blocks degrades the overall success rate of each message transfer since a single fragment loss of a certain message leads to the total loss of that message (i.e., A large N_{block} reduced the chance of successfully receiving a message). On the other hand, larger value of N_{SIPrtx} leads to a higher $P_{success}$, since many retransmissions can be tried in the case of loss.

7.1.2 Implication of Code Block Size

Figures 7.1, 7.2, 7.3, 7.4, 7.5 and 7.6 show the probability of 504-bit code block loss with two convolutional encoders operating in three speed scenarios defined in Chapter 3, and Figure 7.7, 7.8, 7.9, 7.10, 7.11 and 7.12 show the probability of 304bit-code block loss. We discussed the probability of data unit loss in Equation 3.23. Since in this experiments, we are referring to the code block of size N_{size}, the parameter, B_F in Equation 3.23 is equal to the value of N_{size}. The probability of bit error, P_b, was calculated using the effective E_b/N_0 values according to the interleaving TTI[2], channel coding scheme, and fading signal envelop generated by the method discussed in section 3.2.3.

The value of N_i represents the value of TTI of the interleaving scheme (see Chapter 2). All figures have similar trends. The probability of code block loss stays 100% (i.e., complete loss of code blocks $P_{loss} = 1.0 \times 10^0$) for lower values of effective E_B/N_0. This is obvious because at such a low level of signal power, receiver can not identify the message even with the aid of error correction techniques. However, for all cases, the probability of code

[2]Transmission Time Interval. It is the time span of the inter-frame interleaving scheme of UMTS. It artificially inserts the time gap between consecutive radio frames in order to avoid bust of radio frame corruptions; see Chapter 2

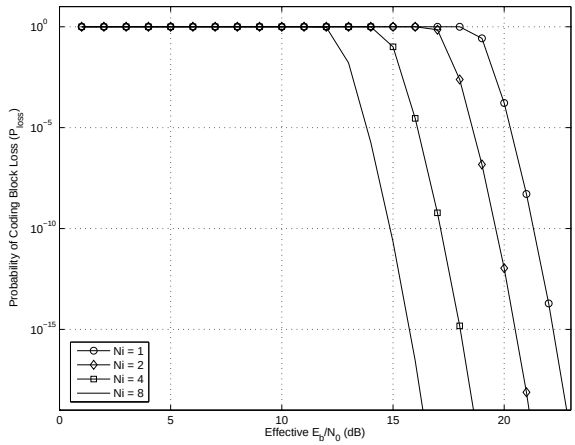

Figure 7.1: Probability of 504-bit code block loss with 1/2 rate encoder with signal source moving at 1km/h

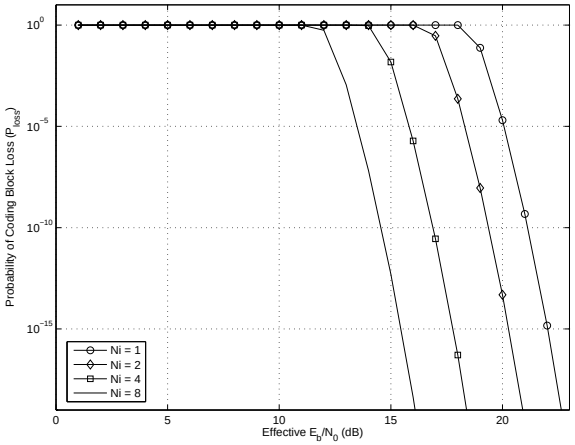

Figure 7.2: Probability of 504-bit code block loss with 1/3 rate encoder with signal source moving at 1km/h

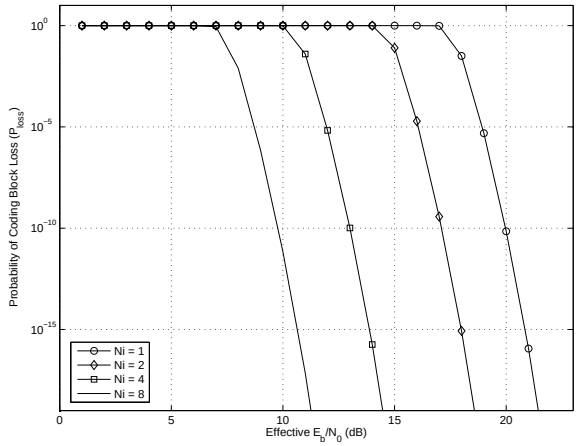

Figure 7.3: Probability of 504-bit code block loss with 1/2 rate encoder with signal source moving at 4km/h

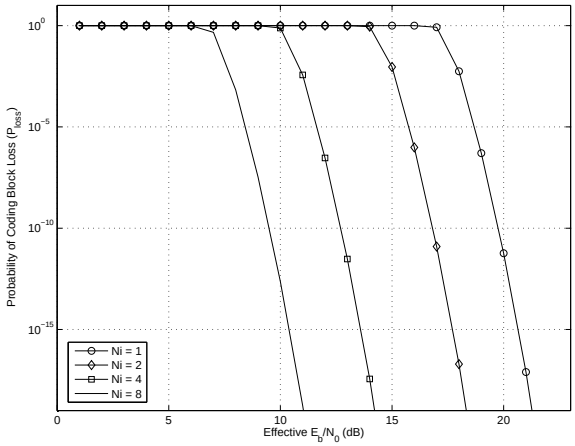

Figure 7.4: Probability of 504-bit code block loss with 1/3 rate encoder with signal source moving at 4km/h

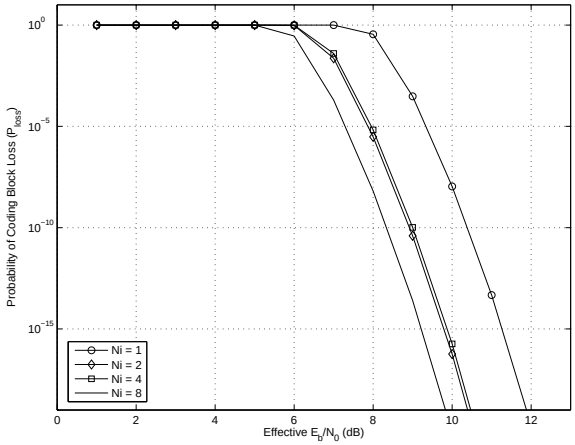

Figure 7.5: Probability of 504-bit code block loss with 1/2 rate encoder with signal source moving at 50km/h

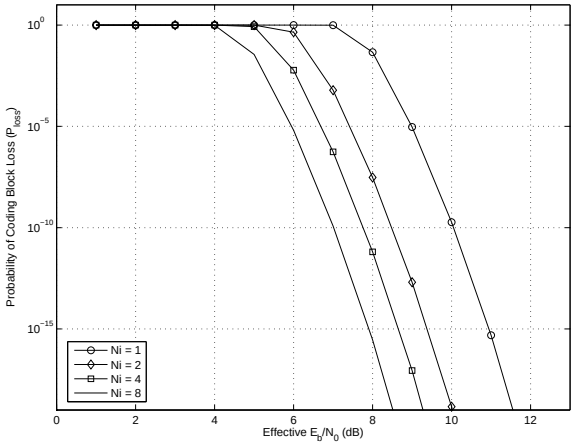

Figure 7.6: Probability of 504-bit code block loss with 1/3 rate encoder with signal source moving at 50km/h

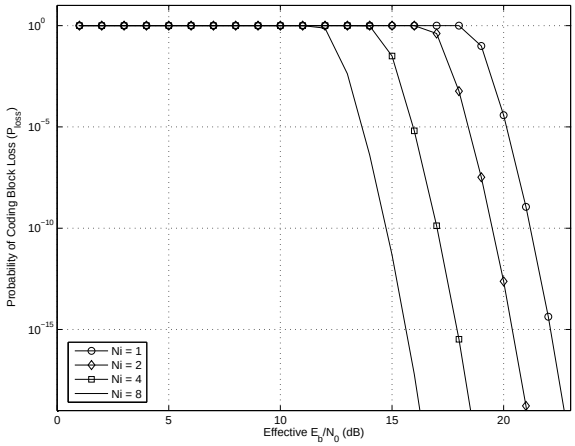

Figure 7.7: Probability of 304-bit code block loss with 1/2 rate encoder with signal source moving at 1km/h

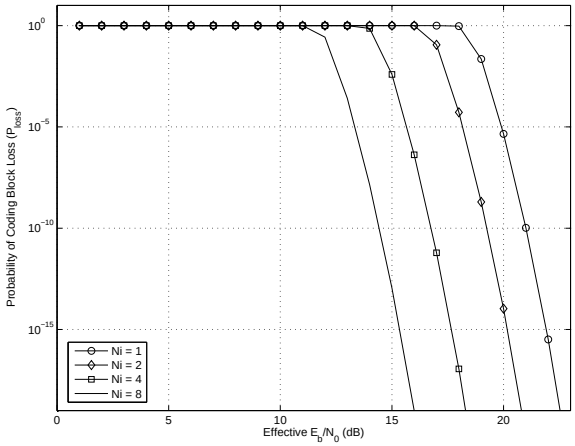

Figure 7.8: Probability of 304-bit code block loss with 1/3 rate encoder with signal source moving at 1km/h

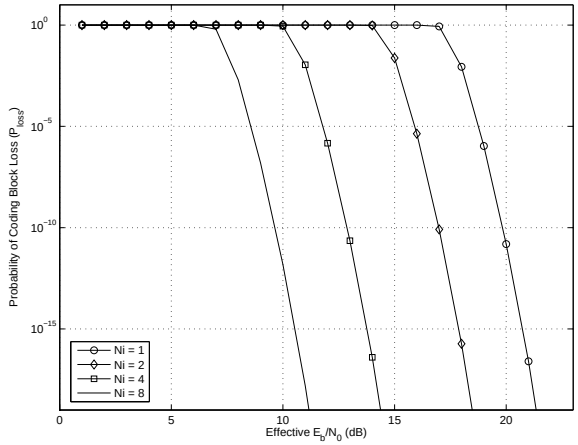

Figure 7.9: Probability of 304-bit code block loss with 1/2 rate encoder with signal source moving at 4km/h

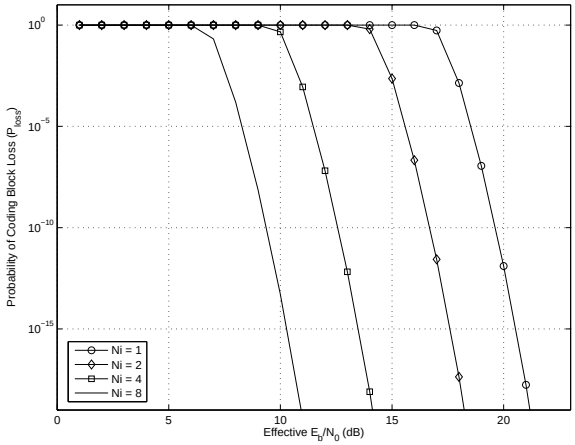

Figure 7.10: Probability of 304-bit code block loss with 1/3 rate encoder with signal source moving at 4km/h

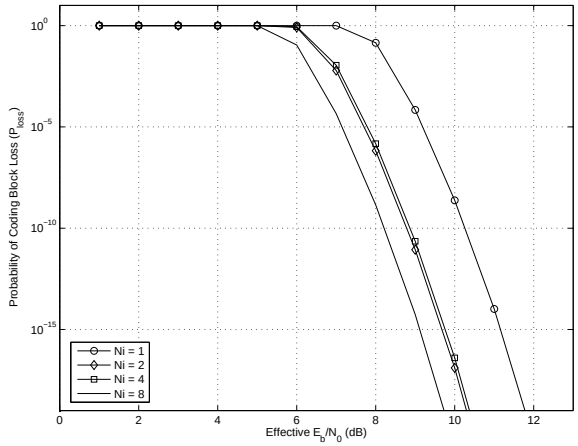

Figure 7.11: Probability of 304-bit code block loss with 1/2 rate encoder with signal source moving at 50km/h

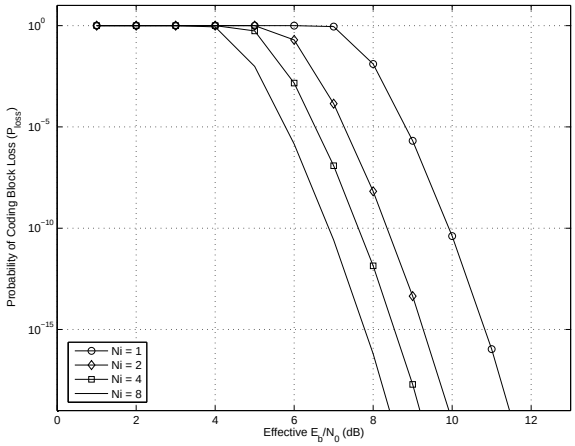

Figure 7.12: Probability of 304-bit code block loss with 1/3 rate encoder with signal source moving at 50km/h

block loss drops drastically from certain threshold value of effective E_B/N_0. From those thresholds values of effective E_B/N_0 and higher, a signal is identifiable to the receiver and the error coding can recover some of the corrupted bits hence lowering P_{loss}. Depending on the speed of the signal source, the size of the code block and the convolutional code rates, the thresholds for the sudden dropping of the probability of code block loss differs.

In general, 1/3 rate code performs better then 1/2 rate code. The falling trends of P_{loss} starts earlier for the figures showing 1/3 rate code cases than the figures showing 1/2 rate code cases. Moreover, a higher TTI has the benefit of saving power while maintaining the code block loss rate. For instance, if we consider Figure 7.1, P_{loss} begins to fall down after 12dB of effective E_B/N_0 if interleaving TTI is 8. However, if TTI is 1, falling trends occurs after 18dB. Another interesting aspect to notice is that at higher speed, less average E_B/N_0 is required than the slower cases. Moreover, the code block drop rate, P_{loss} becomes slightly smaller as the size of code block decreases. Such a difference of P_{loss} is hardly visible in the figures, however if we look at actual values for P_{loss} shown in Appendix D, the difference is rather significant. For example, considering the case when the terminal is moving at 4km/h and N_i is 4 with a 1/2 rate channel encoder, at 11 dB of $(E_b/N_0)_{eff}$, the code block loss rate for 504 bit block is 3.96852×10^{-2} (see Appendix D). If the size of a code block decreases to 304 bits with the exactly same conditions, the code block loss rate becomes 1.10104×10^{-2}, which is less than one-third than that for the 504 bit code block case (see Appendix D). Table 7.2 shows the theoretical message transfer success rate of the each dynamically compressed SIP messages using Equation 7.2, for this example case. Note that the value of N_{block} is shown in Table 7.1, and the value for N_{SIPrtx} for each message is discussed earlier.

As shown in Table 7.2, $P_{success}$ for each message is slightly lower as the size of a code block increases. For smaller code blocks, the number of blocks needed to be transfered in order to assemble the original message increases. However, due to significantly lower loss probability of each code block loss the overall message success rate outperforms the bigger size code blocks. Hence, it is expected that the session initiation would have higher probability of success when the size of the code block is smaller while the channel encoding scheme, modulation technique and the physical channel condition (e.g., fading characteristics, E_b/N_0, etc.) are kept the same. If the size of the code block is smaller, there is a smaller chance that the bits from the same code block occupy the same time-slot, whereas if the size of code block is larger, multiple bits from the same code block can share the same time-slot. Hence, using a smaller size code block has the benefit of more effective prevention

	504 bit code block	**304 bit code block**
P_{loss}	3.96852×10^{-2}	1.10104×10^{-2}
	$P_{success}$	$P_{success}$
INVITE	9.99546×10^{-1}	9.99996×10^{-1}
Trying	9.22205×10^{-1}	9.67331×10^{-1}
Session Progress	7.84299×10^{-1}	9.05160×10^{-1}
PRACK	1.00000×10^{0}	1.00000×10^{-1}
OK (PRACK)	9.22205×10^{-1}	9.67331×10^{-1}
UPDATE	1.00000×10^{0}	1.00000×10^{-1}
OK (UPDATE)	9.22205×10^{-1}	9.56680×10^{-1}
Ringing	9.22205×10^{-1}	9.67331×10^{-1}

Table 7.2: Probabilities of message transmission success for different code block sizes, when the terminal is moving at 4km/h, TTI = 4, and using 1/2 rate code

of bit corruption in a code block. However, using too small size code blocks may lead to overloading of the channel encoders and the decoders.

7.2 Message Exchange Delay

The issue addressed in this section is the time required to complete the control message exchanges sequence during the post-dialing phase shown in Figure 5.2.

7.2.1 Sources of Delay

An obvious source of the delay is the relatively slow data transfer rate of a 3G wireless link, and the problematic size of SIP/SDP messages. Even with the compression scheme, SigComp, the size of the datagrams can be too large for the given link condition to complete the message exchange within a tight delay budget[3] .

The control messages (SIP/SDP) which are carried over a lossy link, such as a wireless link of a cellular network, can suffer serious losses when not enough power is provided to

[3]The network operators may not follow the recommendation of ITU, however, in this project, when we mention the delay budget, we are referring to the recommendation of ITU shown in [34]

the electromagnetic signals. Even with powerful channel coding schemes, some data can not be recovered and then the sender is forced to retransmit the particular information. The retransmission of the message takes an additional message transfer time, hence the extra delay. If the link provides higher data transfer rate (e.g., a few Mbps), than the additional transfer delay would not affect significantly, however it is not generally the case for 3G mobile networks. For instance, resending of the longest message, INVITE, usually requires a substantial amount of time.

Another additional delay comes from the timer mechanism. The sender (or caller) waits for the response for a specific amount of time. If no reception is made until the wait timer expires, the sender retransmits the same SIP/SDP request, and resets the timer with some pre-arranged value. Since the value of timer expiration increases exponentially, the time required for the session initiation varies significantly depending on whether it is successful on the first attempt, or for instance, successful on the seventh attempt even though they both count as successful message exchanges. Hence, any consecutive SIP/SDP message loss may significantly increase the overall delay.

A less significant source of delay is the size of the code block. If the size of the code block is smaller, more code blocks need to be transmitted in order to reassemble the original message. Since it involves processing at both the sending and the receiving end, the delay measured from the time when a given SIP/SDP message is generated until the corresponding request is received slightly affected by the processing delay.

7.2.2 Experimental results

Delays were measured for the two sizes of code blocks, N_{size} (i.e., 304 bit and 504 bit code blocks). One replication of the experiment is basically one life cycle of a dummy UE shown in Figure 6.2. That is, the delay is measured from the time when a dummy UE start until the time when it finishes with success. This precisely measures the time from the first generation of INVITE message to the reception of the *Ringing* message (i.e, delay incurred over UE to P-CSCF link during the post-dialing period). Since the value of delay is only sensible if the session initiation is successful, we do not count replications that finish with failure (see Figure 6.2). The delay was measured by varying the probability of code block loss, P_{loss}, the type of SigComp schemes, and the type of control messages (e.g., SIP or BHSIP).

As mentioned in Chapter 3, the cause of code block loss considered in the project is the

fading due to the speed of mobile terminal. The effective E_b/N_0 were calculated based on the three scenarios defined in Chapter 3, to identify the probability of the code block loss, P_{loss}, and the resultant values for P_{loss} is shown in the previous section 7.1 for various cases.

The value of P_{loss} was varied within the range that measurements can still satisfy the ITU's recommendation of the delay budget (i.e., 2 seconds) For example, beyond 4% of P_{loss} is not obtained for 504-bit code block and 3% for 304-bit code block since with such severe drop rates, session initiation can not be estibished within the delay budget which is recommended by ITU [34] (i.e., measured delays subtitentially exceed 2-second delay budget).

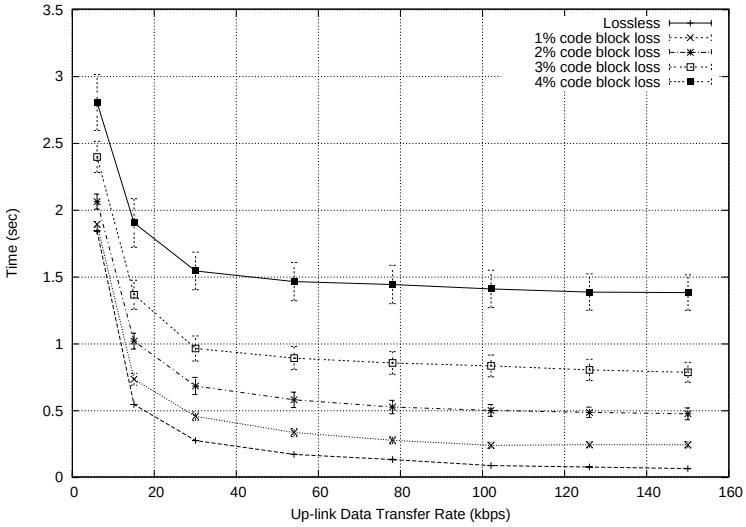

Figure 7.13: Message exchange delay with 504-bit code block when the request is transmitted from a mobile terminal to a Node-B

Figures 7.13 and 7.14 assume that the dynamic SigComp SIP message is fragmented into 504 bit long code blocks, whereas Figure 7.15 assumes 304 bit long code blocks. Note that the the horizontal axes of Figures 7.13 to 7.15 are up-link data transfer rate, and the data transfer rate of the down-link is assumed to be twice of that of up-link (see Chapter 3 for the reason). Figures 7.13 and 7.14 differ, since in Figure 7.13, the requests travel through the up-link and the responses travel through the down-link whereas in Figure 7.14, it is an opposite case. The messages should be transmitted from the initiating end[4] are INVITE, PRACK and UPDATE, and the terminating end[5] should transmit Trying, Session Progress, two OKs and

[4] An initiating end is a network entity which generates SIP requests and transmits the requests

[5] A terminating end is a network entity which generates SIP responses and transmits the responses

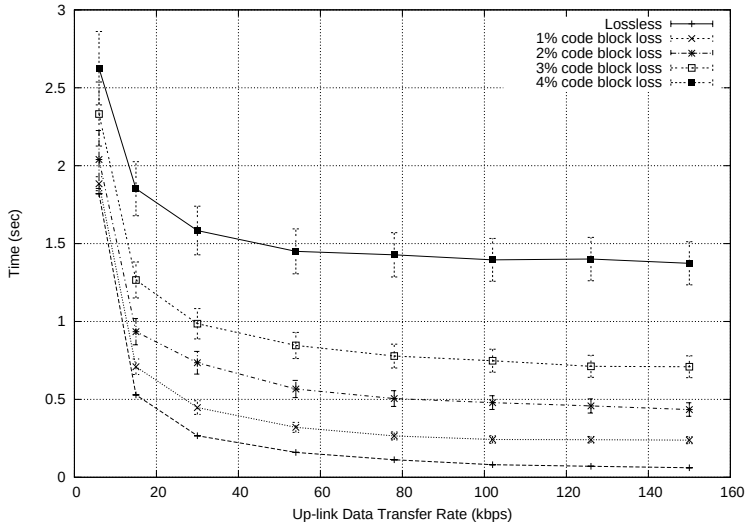

Figure 7.14: Message exchange delay with 504-bit code block when the request is transmitted from a Node-B to a mobile terminal

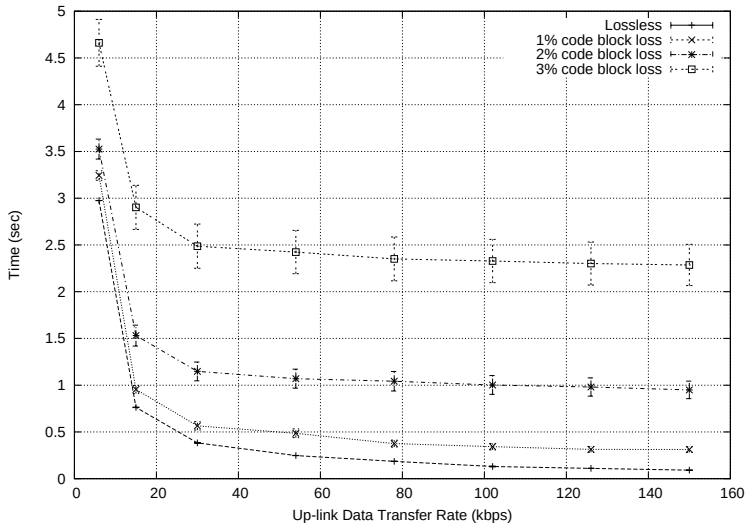

Figure 7.15: Message exchange delay with 304-bit code block when the request is transmitted from a mobile terminal to a Node-B

Ringing messages. If we consider the values in Table 7.1, the total number of code blocks needed to be transmitted by the terminating end is slightly smaller than the total number of code blocks needed to be transmitted by the initiating end in dynamic SigComp cases. However, in static SigComp cases, the terminating end transmits slightly more code blocks. In addition, in dynamic SigComp BHSIP with 504-bit code block case shown in Table 7.1, the terminating end transmits more code blocks. The slight difference in the number of code blocks introduces the deference in the delay. In the case shown in Figure 7.14, the requests are sent over faster down-link, where as the responses are transmitted over slower up-link. Since the number of code blocks needed for request is greater than the number of code blocks for responses, this case shows delay than the case shown in Figure 7.13. In Figure 7.15, smaller code block introduced an extra delay for exchanging the message even in lossless case. That is because it requires extra processing time to check all the received code blocks. For high P_{loss} cases, 304-bit code block case suffers more severely than 504-bit code block case. That is clear predictable by considering Equation 7.2, since as N_{block} is significantly large. However, it was observed that given a code block loss rate P_{loss}, a smaller code block requires more time to successfully transmit the messages. Since smaller size of code block, N_{size} requires the large number of code blocks, N_{block}, there is a higher chance that any of N_{block} code blocks may be lost during the transmission. Such claim is supported by the Equation 7.2. Since the value of N_{block} increases as N_{size} decreases, the overall probability of successful transmission of a message (SIP/SDP or BHSIP/BHSDP) decreases at a given value of P_{loss} However, given the channel condition (i.e., the same effective E_b/N_0 due to the fading, same channel encoders, etc.), a smaller code block results in a smaller value of P_{loss}. If we consider the example given previously in section 7.1.2, a 504 bit code block has approximately 4% of P_{loss} and a 304 bit code block has approximately 1% of P_{loss}, and it was found that the message exchanges based on a 304 bit code block outperform in terms of delay measured (see Figures 7.6 and 7.12 for instance). Since smaller code blocks have a lower probability of code block loss, it is less likely to require retransmission of messages than the larger code blocks, hence the overall time needed might be smaller. However, usage of smaller code blocks introduced more workload to the channel coding mechanisms. Obviously the size of each code block and the number of code blocks required to reassemble the original message influence the overall performance of the session initiation scheme.

For illustration purposes, the half widths of the confidence intervals of the obtained mean values at a confidence level of 95% are included with error bar styled vertical lines. In order

to generate reliable data of average delay, the replications mentioned previously, were at least repeated 500 times. If the resultant average delay, after 500 replications, has the relative measure of the statistical error less than 10%, the value is accepted, otherwise the replication continues until the error bcomes smaller than 10%.

Equations 7.3 and 7.4 show the two fundamental equations for estimating the error which the experimental results have.

$$P\{\overline{X}(N) - \Delta(N) \leq EX \leq \overline{X}(N) + \Delta(N)\} = 1 - \alpha \tag{7.3}$$

$$\delta(N) = \Delta(N)/\overline{X}(N) \tag{7.4}$$

The value of EX is the actual or theoretical mean value and $\overline{X}(N)$ is the estimate of the mean obtained from the experiments by observing N samples (or replications in this experiment). The value of $\Delta(N)$ is the half confidence interval of a given confidence level of $1 - \alpha$.

As mentioned, the average values shown throughout the thesis are based on the mean value of confidence level of 95%. That is, the mean value lies between the value of difference between $\overline{X}(N) - \Delta(N)$ and $\overline{X}(N) + \Delta(N)$ with 95% probability (see Equation 7.3). Equation 7.4 gives the relative measure of the statistical error of the obtained mean value.

As the value of P_{loss} increases, each replication of the experiment results in rather wide range of delay measurements. In some replications message may suffer more severely than others, hence the higher delay measurements. In other cases, no code block loss can be observed. Due to this vast difference of delay measurements, the variance of the measurements also increases, hence larger number of replication was required to get more reliable results. That is precisely why we kept repeating replication until desirable target error level of under 10%, is acheived.

As an alternative to the standard dynamic SigComp, the static SigComp was proposed in Chapter 4 for reducing the workload of SigComp layer, thus reducing the complexity of the layer. However, the static SigComp produces smaller compression ratio than the dynamic variant, hence, it requires more code blocks to transmit than its dynamic variant. The delay measurements shown in Table 7.3, confirms that influence of the disadvantage of having smaller compression ratio. If we consider the size comparison given in Table 5.7, the total size of messages using dynamic SigComp is about one-third of the total size

of messages using static SigComp. Because of such substantial size difference the delays differ significantly. Although the difference ratio in the size is about 3 times, the delay measurements do not exactly follows this ratio. That is, appending extra overheads (e.g.,. IPv6 and UDP headers, and CRCs, etc.) reduced the size difference slightly. Moreover, at higher data transfer rate, the implication of size becomes less significant. For example, at 6 kbps, the the delay difference ratio between the dynamic SigComp and the static SigComp is slightly more than 3 times. However, at 54kbps, the difference ratio becomes approximately 3 times, and at 150 it becomes less than 3 times (see Table 7.3). The Table 7.3 shows the delay measurements of up to 3%. Since we are considering the ITU's recommendation of session initiation delay (i.e., 2 seconds), delays which are excessively larger than the recommendation are of no use.

The same experiments using static and dynamic SigComp on BHSIP were also conducted. Table 7.3 show the results. Obviously the smaller size of BHSIP revealed the benefit of reducing the delay even further than the SigComp with SIP. The delay measured by dynamic SigComp BHSIP outperforms all other alternative combinations of SigComp schemes and control messages.

Overall, in this section, we found that the several factors influence the delay of message exchanges. The major factor was the number of code blocks needed to be exchanges. In other words, the size of control message is the major source of the delay. We found that although static SigComp can simplify the SigComp layer structure, it introduces larger delays, and the smaller size of control message using BHSIP is found to be effective to save the time to exchange messages. The difference in the data transfer rate in up-link and down-link also introduced the deference in the delay measurements.

7.3 Multimedia Session Initiation Failure

As mentioned earlier any one of requests transmitted during the session initiation phase represents a vital process of the multimedia session initiation steps. Hence. any unsuccessful message exchange due to the loss of message, results in the failure of multimedia session initiation. Such a failure case can be viewed as the state diagram in Figure 6.2 being terminated with failure. Note that one message loss does not cause a failure of whole initiation process.

Up-link Speed (Type)	**Probability of code block loss** (P_{loss})			
	0%	**1%**	**2%**	**3%**
6 kbps Dynamic SigComp SIP	2.9752	3.2430	3.5253	4.6617
6 kbps Dynamic SigComp BHSIP	2.7005	2.9616	3.4560	4.1244
6 kbps Static SigComp SIP	8.4613	9.3280	11.3109	13.7091
6 kbps Static SigComp BHSIP	6.9244	8.6924	10.5090	13.3933
54 kbps Dynamic SigComp SIP	0.2473	0.4854	1.0696	2.4239
54 kbps Dynamic SigComp BHSIP	0.2280	0.4347	1.1122	2.2917
54 kbps Static SigComp SIP	0.4890	1.1820	3.3521	7.0676
54 kbps Static SigComp BHSIP	0.4550	1.1713	3.1925	6.7495
150 kbps Dynamic SigComp SIP	0.0905	0.3117	0.9493	2.2857
150 kbps Dynamic SigComp BHSIP	0.0852	0.2933	0.9233	2.1527
150 kbps Static SigComp SIP	0.1773	0.8688	3.0366	6.7987
150 kbps Static SigComp BHSIP	0.1652	0.7635	2.9031	5.7566

Table 7.3: Delay (in sec) incurred over 3G wireless channel (values shown up to 4 decimal places)

Rather, series of message loss up to N_{SIPrtx}[6] losses causes the failure. In addition, the loss of a message is caused by at least one loss of code block which represents a segment for that message. This part of experiment is basically the duplicated experiment for measuring the average delays, however, we measure the average number of multimedia session initiation failure instead.

7.3.1 Experimental Results

The experiments were conducted by varying the value of probability of code block loss, P_{loss} as before. The measurements are obtained only for the cases, where the corresponding delay measurement meets the recommendation of ITU. For example, the delay measurements for over 3% of P_{loss} for 304-bit code blcok cases do not satisfy the recommendation, hence, we did not observe the measurements of failure up to 3% of P_{loss}. For 504-bit code block cases, measurements up to 4% of P_{loss} were obtained.

The replication of experiment defined in the previous section (e.g., one life cycle of Figure 6.2) was repeated 100 times, and the number of failures were recorded. Such process is repeated 15 times (i.e., totally 1500 replications) with different seeds for pseudo random number generator used to drop code blocks traversing the virtual link. The values shown in this section are the average number of failure out of 100 replications. Mathematically, the average number of failure out of 100 replication can be shown as

$$F_{avg} = \frac{F_1 + F2 + ... + F_{14} + F_{15}}{15} \tag{7.5}$$

where F_n (n = 1, 2... 14, 15) is the observed number of failure within nth set of 100 replications.

Table 7.4, shows average session initiation failure over 100 replications, F_{avg}, assuming that the messages were SIP/SDP messages compressed by dynamic SigComp and the code block size is 504 bits, and Table 7.5 shows the values of F_{avg} for 304-bit code blocks. Numbers without parentheses represent the cases where the INVITE messages are transmitted from the mobile terminal to the Node-B, and the numbers within the parentheses represent the opposite case.

In the case of lower probability of code block loss, we observed no failures out of total 1500 replications. The results do not mean that they can not be any failure over this value of

[6]The number retransmission allowed for a particular message see Equation 7.2

Up-link Speed	**Probability of code block loss** (P_{loss})			
	1%	**2%**	**3%**	**4%**
6 kbps	0.00 (0.00)	0.00 (0.00)	0.20 (0.13)	1.00 (1.07)
15 kbps	0.00 (0.00)	0.00 (0.00)	0.13 (0.13)	1.07 (0.87)
30 kbps	0.00 (0.00)	0.00 (0.00)	0.07 (0.07)	0.60 (0.60)
54 kbps	0.00 (0.00)	0.00 (0.00)	0.07 (0.00)	0.47 (0.47)
102 kbps	0.00 (0.00)	0.00 (0.00)	0.00 (0.07)	0.60 (0.33)
150 kbps	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.33 (0.33)

Table 7.4: Average number of session initiation failure, with 504 bit code block using dynamic SigComp SIP (values are shown up to 2 decimal places)

Up-link Speed	**Probability of code block loss (P_{loss})**		
	1%	**2%**	**3%**
6 kbps	0.00 (0.00)	0.07 (0.07)	2.20 (2.33)
15 kbps	0.00 (0.00)	0.07 (0.07)	2.07 (2.20)
30 kbps	0.00 (0.00)	0.13 (0.07)	1.47 (1.33)
54 kbps	0.00 (0.00)	0.07 (0.13)	1.33 (1.20)
102 kbps	0.00 (0.00)	0.00 (0.07)	1.13 (1.13)
150 kbps	0.00 (0.00)	0.07 (0.00)	1.33 (1.13)

Table 7.5: Average number of session attempt failure out of 100 trials, with 304 bit code block using dynamic SigComp SIP (values are shown up to 2 decimal places)

P_{loss}. It merely means, rather, the case of failure was not observed, and may happen if more number of replication is performed.

.

One of the important observations from the results is that given the same value of P_{loss} a smaller code block size suffers more severely than the bigger code block based message exchange. That is because the number of code blocks required to be received increases as the size of the code block decreases, hence there is smaller chance of getting every code block without errors (see Equation 7.2).

One common characteristic of results shown in Tables 7.4 and 7.5 is that the number of failures increases as the data transfer rate decreases. Equation 7.2 is not able to explain such trends. It is due to the timer mechanism of SIP. At low data transfer rates, the message requires more time to complete transfer. For instance, a compressed SIP message of approximately 500 bytes long attached with UDP and IPv6 headers requires 0.73 second to be transfered over a 6 kbps link. It is well beyond the the waiting time for the response for the

last resending of the SIP request[7]. In other words, the SIP transaction ceases and the session fails while the final request is still transfered to the destination. If we consider 15 kbps link, we can find the 500 bytes long attached with UDP and IPv6 headers requires approximately 0.29 second to be transfered to the destination, which is more than the half of the waiting time for the last resending of the SIP request. In this case, the transaction might cease while the response is being transfered to the source of the request. That is why we observed substantially large number of failure in the cases of 6 kbps and 15 kbps than other cases. The SIP specification [49] mentions that the first timer value and the transaction expire timer can be reconfigured, although it is not recommended, to the value of Round Trip Time (RTT) of the largest message. Hence if only 6 kbps or 15 kbps link is available than the timers should take bigger value. Since the size of a request is usually bigger than the response to the request, the RTT of a request over a given link is expected to be enough to avoid unnecessary retransmission of the request. However, a longer wait time of the retransmission timer does not guarantee a successful session initiation and it may degrade the Quality of Service (QoS) of the overall session initiation scheme since every retransmission introduces substantial amount of time. For relatively high speed links, such as 102 kbps and 150 kbps links, the numbers of failures are similar, since they do not have the problems caused by timer mechanism.

Overall, at lower data transfer rates (e.g., 6 kbps and 15 kbps), due to the timer mechanism we observed more failures, and the number of failures were quite similar. For higher data transfer rates (i.e., 30 kbps and above) the timer mechanism does not affect the number of failures. Hence we observed similar number of failures, which is much smaller than the lower data transfer rate cases. From this set of results, we can notice that the number of failure, as long as the timer is set up properly, does not depend on the data transfer rate, rather it only depends on the probability of code block loss, P_{loss}. Since P_{loss} depends on the number of code blocks, and the number of code blocks depends on the size of code blocks and the size of the control messages, we can say the session initiation failure[8] is dependent on the size of the control messages and the size of code blocks.

With a fixed P_{loss}, smaller size code blocks underperform, however, a smaller code block results in a smaller P_{loss} for given channel conditions (see section 7.1.2). If we go back to

[7] After the generation of the last request, SIP application waits for 0.5 second, and if no response, the request transaction ends

[8] Session initiation failure is caused by the failure in the control message exchange.

Types	**Probability of code block loss** (P_{loss})		
Data transfer speed	**1%**	**2%**	**3%**
Dynamic SigComp BHSIP	0.00	0.07	1.87
6 kbps	(0.00)	(0.07)	(2.20)
Static SigComp BHSIP	0.07	3.53	11.00
6 kbps	(0.00)	(0.07)	(2.20)
Static SigComp SIP	0.20	3.33	11.33
6 kbps	(0.00)	(0.07)	(2.20)
Dynamic SigComp BHSIP	0.00	0.00	0.87
54 kbps	(0.00)	(0.07)	(1.33)
Static SigComp BHSIP	0.00	0.47	3.53
54 kbps	(0.00)	(0.07)	(1.33)
Static SigComp SIP	0.07	0.53	4.60
54 kbps	(0.00)	(0.07)	(1.33)
Dynamic SigComp BHSIP	0.00	0.00	0.87
150 kbps	(0.00)	(0.07)	(1.33)
Static SigComp BHSIP	0.00	0.47	2.67
150 kbps	(0.00)	(0.07)	(1.33)
Static SigComp SIP	0.00	0.47	3.40
150 kbps	(0.00)	(0.07)	(1.33)

Table 7.6: Average number of session attempt failure out of 100 trials, with 304 bit code block, using different type of SigComp and control messages

the example given in section 7.1.2, a 504 bit-code block has P_{loss} of 3.96852×10^{-2} whereas for a 304 bit-code block this probability equals to 1.10104×10^{-2}. If we compare the value of P_{loss} with the Tables 7.4 and 7.5, we can reach the conclusion that the 304 bit-code block outperforms overall in this particular case. It does not necessarily mean that a smaller code block outperforms at all times. As discussed in the section 7.2 use of smaller code block introduces more workload to channel coding mechanism.

To show that the number of code blocks affects the number of session initiation failures, similar experiments were conducted again, however, assuming 1) static SigComp SIP, 2)

static SigComp BHSIP and 3) dynamic SigComp BHSIP. Since static SigComp produces messages of bigger sizes than dynamic SigComp, hence there are more code blocks, it is expected that the number of failures would increase at the given P_{loss}. Similarly, dynamic SigComp BHSIP would be more immune to the higher P_{loss} than other variations. The average number of failures observed are shown in Table 7.6. The experiments were conducted in the same way as the previous failure measuring experiment (i.e., 15 sets of 100 replications). The numbers within parentheses represent the dynamic SigComp SIP case with 304 bit code blocks shown in Table 7.5. As expected the number of failures increases for static SigComp SIP and it decreases for dynamic SigComp BHSIP. The performance of static SigComp BHSIP is better than that of static SigComp SIP, however it is outperformed by dynamic SigComp SIP. The results imply that another advantage of the binary hybrid scheme is reduction of the average number of session initiation failure.

The failure measurements shown in this section has higher value of the relative measure of error (see Equation 7.4). That is because, some sets of 100 replications do not have any failure, but some sets have many failures due to different seed for the pseudo random number generator used for code block dropping. More reliable value of average can be obtained if experiment is carried for longer time (i.e., more sets of replications). However, what we were trying to show here is to show that the retransmission of control message can not prevent the session initiation failure if the channel condition is relatively bad, and the timer mechanism can cause additional failure is it is not properly configured. Even with the mean value of the failure with a high relative error, those aspects are clearly noticeable. Hence, no more experiments were done to reduce the relative measure of error in the measurement of the number of failures.

7.4 Conclusions

This chapter addresses the session failure issue and the possible delay experienced in the wireless link of UMTS mobile networks for various code block sizes, data transfer speeds and different mobile terminal moving speeds. It was observed that the size of code block influences the overall code block drop rate, P_{loss}. Smaller code block sizes lead to smaller values of P_{loss}, since fewer time-slots are needed to complete the transfer of each code block, hence reducing the number of time-slots which are degraded by the fading characteristics of the wireless channel. However, smaller sizes of code blocks require a larger number of code

blocks to be transmitted in order to reconstruct the original message. The implication of this is that it would increase the delay during the post-dialing period due to the processing of the code blocks.

The implication of code block sizes was investigated for the dynamic SigComp SIP/SDP based message exchanges. The results from the case based on 304-bit code block were compared with static SigComp SIP/SDP static SigComp BHSIP/BHSDP and dynamic BHSIP/BHSDP variants. The observation confirmed that the smaller size of datagrams using binary hybrid based protocol leads to the message exchanging more resistant to a severe channel conditions than the SIP/SDP based message exchange.

7.4.1 Application of the Findings

The experimental results presented in this chapter alone can not be used to estimate the overall post-dialing delay of a multimedia session. The results shown in this chapter are based on the measurements of delays incurred in the UE to P-CSCF link, disregarding the processing delay of relaying entities[9] mentioned in Chapter 2. The overall post-dialing delay, Δ_{total} can be estimated by

$$\Delta_{total} = \Delta_{init\_ch} + \Delta_{core} + \Delta_{termi\_ch} + \Delta_{resource} \tag{7.6}$$

where Δ_{init_ch} and Δ_{end_ch} represent the message transfer delays measured over the initiating UE to P-CSCF link and over the terminating UE to P-CSCF link respectively. The value of Δ_{core} is the delay caused by exchanging the messages within the core network entities. The overall delays caused by the resource reservations, or the processing delay of network entities, should be summed into $\Delta_{resource}$.

Let us now consider three typical cases, where the 3G based multimedia session can be established

1. when a mobile UE initiates session with a mobile UE,

2. when a mobile UE initiates session with a fixed UE, and

3. when a fixed UE initiates session with a mobile UE.

[9]the processing and the transmission delay of network entities within wired portion of UMTS network is assumed to be negligible in this project

	Initiating End	Terminating End
Message type	Dynamic SigComp SIP	Dynamic SigComp SIP
Code block size	304 bits	504 bits
Rate of Encoder	1/3	1/3
Data transfer rate	54 kbps	78 kbps
Interleaving TTI	4	8
Moving speed	4 km/h	50 km/h
$(E_b/N_0)_{avg}$	20 dB	9 dB

Table 7.7: Description of two mobile terminals initiating multimedia session

	Initiating End	Terminating End
$(E_b/N_0)_{eff}$	11.16 dB	5.86 dB
P_{loss}	8.88×10^{-4}	6.43×10^{-6}
Message exchange delay	$\approx 247ms$	$\approx 112ms$
Delay due to TTI	$80ms$	$160ms$
Total delay	$327ms$	$272ms$

Table 7.8: Delay estimate for two mobile terminals initiating multimedia session

The term *mobile UE* represents a terminal attached to a UMTS network with the UTRAN wireless interface, whereas the term *fixed UE* represents a terminal connected with a wired network, such as Internet and digital telephone line. The communication between fixed UEs does not involve any mobile terminals. Therefore such case is out of scope of this project. The second and the third cases differ. In the second case, the INVITE message traverses through the up-link channel, whereas in the third case, the INVITE message is transmitted through the down-link channel. The difference in delay due to the path difference is addressed in the previous section.

The results presented in this chapter are the estimates for the values of Δ_{init_ch} and Δ_{end_ch}. If the specification of code block size, data transfer rate, channel coding, interleaving and average E_b/N_0, and the speed of the mobile terminal are known then the values included in Appendices C, D and E can be used to estimate the values of Δ_{init_ch} and Δ_{end_ch}.

Let us give a simple example of the the case 1 of multimedia session (i.e., a mobile UE to

a mobile UE) using the values shown in Appendices. Table 7.7 shows the description of the two mobile terminals trying to establish a multimedia session. Both terminals are assumed to be connected to UMTS networks, which are not necessarily the same network.

Using the information about interleaving TTI, code rate and average E_b/N_0[10], we can find the estimate of effective E_b/N_0 using Appendix C. With code rate, code block size, interleaving TTI and the value of effective E_b/N_0, we can find the estimate of the probability of code block loss from Appendix D. Finally from Appendix E we can find the value of average delay. The relevant data found from the appendices are shown in Table 7.8. Note that the interleaving TTI also introduces extra delay (see Chapter 3). Hence, in order to estimate the values of Δ_{init_ch} and Δ_{end_ch}, the delay values obtained from Appendix E should be added with the extra delay caused by interleaving TTI. The sum of Δ_{init_ch} and Δ_{end_ch} is approximately $599ms$. If the overall budget of the post-dialing delay follows what is recommended by ITU [34] (i.e., 2000 ms), the total value of Δ_{core} and $\Delta_{resource}$ should not exceed $1401ms$. Moreover, from the results given in Table 7.4 and 7.5, the failure of the message exchanges are highly unlikely in both wireless links since the values for P_{loss} for both cases are well below 1 %.

[10]It represents the signal power that mobile terminal transmits

Chapter 8

Final Conclusions

In order to obtain reliable experimental data, the UMTS specific wireless channel model was surveyed and the effective E_b/N_0 based model was selected to implement the link between a User Equipment (UE) and a Proxy-CSCF. The major advantage of the concept is that it allows to use the bit error rate estimation formulas in a fading channel. Since the fading due to the speed of the mobile terminal and the characteristics of the physical layer are considered to be the major source of bit error, the concept of effective E_b/N_0 is well suited for the UE to P-CSCF link model. The channel model used in the project accommodates all the necessary characteristics of the lower layers of the UMTS wireless channel.

Since the size of the datagrams of SIP/SDP is not expected to be small enough for a narrow-band link such as the UMTS air link, especially before a multimedia session is initiated, SigComp introduced by the IETF, is expected to be used. The size reduction by SigComp was significant enough and such a size reduction resulted in the reduction of the message transfer delay (see [64]). Even though the original SigComp is based on dynamic updates on the initial dictionary for the compression and decompression, the performance of the static (no updating on the initial dictionary) was also investigated. The static SigComp is expected to avoid the message exchange failure due to the compression state mismatch while increasing the message exchange delay. The preliminary results of such effects were published in the proceedings of International Conference on Information Networking (ICOIN) 2004 [64].

Since the size of the application layer datagram highly contributes to the message transfer delay, an alternative control protocol to the original SIP was proposed. A binary hybrid protocol design is a novel paradigm for designing, especially, application layer protocols.

In most cases, the application layer protocols are designed entirely with plain texts. This may cause unnecessary delays due to the large size of the datagram. In order to optimize the problematic size of the application layer protocols, the novel paradigm of binary hybrid protocol was introduced. The two example protocols, BHSIP and BHSDP, were presented to compare the delay and the session attempt failure performances with the SIP and SDP based multimedia session setup scheme. The basic notion of binary hybrid paradigm is to redefine all the reserved key words used in the datagram with the binary sequences of a fixed size. The usage of a binary hybrid protocols is also beneficial when they are used with the SigComp structure. Since every key word is defined in the binary sequences, the initial dictionary does not required to have the reserved key words. Moreover, due to the size advantage, the updating of the dictionary can be done more easily.

In order to investigate the performance of the multimedia session setup scheme, an effective E_b/N_0 model was implemented using the network link emulator, NISTnet. The selection of the network emulator was based on the survey of the relevant available emulators. The experiments were designed to resemble the possible cases, which user might encounter during the initiation phase of the multimedia sessions. The scenarios were based on the speed of the terminal assuming no hand-off during the session initiation or seamless enough to have no effects on the delay and the packet drop. The scenarios were further refined by varying the lower level functions, such as convolutional encoders, interleaving TTI and code block sizes. The difference caused by the case when a mobile user initiates session (i.e., a mobile user sends a INVITE message to a Node-B) and the case when a mobile user receives an incoming session invitation (i.e., Node-B send a INVITE message to a mobile user) was also investigated in Chapter 7. The message used in the project followed the example messages given in the 3GPP specification. hence other sets of messages would result in different results.

The experimental results show that the lower layer variables and the surroundings significantly affect the performance of the session initiation scheme based on SIP/SDP. According to the results given in Chapter 7, at lower speed, more power (higher E_b/N_0) is usually required to ensure the successful message exchanges. The rate of convolutional encoder and the interleaving TTI also contributed the overall performance heavily.

Overall, a smaller message size resulted in the fewer message exchange failures and introduced the smaller amount of delays. The binary hybrid protocol examples given in the thesis were proven to be more effective than the original SIP/SDP in terms of the session

attempt failure rate and the average delay. However, binary hybrid protocols lack readability and restrict the flexibility.

The major implication of the project is that it presented the implication of the lower level mechanism, such as channel coding, interleaving and code block segmentation, to the application level multimedia control message exchanges. The results presented in this thesis are based on the emulation of the virtual link and the theoretically sound assumptions. However, the real message exchange over the UMTS wireless channel may encounter different obstacles and the further degradation of the performance can be observed. The results obtained may differ from the real experiments significantly depending on the circumstances such as weather or obstacles of significant size. Hence the result should only be used as a guide-line or a references for the network operators deploying the IP and SIP based services over the UMTS system.

8.1 Future Works

The project only focused on the link between UE and P-CSCF. In order to fully estimate the post-dialing delay, the study of the wire-line portion of the 3G network should be conducted and combined with the results presented in this thesis.

The binary hybrid protocols, according to the experimental results, were proven to be effective, due to its size advantages. However, the example protocols given in this thesis, BHSIP and BHSDP, may have other disadvantages and advantages. The real implementation of the binary hybrid protocol stack would help identify more characteristics of BHSIP and BHSDP.

The security issue of the SIP/SDP message exchange was not of interest of this project, however, in order to fully investigate the reliability of the SIP based multimedia session control scheme over UMTS networks, this issue should be throughly investigated.

The work by *Fathi*, *Chakraborty* and *Prasad* attempted to compare the performance of SIP based multimedia session setup scheme for varying underlying protocols (e.g., TCP and UDP) [19]. As mentioned in Chapter 7, at the low data transfer rate, the timer mechanism of SIP is one of the major sources of the delay, and the work shown in [19] also mentions the inadequacy of the timer over slow links. Hence, more research should be performed to design the retransmission timer according to the channel conditions (e.g, fading characteristics and available data transfer rate).

Appendix A

Introduction of Convolutional Coding

This appendix is included for the readers who are not familiar with convolutional coding techniques. The convolutional encoders discussed in the main part of the thesis have much more complicated structure than what is shown in this appendix. However, by throughly understanding the basic of the convolutional coding introduced in this appendix, the reader will be able to understand the contents of Chapters 2 and 3. The information and the examples given in this appendix are mainly obtained and referred to [28].

A.1 Encoding Basics

There are applications where the message bits come in serially rather than in large blocks of bits, in which case the use of a buffer may be undesirable. In this case we use a coding scheme called *convolutional coding*, which basically generates one coded symbol for one incoming bit. In other words, a convolutional encoder operates on the incoming message sequence continuously in a serial manner.

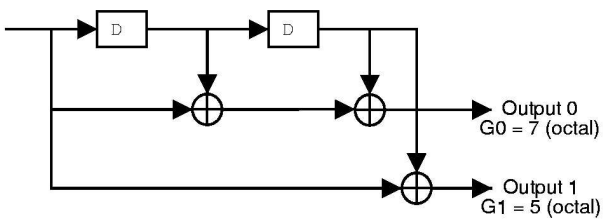

Figure A.1: Convolutional encoder example

A.1.1 Code Rate

The encoder of a binary convolutional code with rate $1/n$, measured in bits per symbol, may be viewed as a finite-state machine that consists of a M-stage shift register with prescribed connection to modulo-2 adders, and generating n number of output symbols.. An L-bit message sequence produces a coded output sequence of length $n(L + M)$ bits. The **code rate**[1]is therefore given by

$$R_c = \frac{L}{n(L+M)} \tag{A.1}$$

Typically, we have $L \gg M$. Hence the code rate simplifies to

$$R_c = \frac{1}{n} \tag{A.2}$$

Hence, if $R_c = 1/3$, encoder 3 symbols for one incoming bit.

For the example shown in Figure A.1, the value for M is 2 (two registers[2]) and the value for n is 2 (number of output symbol). Hence the coding rate of the encoder shown in Figure A.1 is $1/2$ if incoming bit stream is longer than the value of M, which is obviously the case.

A.1.2 Constraint Length

The **constraint length** of a convolutional code, expressed in terms of message bits, is defined as the number of shifts over which a single message bit can influence the encoder output. In an encoder with an M-stage shift register, the memory of the encoder equals M message bits, and $K = M + 1$ shifts are required for a message bit to enter the shift register and finally come out. Hence, the constraint length of the encoder is K. The constraint length of the encoder shown in Figure A.1 is 3, $K = M + 1$ where $M = 2$.

A.1.3 Generator Polynomial

The octal numbers shown in Figure A.1, represents, so called, **generator polynomial** of the encoder. The octal number 7,G_0, is equivalent to the binary number of 111. This means, that in order to generate the first output symbol, the encoder must perform modular-2 addition on the incoming bit (since the first digit of binary representation of G_0 is one), the bit stored in the first register (second digit of binary number is one), and the bit stored in the second

[1]It is also called “coding rate” or “encoding rate”

[2]sometimes referred to as memories or delay units, in Figure A.1 it is represented in a box with a letter ’D’

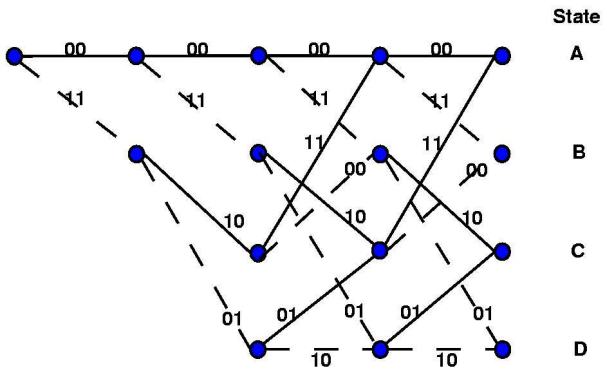

Figure A.2: Trellis for the convolutional encoder shown in Figure A.1

register (second digit of binary number is one). In the case of G_1, which is 5 in octal number and 101 in binary. Hence, the second output symbol, is a modular-2 addition of the incoming bit and the bit stored in the second register.

A.1.4 Trellis View

Figure A.2 shows the trellis representation of the convolutional encoder shown in Figure A.1. The encoding starts from the state A of the left hand side. This state is where two registers have initial values of 0; we denote this state as *A* or state 00 representing the contents of the registers. If the incoming bit is 0, the encoder follows the solid line one step to right hand side generating the output symbols of 00, and it stays at the state 00. The numbers shown beside the lines are the output symbol combination. If the incoming bit is 1, the encoder follows the dashed line, and the state changes into the state *B* or state 10 (the first register stores the value of 1 and the second register stores the value of 0). The process continues until the bit stream finishes.

The trellis representation can also be shown as a state diagram shown in Figure A.3. The label of the state, the notation for lines and output symbols are the same as ones used in Figure A.2.

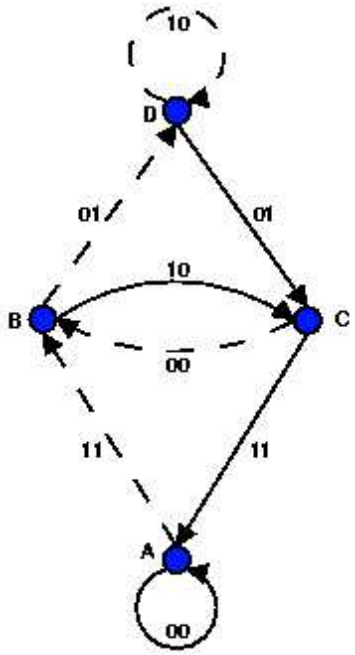

Figure A.3: State diagram of convolutional encoder shown in Figure A.1

A.2 Decoding Basics

One algorithm that is employed in the UMTS channel decoding is Viterbi soft decoding algorithm. The algorithm basically reconstructs the encoding trellis from the received symbols and finds out the most probable path the bit stream has taken for encoding. According to the prediction of the path, it generates the original bit stream. Two examples are given here. Firstly, suppose that the encoder of Figure A.1 generates an all-zero sequence that is sent over a noisy channel, and that the received sequence is 0100010000.... There are two errors in the received sequence due to the noise: one in the second bit and the other in the sixth bit. Let us consider if these errors are correctable.

Figure A.4 shows the step-by-step of the decoding sequences for Viterbi soft decoding algorithm. As for encoding trellis, the decoding trellis starts from the initial state. Since two encoded symbols make up one original bit (Note that the code rate is 1/2) we should consider two symbols as one set. If you look at the encoding trellis shown in Figure A.2, it is not possible to generate encoded symbol set of *01*. So we compute the metric as we proceed. The metric is the bit difference of the received symbol set from the possible symbol set given

in Figure A.2. For instance, the solid line from the start state of Figure A.2 corresponds for the output of *00*, hence the difference of bits with this set (00) and the received set of the symbols (01) is 1.

We mark this metric at the corresponding node. The metric computation continues until the trellis grows fully, (i.e. the pattern of trellis stays the same after some transition) and the value of metric gets accumulated as decoding proceeds. From the third level, when $j = 3$[3], we eliminate the unlikely paths. The unlikely path according to the algorithm is the path which results in the largest metric. We only keeps the paths with the smallest and the second smallest metrics and call such paths *survivors*. .

After 5 steps, it is found that the path with the smallest metric is the path straight path at the top. This path generates sequence of all-zeros, which is the exact match with the original bit sequence transmitted.

For next example, suppose we have the received sequence of 110001000....., which contains three errors compared to all-zero-sequences. The same decoding algorithm is applied to this received sequence. The step-by-step decoding is shown in Figure A.5.

We can see that the correct path has been eliminated by level $j = 3$. Clearly, a triple-error pattern is uncorrectable by the Viterbi algorithm when it is applied to a convolutional code of rate $1/2$ and the constraint length $K = 3$. More discussions on this is included in A.3.

A.3 Property of Free Distance

The performance of a convolutional code depends not only on the decoding algorithm used but also the distance properties of the code. In this context, the most important single measure of a convolutional code's ability to combat channel noise is the *free distance*, denoted by d_{free}. The free distance of a convolutional code is defined as the *minimum Hamming distance*[4] between any two code words in the code. A convolutional code with free distance d_{free} can correct t errors if and only if d_{free} is greater than $2t$.

The free distance can be obtained quite simply from the state diagram of the convolutional encoder (see Figure A.3). Any non-zero code sequence corresponds to a complete path beginning and ending at the 00 state (or state A). Thus, we can split state A in the manner shown in Figure A.6.

[3]the variable j is used to identify how many step is taken to decode the sequence

[4]Hamming distance is the bit difference between tow code words

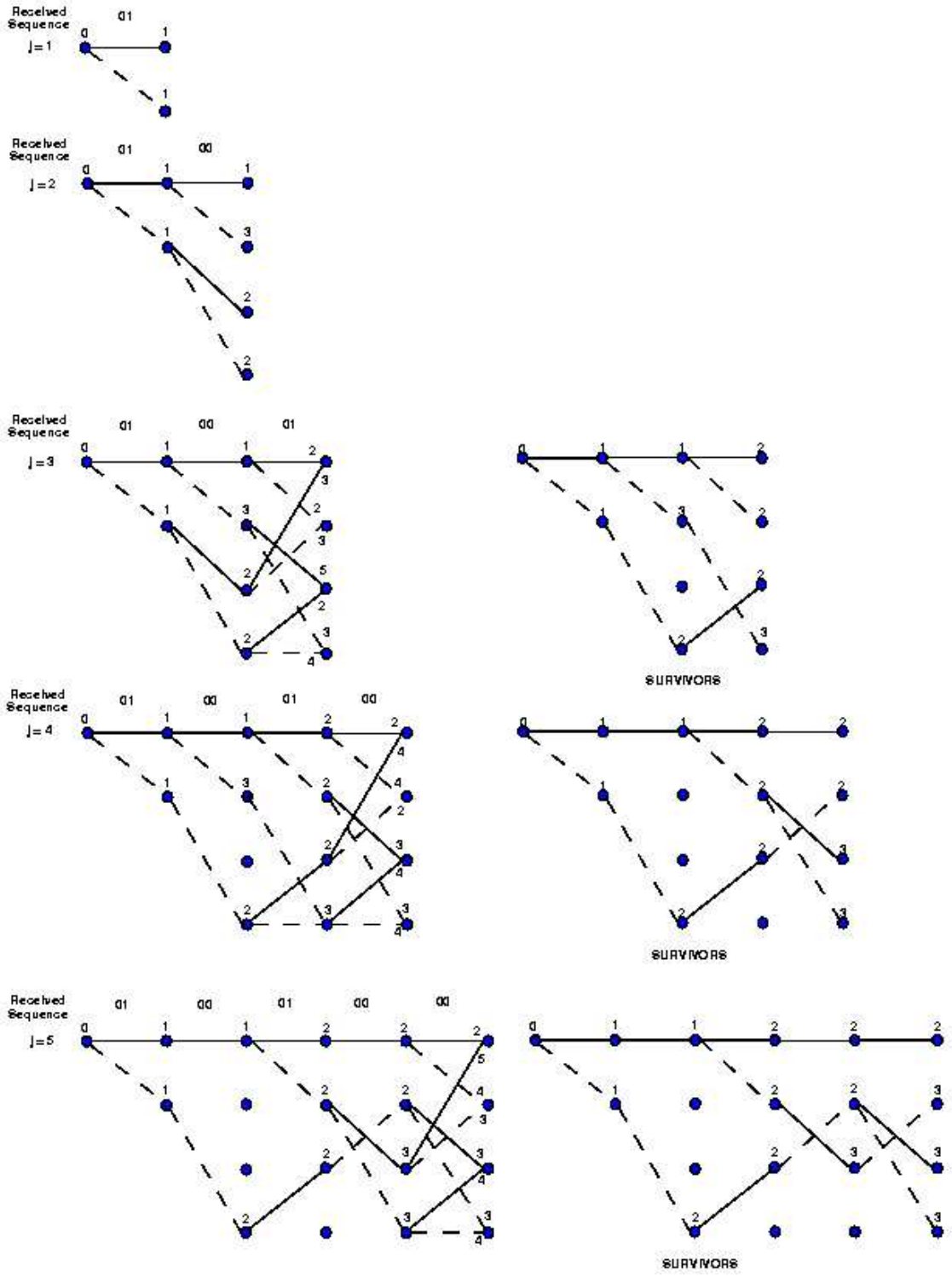

Figure A.4: Viterbi decoding when received sequence is 010001000...

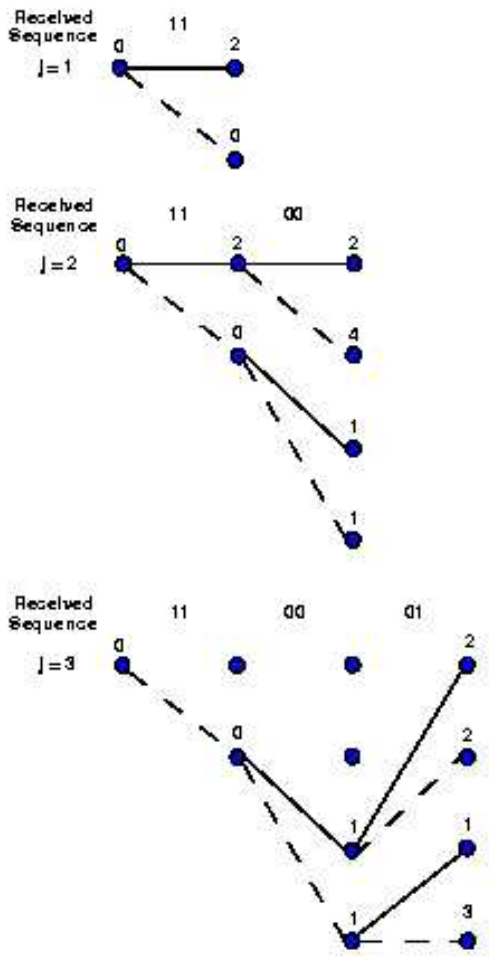

Figure A.5: Viterbi decoding when received sequence is 110001000... (only survivor paths are shown)

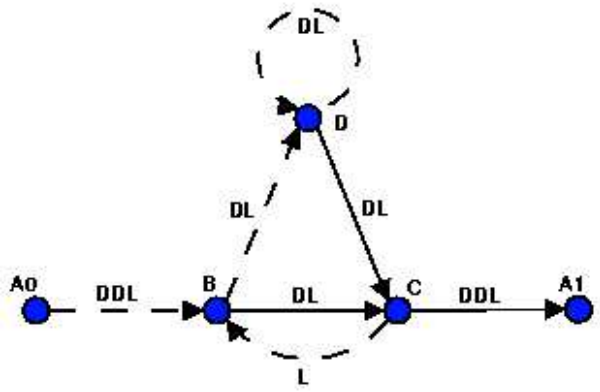

Figure A.6: Modified version of Figure A.3

One of methods used to obtain the value of d_{free} is to calculate a transfer function of the graph shown in Figure A.6. Let $T(D, L)$ denote the transfer function of the signal flow graph shown in Figure A.6, with D and L are the dummy variables. The variable L is attached to every branch and the variable D represents the number of 1's generated by going that branch. The rules for this graph is as follows

1. A branch multiplies the signal as its input node by the *transmittance* characterizing the branch (use of variable D).

2. A node with incoming breeches *sums* the signals produced by all of those branches.

3. The signal at a node is applied equally to all the branches outgoing from that node.

4. The *transfer function* of the graph is the ratio of the output signal to the input signal.

For the example of Figure A.6, we may readily use rules 1, 2 and 3 to obtain the following input-output relations: (Note that the notation DD in Figure A.6 is equivalent to the notation of D^2 in the following equations)

$$b = D^2La_0 + Lc$$

$$c = DLb + DLd$$

$$d = DLb + DLd$$

$$a_1 = D^2Lc$$

where a_0, a_1, b, c, and d denote the node of the graph. Solving the set of the equations for a_1/a_0, we thus find he transfer function of the graph shown in Figure A.6. The transfer function is given by

$$T(D, L) = \frac{a_1}{a_0} = \frac{D^5 L^3}{1 - DL(1 + L)} \tag{A.3}$$

Using the binomial expansion, we may find equivalently write (see [28] for more detail)

$$T(D, L) = D^5 L^3 \sum_{i=0}^{\infty} (DL(1 + L)^i \tag{A.4}$$

Setting $L = 1$, we thus get the *distance transfer function* expressed in the form of a power series:

$$T(D) = D^5 + 2D^6 + 4D^7 + \ldots \tag{A.5}$$

Since the free distance is the minimum Hamming distance between any two code words in the code and the distance transfer function $T(D)$ enumerates the number of code words that are a given distance apart, it follows that the exponent of the first term in the expansion of $T(D)$ defines the free distance. Thus the convolutional encoder shown in Figure A.1 has a free distance $d_{free} = 5$. Let us consider Figure A.2. If we compare the path of trellis which stays at the state A and the path which splits from state A and re-merges at state A after 3 incoming bits (3 transition of the states), we can find that the first path generates 000000 code word and the second path generates 111011 code word. The Hamming distance between two is exactly 5.

As shown in the decoding examples the code only corrects up to 2 bits; it fails to correct 3 bit errors. However, if those erroneous bits are spread far apart the decoder can correct the error. More specifically, it only success to correct the error, if and only if up to 2 erroneous bits appear in a code word, which is produced by the bit stream of length equal to the constraint length of the encoder. For example. the constraint length of the encoder shown in Figure A.1 is 3, and the received sequence given for the first decoding example is 0100010000. The first six bits, which is produced by 3 incoming bits (i.e., $R_c = 1/2$), have only two errors, hence the decoder is able to correct them. However, for the second sequence, 1100010000, the first six bits have three errors and the decoder fails to recover the original sequence.

A.4 Note for convolutional encoders defined for UMTS

The constraint length of two convolutional encoders defined for UMTS is 9. That is, they have eight registers, and have $2^8 = 256$ states. Hence, it is not feasible method to find the value of d_{free} in a conventional way discussed previously. Since only nine bits (= constraint length) are required to compute the code word of the encoders, all the possible nine-bit sequences (i.e., from 000000000 to 111111111) were fed into the encoder and all the corresponding code words are stored and compared to compute the value of d_{free}.

Another, parameter used to determine the bit error rate (see Equation 3.13), is β_d. It is the total number of information errors produced by the wrong paths of Hamming weight d, that diverge from the correct path and re-merge with it at some later stage. It can be easily recognizable when we consider Figure A.5. The correct path should stay at the top state, but the decoding produced a path which moves from $A \rightarrow B \rightarrow C \rightarrow A$. The path diverges from the correct one and re-merges after three stages or state transition. The Hamming distance between this path and the correct on is 2. In order to find out the values for β_d all those paths should be compared. We implemented the program which computes β_d for $d \geq d_{free}$ and used to eventually find out the values for β_d.

Appendix B

BHSIP and BHSDP Binary Sequences

This appendix includes the binary sequences defined for BHSIP and BHSDP to replace SIP and SDP key words.

B.1 BHSIP

SIP key words and corresponding BHSIP binary sequence

SIP	BHSIP	SIP	BHSIP
INVITE	0x01	REGISTER	0x02
CANCEL	0x03	INFO	0x04
PRACK	0x05	UPDATE	0x06
SUBSCRIBE	0x07	NOTIFY	0x08
MESSAGE	0x09	REFER	0x0a
ACK	0x0b	OPTIONS	0x0c
BYE	0x0d	COMET	0x0e
Accept	0x01	Accept-Encording	0x02
Accept-Language	0x03	Alert-Info	0x04
Allow	0x05	Authentication-Info	0x06
Authorization	0x07	Call-ID	0x08
Call-Info	0x09	Contact	0x0a
Content-Disposition	0x0b	Content-Encoding	0x0c

SIP key words and corresponding BHSIP binary sequence

Content-Language	0x0d	Content-Length	0x0e
Content-Type	0x000f	CSeq	0x10
Date	0x11	Error-Info	0x12
Expires	0x13	From	0x14
In-Reply-To	0x15	Max-Forwards	0x16
Min-Expires	0x17	MIME-Version	0x18
Organisation	0x19	Priority	0x1a
Proxy-Authenticate	0x1b	Proxy-Authorization	0x1c
Proxy-Require	0x1d	Record-Route	0x1e
Reply-To	0x1f	Require	0x20
Retry-After	0x21	Route	0x22
Server	0x23	Subject	0x24
Supported	0x25	Timestamp	0x26
To	0x27	Unsupported	0x28
User-Agent	0x29	Via	0x2a
Warning	0x2b	WWW-Authenticate	0x2c
Security-Client	0x2d	Security-Server	0x2e
Security-Verify	0x2f	Path	0x30
P-Preferred-Identity	0x31	P-Associated-URI	0x32
P-Called-Party-ID	0x33	P-Visited-Network-ID	0x34
P-Access-Network-Info	0x35	P-Charging-Vector	0x36
P-Charging-Function-Addresses	0x37	Accept-Contact	0x38
Reject-Contact	0x39	Request-Disposition	0x003a
Proxy-Max-Size	0x3b	Proxy-Seen-Size	0x3c
Join	0x3d	Refer-To	0x3e
Referred-By	0x3f	Replaces	0x40
Service-Route	0x41	text	0x01
plain	0x01	enriched	0x02
richtext	0x03	html	0x04
xml	0x05	htm	0x06

SIP key words and corresponding BHSIP binary sequence

MIME	0x07	image	0x02
jpeg	0x08	gif	0x09
audio	0x03	basic	0x0a
video	0x04	mpeg	0x0b
application	0x05	octet-stream	0x0c
PostScript	0x0d	sdp	0x0e
tls	0x01	digest	0x02
ipsec-ike	0x03	ipsec-man	0x04
others	0x05	non-urgent	0x01
normal	0x02	urgent	0x03
emergency	0x04		

B.2 BHSDP

SDP key words and corresponding BHSDP binary sequence

SDP	BHSDP	SDP	BHSDP
v	0x01	o	0x02
s	0x03	u	0x04
e	0x05	p	0x06
h	0x07	z	0x08
t	0x11	r	0x12
m	0x21	i	0x22
c	0x23	b	0x24
k	0x25	a	0x26
IN	0x01	IP4	0x11
IP6	0x12	SDP	0x01
CT	0x01	AS	0x02
X-	0x10	clear:	0x01
base64:	0x02	uri:	0x03

SDP key words and corresponding BHSDP binary sequence

prompt	0x04	media	0x02
proto	0x03	smt	0x04
att-field	0x05	bwtype	0x06
nettype	0x07	addrtype	0x08
rtpmap:	0x09	cat:	0x11
keywds:	0x12	tools:	0x13
ptime	0x14	recvonly	0x15
sendrecv	0x16	sendonly	0x17
orient:	0x18	type:	0x19
broadcast	0x1a	meeting	0x1b
moderated	0x1c	test	0x1d
H332	0x1e	charset:	0x1f
sdplang:	0x20	lang:	0x21
framerate:	0x22	quality:	0x23
fmtp:	0x24	audio	0x01
video	0x02	application	0x03
data	0x04	control	0x05
RTP/AVP	0x06	udp	0x07

Appendix C

Effective E_b/N_0 Values

This appendix shows the values of effective E_b/N_0 obtained for this project. The values in the table are in decibel (dB). Each column of the table illustrates the effective E_b/N_0 value of a given encoding rate, speed and interleaving span (TTI).

Effective E_b/N_0 values

TTI $N_i = 1$						
$(E_b/N_0)_{avg}$	1km/h		4km/h		50km/h	
	1/2	1/3	1/2	1/3	1/2	1/3
1	-16.24	-16.26	-14.88	-14.89	-5.31	-5.18
2	-15.24	-15.26	-13.88	-13.89	-4.31	-4.18
3	-14.24	-14.26	-12.88	-12.89	-3.31	-3.18
4	-13.24	-13.26	-11.88	-11.89	-2.31	-2.18
5	-12.24	-12.26	-10.88	-10.89	-1.31	-1.18
6	-11.24	-11.26	-9.88	-9.89	-0.31	-0.18
7	-10.24	-10.26	-8.88	-8.89	0.69	0.82
8	-9.24	-9.26	-7.88	-7.89	1.69	1.82
9	-8.24	-8.26	-6.88	-6.89	2.69	2.82
10	-7.24	-7.26	-5.88	-5.89	3.69	3.82
11	-6.24	-6.26	-4.88	-4.89	4.69	4.82
12	-5.24	-5.26	-3.88	-3.89	5.69	5.82
13	-4.24	-4.26	-2.88	-2.89	6.69	6.82

Effective E_b/N_0 values

14	-3.24	-3.26	-1.88	-1.89	7.69	7.82
15	-2.24	-2.26	-0.88	-0.89	8.69	8.82
16	-1.24	-1.26	0.12	0.11	9.69	9.82
17	-0.24	-0.26	1.12	1.11	10.69	10.82
18	0.76	0.74	2.12	2.11	11.69	11.82
19	1.76	1.74	3.12	3.11	12.69	12.82
20	2.76	2.74	4.12	4.11	13.69	13.82
21	3.76	3.74	5.12	5.11	14.69	14.82
22	4.76	4.74	6.12	6.11	15.69	15.82
23	5.76	5.74	7.12	7.11	16.69	16.82

TTI $N_i = 2$						
$(E_b/N_0)_{avg}$	1km/h		4km/h		50km/h	
	1/2	1/3	1/2	1/3	1/2	1/3
1	-14.55	-14.52	-12.02	-11.95	-3.83	-3.63
2	-13.55	-13.52	-11.02	-10.95	-2.83	-2.63
3	-12.55	-12.52	-10.02	-9.95	-1.83	-1.63
4	-11.55	-11.52	-9.02	-8.95	-0.83	-0.63
5	-10.55	-10.52	-8.02	-7.95	0.17	0.37
6	-9.55	-9.52	-7.02	-6.95	1.17	1.37
7	-8.55	-8.52	-6.02	-5.95	2.17	2.37
8	-7.55	-7.52	-5.02	-4.95	3.17	3.37
9	-6.55	-6.52	-4.02	-3.95	4.17	4.37
10	-5.55	-5.52	-3.02	-2.95	5.17	5.37
11	-4.55	-4.52	-2.02	-1.95	6.17	6.37
12	-3.55	-3.52	-1.02	-0.95	7.17	7.37
13	-2.55	-2.52	-0.02	0.05	8.17	8.37
14	-1.55	-1.52	0.98	1.05	9.17	9.37
15	-0.55	-0.52	1.98	2.05	10.17	10.37
16	0.45	0.48	2.98	3.05	11.17	11.37
17	1.45	1.48	3.98	4.05	12.17	12.37

Effective E_b/N_0 values

18	2.45	2.48	4.98	5.05	13.17	13.37
19	3.45	3.48	5.98	6.05	14.17	14.37
20	4.45	4.48	6.98	7.05	15.17	15.37
21	5.45	5.48	7.98	8.05	16.17	16.37
22	6.45	6.48	8.98	9.05	17.17	17.37
23	7.45	7.48	9.98	10.05	18.17	18.37

TTI $N_i = 4$						
$(E_b/N_0)_{avg}$	1km/h		4km/h		50km/h	
	1/2	1/3	1/2	1/3	1/2	1/3
1	-12.06	-12.02	-7.91	-7.84	-3.91	-2.90
2	-11.06	-11.02	-6.91	-6.84	-2.91	-1.90
3	-10.06	-10.02	-5.91	-5.84	-1.91	-0.90
4	-9.06	-9.02	-4.91	-4.84	-0.91	0.10
5	-8.06	-8.02	-3.91	-3.84	0.09	1.10
6	-7.06	-7.02	-2.91	-2.84	1.09	2.10
7	-6.06	-6.02	-1.91	-1.84	2.09	3.10
8	-5.06	-5.02	-0.91	-0.84	3.09	4.10
9	-4.06	-4.02	0.09	0.16	4.09	5.10
10	-3.06	-3.02	1.09	1.16	5.09	6.10
11	-2.06	-2.02	2.09	2.16	6.09	7.10
12	-1.06	-1.02	3.09	3.16	7.09	8.10
13	-0.06	-0.02	4.09	4.16	8.09	9.10
14	0.94	0.98	5.09	5.16	9.09	10.10
15	1.94	1.98	6.09	6.16	10.09	11.10
16	2.94	2.98	7.09	7.16	11.09	12.10
17	3.94	3.98	8.09	8.16	12.09	13.10
18	4.94	4.98	9.09	9.16	13.09	14.10
19	5.94	5.98	10.09	10.16	14.09	15.10
20	6.94	6.98	11.09	11.16	15.09	16.10
21	7.94	7.98	12.09	12.16	16.09	17.10

Effective E_b/N_0 values

22	8.94	8.98	13.09	13.16	17.09	18.10
23	9.94	9.98	14.09	14.16	18.09	19.10

TTI $N_i = 8$						
$(E_b/N_0)_{avg}$	1km/h		4km/h		50km/h	
	1/2	1/3	1/2	1/3	1/2	1/3
1	-9.78	-9.70	-4.69	-4.64	-3.26	-2.14
2	-8.78	-8.70	-3.69	-3.64	-2.26	-1.14
3	-7.78	-7.70	-2.69	-2.64	-1.26	-0.14
4	-6.78	-6.70	-1.69	-1.64	-0.26	0.86
5	-5.78	-5.70	-0.69	-0.64	0.74	1.86
6	-4.78	-4.70	0.31	0.36	1.74	2.86
7	-3.78	-3.70	1.31	1.36	2.74	3.86
8	-2.78	-2.70	2.31	2.36	3.74	4.86
9	-1.78	-1.70	3.31	3.36	4.74	5.86
10	-0.78	-0.70	4.31	4.36	5.74	6.86
11	0.22	0.30	5.31	5.36	6.74	7.86
12	1.22	1.30	6.31	6.36	7.74	8.86
13	2.22	2.30	7.31	7.36	8.74	9.86
14	3.22	3.30	8.31	8.36	9.74	10.86
15	4.22	4.30	9.31	9.36	10.74	11.86
16	5.22	5.30	10.31	10.36	11.74	12.86
17	6.22	6.30	11.31	11.36	12.74	13.86
18	7.22	7.30	12.31	12.36	13.74	14.86
19	8.22	8.30	13.31	13.36	14.74	15.86
20	9.22	9.30	14.31	14.36	15.74	16.86
21	10.22	10.30	15.31	15.36	16.74	17.86
22	11.22	11.30	16.31	16.36	17.74	18.86
23	12.22	12.30	17.31	17.36	18.74	19.86

Appendix D

Probability of Coding Block Loss P_{loss}

Depending on the convolutional encoder rate, the value of interleaving TTI and the code block size, the analytical estimations of the probabilities of code block loss, P_{loss}, are included in this appendix

The values in shown in table are the probablities which the code block is lost. For example, the probablity of $1.00000 \times 10^{+00}$ implies the complete loss of the code block.

D.1 Terminal Moving at 1km/h

Prbability of Code Block Loss Rate

Coding Rate = $1/2$, Code Block Size = 504 bits				
$(E_b/N_0)_{eff}$	$N_i = 1$	$N_i = 2$	$N_i = 4$	$N_i = 8$
1 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
2 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
3 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
4 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
5 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
6 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
7 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
8 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
9 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$

Prbability of Code Block Loss Rate

10 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
11 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	1.00000×10^{-00}
12 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	9.45627×10^{-01}
13 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	1.61395×10^{-02}
14 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	9.98393×10^{-01}	1.84329×10^{-06}
15 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	1.01578×10^{-01}	2.19888×10^{-11}
16 dB	$1.00000 \times 10^{+00}$	1.00000×10^{-00}	2.88350×10^{-05}	2.85388×10^{-17}
17 dB	$1.00000 \times 10^{+00}$	7.13570×10^{-01}	6.05636×10^{-10}	2.24453×10^{-24}
18 dB	9.99964×10^{-01}	2.44661×10^{-03}	1.50647×10^{-15}	4.27284×10^{-33}
19 dB	2.66231×10^{-01}	1.49758×10^{-07}	2.76744×10^{-22}	$0.00000 \times 10^{+00}$
20 dB	1.66628×10^{-04}	1.08805×10^{-12}	1.62480×10^{-30}	$0.00000 \times 10^{+00}$
21 dB	5.15410×10^{-09}	7.75405×10^{-19}	1.75195×10^{-40}	$0.00000 \times 10^{+00}$
22 dB	1.94428×10^{-14}	2.76360×10^{-26}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
23 dB	6.09451×10^{-21}	1.81893×10^{-35}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$

Coding Rate = $1/3$, Code Block Size = 504 bits				
$(E_b/N_0)_{eff}$	$N_i = 1$	$N_i = 2$	$N_i = 4$	$N_i = 8$
1 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
2 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
3 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
4 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
5 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
6 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
7 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
8 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
9 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
10 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
11 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	1.00000×10^{-00}
12 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	5.55610×10^{-01}
13 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	1.11970×10^{-03}
14 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	9.33782×10^{-01}	6.44752×10^{-08}

Prbability of Code Block Loss Rate

15 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	1.50069×10^{-02}	4.98691×10^{-13}
16 dB	$1.00000 \times 10^{+00}$	9.99973×10^{-01}	1.91274×10^{-06}	4.27968×10^{-19}
17 dB	$1.00000 \times 10^{+00}$	2.93969×10^{-01}	2.80482×10^{-11}	2.16598×10^{-26}
18 dB	9.95759×10^{-01}	2.30048×10^{-04}	5.16270×10^{-17}	2.46339×10^{-35}
19 dB	7.54960×10^{-02}	9.03963×10^{-09}	7.15114×10^{-24}	$0.00000 \times 10^{+00}$
20 dB	2.02547×10^{-05}	4.85750×10^{-14}	3.15986×10^{-32}	$0.00000 \times 10^{+00}$
21 dB	4.73200×10^{-10}	2.65674×10^{-20}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
22 dB	1.47370×10^{-15}	7.37698×10^{-28}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
23 dB	4.03629×10^{-22}	3.84904×10^{-37}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$

Coding Rate = $1/2$, Code Block Size = 304 bits				
$(E_b/N_0)_{eff}$	$N_i = 1$	$N_i = 2$	$N_i = 4$	$N_i = 8$
1 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
2 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
3 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
4 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
5 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
6 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
7 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
8 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
9 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
10 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
11 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	1.00000×10^{-00}
12 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	7.53551×10^{-01}
13 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	4.18406×10^{-03}
14 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	9.68405×10^{-01}	4.07481×10^{-07}
15 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	3.14060×10^{-02}	4.82618×10^{-12}
16 dB	$1.00000 \times 10^{+00}$	9.99993×10^{-01}	6.44565×10^{-06}	6.26277×10^{-18}
17 dB	$1.00000 \times 10^{+00}$	4.07101×10^{-01}	1.32972×10^{-10}	4.92555×10^{-25}
18 dB	9.96790×10^{-01}	5.84232×10^{-04}	3.30592×10^{-16}	9.37660×10^{-34}
19 dB	9.92687×10^{-02}	3.29682×10^{-08}	6.07306×10^{-23}	$0.00000 \times 10^{+00}$

Prbability of Code Block Loss Rate

20 dB	3.78092×10^{-05}	2.38783×10^{-13}	3.56558×10^{-31}	$0.00000 \times 10^{+00}$
21 dB	1.13222×10^{-09}	1.70160×10^{-19}	3.84460×10^{-41}	$0.00000 \times 10^{+00}$
22 dB	4.26673×10^{-15}	6.06463×10^{-27}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
23 dB	1.33742×10^{-21}	3.99158×10^{-36}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$

Coding Rate = $1/3$, Code Block Size = 304 bits				
$(E_b/N_0)_{eff}$	$N_i = 1$	$N_i = 2$	$N_i = 4$	$N_i = 8$
1 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
2 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
3 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
4 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
5 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
6 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
7 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
8 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
9 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
10 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
11 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	1.00000×10^{-00}
12 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	7.53551×10^{-01}
13 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	4.18406×10^{-03}
14 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	9.68405×10^{-01}	4.07481×10^{-07}
15 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	3.14060×10^{-02}	4.82618×10^{-12}
16 dB	$1.00000 \times 10^{+00}$	9.99993×10^{-01}	6.44565×10^{-06}	6.26277×10^{-18}
17 dB	$1.00000 \times 10^{+00}$	4.07101×10^{-01}	1.32972×10^{-10}	4.92555×10^{-25}
18 dB	9.96790×10^{-01}	5.84232×10^{-04}	3.30592×10^{-16}	9.37660×10^{-34}
19 dB	9.92687×10^{-02}	3.29682×10^{-08}	6.07306×10^{-23}	$0.00000 \times 10^{+00}$
20 dB	3.78092×10^{-05}	2.38783×10^{-13}	3.56558×10^{-31}	$0.00000 \times 10^{+00}$
21 dB	1.13222×10^{-09}	1.70160×10^{-19}	3.84460×10^{-41}	$0.00000 \times 10^{+00}$
22 dB	4.26673×10^{-15}	6.06463×10^{-27}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
23 dB	1.33742×10^{-21}	3.99158×10^{-36}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$

D.2 Terminal Moving at 4km/h

Code Block Loss Rate

Coding Rate = $1/2$, Code Block Size = 504 bits				
$(E_b/N_0)_{eff}$	$N_i = 1$	$N_i = 2$	$N_i = 4$	$N_i = 8$
1 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
2 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
3 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
4 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
5 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
6 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	1.00000×10^{-00}
7 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	8.82256×10^{-01}
8 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	7.94496×10^{-03}
9 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	7.01891×10^{-07}
10 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	9.86095×10^{-01}	6.90914×10^{-12}
11 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	3.96852×10^{-02}	7.12761×10^{-18}
12 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	6.67389×10^{-06}	4.14322×10^{-25}
13 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	1.03295×10^{-10}	5.23130×10^{-34}
14 dB	$1.00000 \times 10^{+00}$	9.96886×10^{-01}	1.81816×10^{-16}	$0.00000 \times 10^{+00}$
15 dB	$1.00000 \times 10^{+00}$	7.95405×10^{-02}	2.13120×10^{-23}	$0.00000 \times 10^{+00}$
16 dB	$1.00000 \times 10^{+00}$	1.93861×10^{-05}	6.87156×10^{-32}	$0.00000 \times 10^{+00}$
17 dB	9.79751×10^{-01}	3.74343×10^{-10}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
18 dB	3.19805×10^{-02}	8.47732×10^{-16}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
19 dB	4.86291×10^{-06}	1.37918×10^{-22}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
20 dB	7.05560×10^{-11}	6.88404×10^{-31}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
21 dB	1.15231×10^{-16}	1.75195×10^{-40}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
22 dB	1.22485×10^{-23}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
23 dB	3.47159×10^{-32}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$

Coding Rate = $1/3$, Code Block Size = 504 bits				
$(E_b/N_0)_{eff}$	$N_i = 1$	$N_i = 2$	$N_i = 4$	$N_i = 8$
1 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$

Code Block Loss Rate

2 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
3 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
4 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
5 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
6 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	9.99999×10^{-01}
7 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	4.64069×10^{-01}
8 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	6.71892×10^{-04}
9 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	1.00000×10^{-00}	3.40154×10^{-08}
10 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	7.63634×10^{-01}	2.33623×10^{-13}
11 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	3.67041×10^{-03}	1.73285×10^{-19}
12 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	2.93213×10^{-07}	7.22359×10^{-27}
13 dB	$1.00000 \times 10^{+00}$	1.00000×10^{-00}	3.01034×10^{-12}	6.29599×10^{-36}
14 dB	$1.00000 \times 10^{+00}$	8.86791×10^{-01}	3.63969×10^{-18}	$0.00000 \times 10^{+00}$
15 dB	$1.00000 \times 10^{+00}$	9.09409×10^{-03}	2.90126×10^{-25}	$0.00000 \times 10^{+00}$
16 dB	1.00000×10^{-00}	9.68933×10^{-07}	6.15252×10^{-34}	$0.00000 \times 10^{+00}$
17 dB	8.26607×10^{-01}	1.24732×10^{-11}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
18 dB	5.60026×10^{-03}	1.97212×10^{-17}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
19 dB	5.09017×10^{-07}	2.23838×10^{-24}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
20 dB	5.79885×10^{-12}	7.57606×10^{-33}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
21 dB	7.93675×10^{-18}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
22 dB	7.45123×10^{-25}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
23 dB	1.96909×10^{-33}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$

Coding Rate = $1/2$, Code Block Size = 304 bits				
$(E_b/N_0)_{eff}$	$N_i = 1$	$N_i = 2$	$N_i = 4$	$N_i = 8$
1 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
2 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
3 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
4 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
5 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
6 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	1.00000×10^{-00}

Code Block Loss Rate

7 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	6.24115×10^{-01}
8 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	1.98265×10^{-03}
9 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	1.00000×10^{-00}	1.54847×10^{-07}
10 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	8.86909×10^{-01}	1.51636×10^{-12}
11 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	1.10104×10^{-02}	1.56413×10^{-18}
12 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	1.48119×10^{-06}	9.09216×10^{-26}
13 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	2.26741×10^{-11}	1.14799×10^{-34}
14 dB	$1.00000 \times 10^{+00}$	9.53149×10^{-01}	3.98991×10^{-17}	$0.00000 \times 10^{+00}$
15 dB	$1.00000 \times 10^{+00}$	2.37892×10^{-02}	4.67685×10^{-24}	$0.00000 \times 10^{+00}$
16 dB	1.00000×10^{-00}	4.32348×10^{-06}	1.50794×10^{-32}	$0.00000 \times 10^{+00}$
17 dB	8.59390×10^{-01}	8.21836×10^{-11}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
18 dB	8.70770×10^{-03}	1.86033×10^{-16}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
19 dB	1.07803×10^{-06}	3.02656×10^{-23}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
20 dB	1.54871×10^{-11}	1.51068×10^{-31}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
21 dB	2.52871×10^{-17}	3.84460×10^{-41}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
22 dB	2.68790×10^{-24}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
23 dB	7.61828×10^{-33}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$

Coding Rate = $1/3$, Code Block Size = 304 bits				
$(E_b/N_0)_{eff}$	$N_i = 1$	$N_i = 2$	$N_i = 4$	$N_i = 8$
1 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
2 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
3 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
4 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
5 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
6 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	9.99646×10^{-01}
7 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	2.07962×10^{-01}
8 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	1.55609×10^{-04}
9 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	9.99997×10^{-01}	7.47900×10^{-09}
10 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	4.60773×10^{-01}	5.12698×10^{-14}
11 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	8.87940×10^{-04}	3.80268×10^{-20}

Code Block Loss Rate

12 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	6.46003×10^{-08}	1.58519×10^{-27}
13 dB	$1.00000 \times 10^{+00}$	1.00000×10^{-00}	6.60666×10^{-13}	1.38163×10^{-36}
14 dB	$1.00000 \times 10^{+00}$	6.31917×10^{-01}	7.98720×10^{-19}	$0.00000 \times 10^{+00}$
15 dB	$1.00000 \times 10^{+00}$	2.28411×10^{-03}	6.36672×10^{-26}	$0.00000 \times 10^{+00}$
16 dB	9.99999×10^{-01}	2.13890×10^{-07}	1.35015×10^{-34}	$0.00000 \times 10^{+00}$
17 dB	5.39372×10^{-01}	2.73758×10^{-12}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
18 dB	1.37644×10^{-03}	4.32777×10^{-18}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
19 dB	1.12236×10^{-07}	4.91205×10^{-25}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
20 dB	1.27267×10^{-12}	1.66254×10^{-33}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
21 dB	1.74170×10^{-18}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
22 dB	1.63515×10^{-25}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
23 dB	4.32111×10^{-34}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$

D.3 Terminal Moving at 50km/h

Code Block Loss Rate

Coding Rate = $1/2$, Code Block Size = 504 bits				
$(E_b/N_0)_{eff}$	$N_i = 1$	$N_i = 2$	$N_i = 4$	$N_i = 8$
1 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
2 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
3 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
4 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
5 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	9.99978×10^{-01}
6 dB	$1.00000 \times 10^{+00}$	9.66082×10^{-01}	9.86133×10^{-01}	2.89541×10^{-01}
7 dB	9.99994×10^{-01}	2.29236×10^{-02}	3.97446×10^{-02}	1.98478×10^{-04}
8 dB	3.51142×10^{-01}	3.01535×10^{-06}	6.68864×10^{-06}	6.39252×10^{-09}
9 dB	3.03220×10^{-04}	3.97106×10^{-11}	1.03571×10^{-10}	2.51442×10^{-14}
10 dB	1.07895×10^{-08}	5.79185×10^{-17}	1.82394×10^{-16}	8.31227×10^{-21}
11 dB	4.69778×10^{-14}	5.30885×10^{-24}	2.13943×10^{-23}	1.06721×10^{-28}

Code Block Loss Rate

12 dB	1.76660×10^{-20}	1.23416×10^{-32}	6.89976×10^{-32}	2.18994×10^{-38}
13 dB	2.69240×10^{-28}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
14 dB	6.00920×10^{-38}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
15 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
16 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
17 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
18 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
19 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
20 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
21 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
22 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
23 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$

Coding Rate = $1/3$, Code Block Size = 504 bits				
$(E_b/N_0)_{eff}$	$N_i = 1$	$N_i = 2$	$N_i = 4$	$N_i = 8$
1 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
2 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
3 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
4 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	1.00000×10^{-00}	9.80392×10^{-01}
5 dB	$1.00000 \times 10^{+00}$	9.99999×10^{-01}	8.34873×10^{-01}	3.53934×10^{-02}
6 dB	$1.00000 \times 10^{+00}$	4.46893×10^{-01}	5.94975×10^{-03}	6.43499×10^{-06}
7 dB	9.87503×10^{-01}	6.08698×10^{-04}	5.51253×10^{-07}	1.19507×10^{-10}
8 dB	4.55222×10^{-02}	3.00787×10^{-08}	6.37548×10^{-12}	2.88285×10^{-16}
9 dB	9.32795×10^{-06}	2.01923×10^{-13}	8.88340×10^{-18}	5.67894×10^{-23}
10 dB	1.86452×10^{-10}	1.45618×10^{-19}	8.53887×10^{-25}	4.02028×10^{-31}
11 dB	4.88539×10^{-16}	5.84693×10^{-27}	2.30281×10^{-33}	1.75195×10^{-40}
12 dB	1.07109×10^{-22}	4.73027×10^{-36}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
13 dB	8.74477×10^{-31}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
14 dB	1.75195×10^{-40}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
15 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
16 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$

Code Block Loss Rate

17 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
18 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
19 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
20 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
21 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
22 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
23 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
Coding Rate = $1/2$, Code Block Size = 304 bits				
$(E_b/N_0)_{eff}$	$N_i = 1$	$N_i = 2$	$N_i = 4$	$N_i = 8$
1 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
2 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
3 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
4 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
5 dB	$1.00000 \times 10^{+00}$	1.00000×10^{-00}	1.00000×10^{-00}	9.97591×10^{-01}
6 dB	$1.00000 \times 10^{+00}$	8.11063×10^{-01}	8.87089×10^{-01}	1.10434×10^{-01}
7 dB	9.98861×10^{-01}	6.08340×10^{-03}	1.10284×10^{-02}	4.51288×10^{-05}
8 dB	1.41929×10^{-01}	6.67454×10^{-07}	1.48447×10^{-06}	1.40437×10^{-09}
9 dB	6.93262×10^{-05}	8.71613×10^{-12}	2.27346×10^{-11}	5.51791×10^{-15}
10 dB	2.37084×10^{-09}	1.27100×10^{-17}	4.00260×10^{-17}	1.82410×10^{-21}
11 dB	1.03094×10^{-14}	1.16501×10^{-24}	4.69491×10^{-24}	2.34195×10^{-29}
12 dB	3.87674×10^{-21}	2.70833×10^{-33}	1.51413×10^{-32}	4.80575×10^{-39}
13 dB	5.90839×10^{-29}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
14 dB	1.31870×10^{-38}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
15 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
16 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
17 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
18 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
19 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
20 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
21 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$

Code Block Loss Rate

22 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
23 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
Coding Rate = $1/3$, Code Block Size = 304 bits				
$(E_b/N_0)_{eff}$	$N_i = 1$	$N_i = 2$	$N_i = 4$	$N_i = 8$
1 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
2 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
3 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$
4 dB	$1.00000 \times 10^{+00}$	$1.00000 \times 10^{+00}$	9.99999×10^{-01}	8.61979×10^{-01}
5 dB	$1.00000 \times 10^{+00}$	9.99562×10^{-01}	5.50874×10^{-01}	9.72017×10^{-03}
6 dB	$1.00000 \times 10^{+00}$	1.97159×10^{-01}	1.46598×10^{-03}	1.42797×10^{-06}
7 dB	8.93728×10^{-01}	1.40719×10^{-04}	1.21564×10^{-07}	2.62332×10^{-11}
8 dB	1.27942×10^{-02}	6.61292×10^{-09}	1.39923×10^{-12}	6.32633×10^{-17}
9 dB	2.07300×10^{-06}	4.43130×10^{-14}	1.94944×10^{-18}	1.24623×10^{-23}
10 dB	4.09301×10^{-11}	3.19555×10^{-20}	1.87383×10^{-25}	8.82238×10^{-32}
11 dB	1.07209×10^{-16}	1.28309×10^{-27}	5.05344×10^{-34}	3.84460×10^{-41}
12 dB	2.35047×10^{-23}	1.03804×10^{-36}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
13 dB	1.91901×10^{-31}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
14 dB	3.84460×10^{-41}	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
15 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
16 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
17 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
18 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
19 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
20 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
21 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
22 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$
23 dB	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$	$0.00000 \times 10^{+00}$

Appendix E

Message Transfer Delay

This appendix shows the average dynamic SigComp SIP message transfer delays measured over UE to P-CSCF link. The values of the delay easimations are in second. The vaules shown in this appendix is the average balues with confidence level of 95% and the relative mesure of error is kept under 10% when they are measured.

- **Case 1:** INVITE message travels from UE to Node-B
- **Case 2:** INVITE message travels from Node-B to UE

E.1 504 bit long code block

Average Message Air Travel Delay (case 1)

Data Transfer Rate	Probability of code block loss (P_{loss})				
	0%	**1%**	**2%**	**3%**	**4%**
6 kbps	1.840645	1.890050	2.059111	2.392400	2.799352
15 kbps	0.541626	0.730202	1.015104	1.361329	1.898275
30 kbps	0.272044	0.453132	0.679361	0.959613	1.539506
54 kbps	0.168606	0.331448	0.575877	0.887903	1.459761
78 kbps	0.129823	0.273227	0.521971	0.851328	1.458213
102 kbps	0.084613	0.235843	0.497043	0.829189	1.404811
126 kbps	0.073897	0.240196	0.482109	0.799389	1.381264

Average Message Air Travel Delay (case 1)

150 kbps	0.062285	0.240868	0.471444	0.781304	1.377378

Average Message Air Travel Delay (case 2)

Data Transfer Rate	Probability of code block loss (P_{loss})				
	0%	**1%**	**2%**	**3%**	**4%**
6 kbps	1.820111	1.884198	2.039866	2.331449	2.625566
15 kbps	0.529264	0.709690	0.934700	1.266377	1.852158
30 kbps	0.266660	0.447259	0.735205	0.985560	1.583701
54 kbps	0.159953	0.320457	0.566783	0.846136	1.450252
78 kbps	0.112358	0.265482	0.505482	0.778132	1.428121
102 kbps	0.080723	0.242432	0.479394	0.748267	1.395649
126 kbps	0.071056	0.241230	0.457925	0.712220	1.400733
150 kbps	0.061056	0.238280	0.434750	0.709704	1.373539

E.2 304 bit long code block

Average Message Air Travel Delay (case 1)

Data Transfer Rate	Probability of code block loss (P_{loss})			
	0%	**1%**	**2%**	**3%**
6 kbps	2.975296	3.243097	3.525394	4.661753
15 kbps	0.762765	0.954179	1.530337	2.901935
30 kbps	0.382642	0.564793	1.147291	2.486830
54 kbps	0.247321	0.485419	1.069629	2.423988
78 kbps	0.185248	0.375179	1.042341	2.350417
102 kbps	0.130007	0.342094	1.002197	2.328519
126 kbps	0.110258	0.312841	0.980264	2.301245
150 kbps	0.090520	0.311756	0.949371	2.285759

Average Message Air Travel Delay (case 2)

Data Transfer Rate	Probability of code block loss (P_{loss})			
	0%	**1%**	**2%**	**3%**
6 kbps	2.842158	3.212385	3.682132	4.521387
15 kbps	0.662135	0.835125	1.421320	2.921086
30 kbps	0.395432	0.575863	1.105481	2.541299
54 kbps	0.235812	0.435412	1.062120	2.453217
78 kbps	0.184213	0.356812	1.020842	2.405489
102 kbps	0.130218	0.344721	0.995432	2.331258
126 kbps	0.110075	0.302842	0.965421	2.299941
150 kbps	0.089020	0.303584	0.944584	2.302481

Appendix F

Glossary

The definitions of the terms used throughout this thesis are defined in this appendix. The definitions are obtained from [57], [28] and `SearchNetworking.com`

3G Third Generation or short for third generation mobile network

3GPP Third Generation Partnership Project, an European collaborative agreement of a number of telecommunication standard bodies, focusing on CDMA-2000 based wireless communications

3GPP2 Third Generation Partnership Project 2, a North-American and Asian collaborative agreement of a number of telecommunication standard bodies, focusing on W-CDMA based wireless communications

AAL5 It is the most popular ATM adaptation layer protocol used for data transmission. It is connection-oriented and supplies a variable bit rate.

ARQ Automatic Repeat reQuest. It is a protocol for error control in data transmission. When the receiver detects an error in a packet, it automatically requests the transmitter to resend the packet. This process is repeated until the packet is error free or the error continues beyond a predetermined number of transmissions.

ATM Asynchronous Transfer Mode. It is a dedicated-connection switching technology that organizes digital data into 53-byte cell units and transmits them over a physical medium using digital signal technology. Individually, a cell is processed asyn-

chronously relative to other related cells and is queued before being multiplexed over the transmission path.

AWGN Addictive White Gaussian Noise. Noise having a frequency spectrum that is continuous and uniform over a specified frequency band.

BER Bit Error Rate. see Error Rate.

BPSK Binary Phase Shift Keying. It is a digital frequency modulation technique used for sending data. This type of modulation is less efficient but also less susceptible to noise than similar modulation techniques.

CCITT A former name for ITU-T (see ITU-T)

CDMA Code-Division Multiple Access. It is a form of multiplexing, which allows numerous signals to occupy a single transmission channel, optimizing the use of available bandwidth.

CDMA-2000 also known as IMT-CDMA Multi-Carrier or 1xRTT, is a CDMA version of the IMT-2000 standard developed by ITU. The CDMA2000 standard is 3G mobile wireless technology.

CRC Cyclic Redundancy Code. It is a method of checking for errors in data that has been transmitted on a communications link.

CSCF Call Session Contor Function. Logical SIP servers in IMS; see section 2.2.2.

Data transfer rate (or just *data rate*) is the amount of digital data that is moved from one place to another in a given time, usually in a second's time. The data transfer rate can be viewed as the speed of travel of a given amount of data from one place to another. In telecommunications, data transfer rate is usually measured in bits per second (bps).

Error Rate The ratio of the number of data units in error to the total number of data unit.

FDMA Frequency-Division Multiple Aceess. It is the division of the frequency band allocated for wireless cellular telephone communication into several channels, each of which can carry a voice conversation or, with digital service, carry digital data.

FER Frame Error Rate; see Frame and Error Rate

Frame A group of bits that includes data plus one or more adresses and other protocols control information. Generally refers to a link layer (OSI layer 2) protocol data unit.

H.323 Recommendation of multimedia related routines, defined by ITU

HSS Home Subscriber Server. A logical entity of IMS, which stores information about users using an UMTS network

IETF Internet Engineering Task Force. It is the body that defines standard Internet operating protocols.

IMT-2000 International Mobile Telecommunications 2000. A formal name for 3G network technology nu ITU. The number 2000 came from the carrier frequency of approximately 2000MHz.

IMS IP Multimedia Subsystem. One of core network structures of UMTS which supports and controls IP-based services.

IP Internet Protocol. It is the protocol by which data is sent from one computer to another on the Internet.

ITU International Telecommunications Union

ITU-T Telecommunication Standardization Sector of the International Telecommunications Union. It is the primary international body for fostering cooperative standards for telecommunications equipment and systems. It was formerly known as the CCITT. It is located in Geneva, Switzerland.

Node-B The formal name for a base station used for UMTS

OSI Open Systems Interconnection. It is a standard description or ”reference model” for how messages should be transmitted between any two points in a telecommunication network.

PSTN Public Switched Telephone Network. It is the world’s collection of interconnected voice-oriented public telephone networks.

QoS Quality of Service. On the Internet and in other networks, QoS is the idea that data transmission rates, error rates, and other characteristics can be measured, improved,

and, to some extent, guaranteed in advance. QoS is of particular concern for the continuous transmission of high-bandwidth video and multimedia information. Transmitting this kind of content dependably is difficult in public networks using ordinary ”best effort” protocols.

QPSK Quadrature Phase Shift Keying. It is a digital frequency modulation technique used for sending data. Since it’s both easy to implement and fairly resistant to noise, QPSK is used primarily for sending data from the cable subscriber upstream to the Internet.

Registrar A logical entity of SIP. It stores information about users.

RNC Radio Network Controller. It controls several Node-Bs

SCIP Simple Conference Invitation Protocol. Obsolete protocol for initiating a multimedia conference over Internet. Its functionaliies are added to SIP

SDP Session Description Protocol, an application layer protocol containing the descrption of a multimedia session

SIP Session Initiation Protocol, an application layer protocol used for initiating, modifying and terminating multimedia sessions

SNR In analog and digital communications, signal-to-noise ratio, often written S/N or SNR, is a measure of signal strength relative to background noise. The ratio is usually measured in decibels (dB).

TCP Transmission Control Protocol. It is a set of rules (protocol) used along with the Internet Protocol (IP) to send data in the form of message units between computers over the Internet. While IP takes care of handling the actual delivery of the data, TCP takes care of keeping track of the individual units of data (called packets) that a message is divided into for efficient routing through the Internet.

TDMA Time-Division Multiple Access. It is a technology used in digital cellular telephone communication that divides each cellular channel into several time slots in order to increase the amount of data that can be carried.

TTI Transmission Time Interval. It is the time span of the inter-frame interleaving scheme of UMTS. It artificialy inserts the time gap between consequitive radio frames in order to avoid bust of radio frame corruptions.

UAC User Agent Client. A logical entity of SIP, which generates SIP requests.

UAS User Agent Server. A logical entity of SIP, which responces to SIP requests.

UDP User Datagram Protocol. It is a communications protocol that offers a limited amount of service when messages are exchanged between computers in a network that uses the Internet Protocol (IP). UDP is an alternative to the Transmission Control Protocol (TCP) and, together with IP, is sometimes referred to as UDP/IP.

UDVM Universal Decompressor Virtual Machine, A logical entity SigComp decompressor, which is capable of decodin any compression algorithm-based messages.

UE User Equipment. In this thesis, this term is generally refered to as an UMTS mobile terminal.

UMTS Universal Mobile Telecommunications System. European version of 3G mobile networks using W-CDMA. It is standardized by 3GPP.

URI Universal Resource Identifier. An addressing mechanism for SIP. It identifies a communications resource. Examples of communications resources include 1) user of an online service, 2) an appearance on a multi-line phone, 3) a mailbox on a messaging system, 4) a PSTN number at a gateway service, 5) a group (such as ”sales” or ”helpdesk”) in an organization

URL Uniform Resource Locator or Universal Resource Locator. It is the unique address for a file that is accessible on the Internet. A common way to get to a Web site is to enter the URL of its home page file in your Web browser’s address line. However, any file within that Web site can also be specified with a URL. Such a file might be any Web (HTML) page other than the home page, an image file, or a program such as a common gateway interface application or Java applet.

UTRAN UMTS Terrestrial Radio Access Network. It consists of RNC and Node-B. It provides wireless access using W-CDMA

VoIP Voice over IP. It is a term used in IP telephony for a set of facilities for managing the delivery of voice information using the Internet Protocol (IP). In general, this means sending voice information in digital form in discrete packets rather than in the traditional circuit-committed protocols of the public switched telephone network (PSTN).

W-CDMA Wideband Code-Division Multiple Access, an ITU standard derived from Code-Division Multiple Access (CDMA), is officially known as IMT-2000 direct spread. W-CDMA is a 3G mobile wireless technology that promises much higher data speeds to mobile and portable wireless devices .The input signals are digitized and transmitted in coded, spread-spectrum mode over a broad range of frequencies. A 5 MHz-wide carrier is used.

Bibliography

[1] BPSK Basic. Available from *http://www.computerpro.com/ lyle/watsbpsk.htm* (Accessed on June 2004).

[2] 3GPP. *"3rd Generation Partnership Project, Technical Specification Group Core Network Signaling flows for the IP multimedia call control based on SIP and SDP (Release 5)"*. 3GPP TS 24.228, *http://www.3gpp.org/ftp/Specs/html-info/24228.htm*.

[3] 3GPP. *"3rd Generation Partnership Project, Technical Specification Group Radio Access Network; Multiplexing and channel coding (FDD) (Release 5)"*. 3GPP TS 25.212, *http://www.3gpp.org/ftp/Specs/html-info/25212.htm*.

[4] 3GPP. *"3rd Generation Partnership Project, Technical Specification Group Radio Access Network; Multiplexing and channel coding (TDD) (Release 5)"*. 3GPP TS 25.222, *http://www.3gpp.org/ftp/Specs/html-info/25222.htm*.

[5] 3GPP. *"3rd Generation Partnership Project, Technical Specification Group Radio Access Network; Spreading and modulation (FDD) (Release 6)"*. 3GPP TS 25.213, *http://www.3gpp.org/ftp/Specs/html-info/25213.htm*.

[6] 3GPP. *"3rd Generation Partnership Project, Technical Specification Group Radio Access Network; Spreading and modulation (TDD) (Release 6)"*. 3GPP TS 25.223, *http://www.3gpp.org/ftp/Specs/html-info/25223.htm*.

[7] 3GPP. *"3rd Generation Partnership Project, Technical Specification Group Service and System Aspects; IP Multimedia Subsystem (IMS); Stage2, Release 5"*. 3G TS 23.228, *http://www.3gpp.org/ftp/Specs/html-info/23228.htm*.

[8] 3GPP. *"3rd Generation Partnership Project, Technical Specification Group Services*

and System Aspects; Quality of Service (QoS) concept and architecture (Release 5)". 3GPP TS 23.107, *http://www.3gpp.org/ftp/Specs/html-info/23107.htm*.

[9] Arkko, J., Torvinen, V., Camarillo, G., Niemi, A., and Haukka, T. *"Security Mechanism Agreement for the Session Initiation Protocol (SIP)"*. IETF, January 2003. RFC 3329, *http://www.ietf.org/rfc/rfc3329.txt*.

[10] Camarillo, G. *"SIP Demystified"*. McGraw-Hill, 2002. ISBN 0-07-137340-3.

[11] Chang, C. S. and Thomas, J.A. "Effective Bandwidth in High Speed Digital Network". *IEEE Journal on Selected Areas in Communications*, Vol. 13, Issue 6:1091–1100, August 1995.

[12] Cox, D. C. "Delay Doppler Characteristics of Multipath Delay Spread and Average Excess Dealy for 910MHz Urban Mobile Radio Path". *IEEE Transactions on Antennas and Propagation*, AP-20:625–635, September 1972.

[13] Cox, D. C. and Leck, R. P. "Distribution of Multipath Delay Spread and Average Excess Delay for 910 MHz Urban Mobile Radio Path". *IEEE Transactions on Antennas and Propagation*, AP-23:206–213, March 1975.

[14] Crocker D. and Overell, P. *"Augmented BNF for Syntax Specifications: ABNF"*. IETF, November 1997. RFC 2234, *http://www.ietf.org/rfc/rfc2234.txt*.

[15] Curcio, I. and Lundan, M. "SIP Call Setup Delay in 3G network". pages 835–840, July 2002.

[16] Deutsch, P. *"DEFLATE Compressed Data Format Specification"*. IETF, May 1996. RFC1951, *http://www.ietf.org/rfc/rfc1951.txt*.

[17] Devasirvatham, D. M. J., Benerjee, C., Krain, M. J., and Rappaport D. A. "Multi-frequency Radiowave Propagation Measurements in the Portable Radio Environment". In *IEEE International Conference on Communications*, pages 1334–1340, 1990.

[18] DummyNet. Dipartimento di Ingegneria dell'Informazione of the Universita di Pisa, Italy. Available from *http://info.iet.unipi.it/l̃uigi/ip_dummynet/* (Accessed on July 2003).

[19] Fathi, H., Chakraborty, S., and Prasad, R. ”Performance of SIP session setup for VoIP over wireless link”. In *Proceedings of the 7th international Symposium on Wireless Personal Multimedia Communications (WPMC)*. To be pubilished in September 2004.

[20] Feldspar, A. An explanation of the deflate algorithm, August 1997. Available from *http://www.gzip.org/zlib/feldspar.html* (Accessed on August 2003).

[21] FreeBSD. The FreeBSD Project. Available from *http://www.freebsd.org/* (Accessed on July 2003).

[22] Garcia-Martin, M., Bormann, C., Ott, J., Price, R., and Roach, A. B. *“The Session Initiation Protocol (SIP) and Session Description Protocol (SDP) Static Dictionary for Signaling Compression (SigComp)”*. IETF, February 2003. RFC3485, *http://www.ietf.org/rfc/rfc3485.txt*.

[23] Garcia-Martin, M., Henrikson, E., and Mills, D. *“Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3rd-Generation Partnership Project (3GPP)”*. IETF, January 2003. RFC3455, *http://www.ietf.org/rfc/rfc3455.txt*.

[24] Glasmann, J., Kellerer, W., and Muller, H. ”Service Development and Deployment in H.323 and SIP”. In *Computers and Communications, 2001. Proceedings. Sixth IEEE Symposium on*, pages 378–385, 2001.

[25] Grech, M., Torabi, M., and Unmehopa, M. “Service Control Architecture in the UMTS IP Multimedia Core Network Subsystem”. In *3G Mobile Communication Technologies, IEE Conference Publication*, volume Conf. Publ. No. 489, pages 22–26, May 2002.

[26] Gyasi-Agyei, A. and Coutts, R. P. “Analytical Model of a Differenriated Service Scheme over Wireless IP links”. In *10th IEEE International Conference on Networks, ICON 2002*, pages 223–228, August 2002.

[27] Handley M. and Jacobson, W. *“SDP: Session Description Protocol”*. IETF, April 1998. RFC2327, *http://www.ietf.org/rfc/rfc2327.txt*.

[28] Haykin, S. *“Communication Systems”*. John Wiley & Sons, third edition, 1994. ISBN 0-471-57176-8.

[29] Hernandez, E. and Abdelsalam, H. ”RAMON: Rapid-Mobility Network Emulator”. In *Proceedings of the 27th Annual IEEE Conference on Local Computer Networks (LCN)*, pages 809–817. IEEE, Nvember 2002.

[30] Ho, C. Y., Lam, T. K. K., and Van Nguyen Tran. “Third Generation Wireless Signal Profiling”. In *IEEE Radio Frequency Integrated Circuits Symposium*, pages 539–542, 2003.

[31] IETF Dependencies and Priorities. 3GPP. Available from *http://www.3gpp.org/TB/Other/IETF.htm* (Accessed on May 2003).

[32] IETF Draft Dependencies. 3GPP2. Available from *http://www.3gpp2.org/Public_html/IETF/IETF_Dependencies_printer.cfm* (Accessed on May 2003).

[33] Internation Telecommunication Union (ITU). *”Packet-Based Multimedia Communications Systems”*, 1998.

[34] ITU. *“Digital Exchange Performance, Design Objectives”*, March 1993. Recommendation Q.543.

[35] Jakes, W. C. *Microwave Mobile Communications*. Wiley-Interscience, 1974.

[36] Jennings,C., Peterson, J., and Watson, M. *“Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks”*. IETF, November 2002. RFC3325, *http://www.ietf.org/rfc/rfc3325.txt*.

[37] Mark, J. W. and Zhuang, W. *“Wireless Communications and Networking”*. Prentice Hall, 2003. ISBN 0-13-040905-7.

[38] MATLAB. MathWorks. Commercial Website *http://www.mathworks.com/* (Last accessed on February 2004).

[39] Mills, D. *“Network Time Protocol (version 3) Specification and Implementation”*. IETF, March 1992. RFC 1305, *http://www.ietf.org/rfc/rfc1305.txt*.

[40] Muller Veerse, F. “Mobile Commerce Report”. Available from *http://www.durlancher.com/fr-research-reps.htm/* (Accessed on July 2002).

[41] Nanda, S. and Rege, M. "Frame Error Rates for Convolutional Codes on Fading Channels and the Concept of Effective E_b/N_0". *IEEE Transaction on Vehicular Technology*, Vol. 47, No. 4:1245–1250, November 1998.

[42] NISTnet. National Institute of Standards and Technology, USA. Downloadble from *http://snad.ncsl.nist.gov/nistnet/* (Accessed on April 2003).

[43] Poppe, F., De Vleeschauwer, D., and Petit G. H. "Guranting Qulity of Service to Packetised Voice over the UMTS Air Interface". *Eighth International Workshop on Quality of Service, 2000. IWQOS. 2000*, pages 85–91, June 2000.

[44] Prasad, R. *"CDMA for Wireless Personal Communications"*. Number ISBN 0-89006-571-3 in Mobile Communications Series. Artech House Publishers, 1996.

[45] Price, R., Bormann, C., Christoffersson, J., Hannu, H., Liu, Z., and Rosenberg, J. *"Signaling Compression (SigComp)"*. IETF, January 2003. RFC3320, *http://www.ietf.org/rfc/rfc3320.txt*.

[46] Proakis, J.G. *"Digital Communications"*. McGraw-Hill, fourth edition, 2000. ISBN 0-07-232111-3.

[47] Rappaport, T. S. *"Wireless Communications Princeples and Practice"*. Prentice Hall, 1996. ISBN 0-13-375536-3.

[48] Rappaport, T. S., Seidel, S. Y., and Singh, R. "900MHz Multipath Propagation Measurements for U.S. Digital Cellualr Radiotelephone". *IEEE Transactions on Vehicular Technology*, pages 132–139, May 1990.

[49] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and Schooler, E. *"SIP: Session Initiation Protocol"*. IETF, June 2002. RFC3261, *http://www.ietf.org/rfc/rfc3261.txt*.

[50] Saleh, A. A. M. and Valenzula, R. A. "A Statistical Model for Indoor Multipath Propagation". *IEEE Journal on Selected Areas in Communication*, JSAC-5, No. 2:128–137, February 1987.

[51] Salomon, D. *"Data Compression"*. Springer-Verlag, 3rd edition, 2004. ISBN 0-387-40697-2.

[52] Salsano, S., Veltri, L., and Papalilo, D. "SIP Security Issues: The SIP Authentication Procedure and its Processing Load", Novenber/December 2002.

[53] Seawind. Department of Computer Science, University of Helsinki University, Finland. Available from *http://www.cs.helsinki.fi/research/iwtcp/seawind/* (Last accessed on March 2004).

[54] Seidel, S. Y. "The Impact of Surronding Buildings on Propagation for Wireless Inbuilding Personal Communication System Design". In *IEEE Vehicular Technology Conference*, pages 814–818, May 1992.

[55] Simmonds, R. and Unger, B. W. Towards scalable network emulation. *Elsevier Computer Communcations*, (26):264–277, 2003.

[56] Smith, J. I. "A Computer Generated Multipath Fading Simulation for Mobile Radio". *IEEE Transaction on Vehicular Technology*, VT-24, No 3.:39–40, August 1975.

[57] Stallings, W. *"Data & Computer Communications"*. Prentice Hall, sixth edition, 2000. ISBN 0-13-086388-2.

[58] Stevens R. W. *UNIX Network Programming: Networking APIs: Sockets and XTI*, volume 1. Prentice Hall, second edition, 1998. ISBN 0-13-490012-X.

[59] Tsuno, Y. "Nifty New Cellular Phone Systems Race to Capture Japan's Consumers". *IEEE Spectrum*, page 14, October 2003.

[60] Wang, L., Agarwal, A., and Atwood, J. W. "Modelling and Verification of Interworking between SIP and H.323". *Computer Networks*, 45 issue 2:77–98, June 2004.

[61] West, M., Conroy, L., Hancock, R., Price, R., and Surtees, A. "IP Header and Signaling Compression for 3G System". In *3G Mobile Communication Technologies, IEE Conference Publication*, volume Conf. Publ. No. 489, pages 102–106, May 2002.

[62] Wisely, D., Eardley, P., and Burness, L. *"IP for 3G, Networking Technologies for Mobile Communications"*. John Wiley & Sons, 2002. ISBN 0-471-48697-3.

[63] Wu, D. and Negi, R. "Effective Capacity: A Wireless Link Model for Supprt of Quality of Service". *IEEE Transaction Wireless Communications*, Vol. 2, No. 4:630–643, July 2003.

[64] Yi, S. J., Pawlikowski, K., Sirisena, H. R., and De Silva, P. ”Post-dialing Delay of Multimedia Sessions in 3G Mobile Networks”. In *Proceedings of The International Conference on Information Networking 2004 (ICOIN2004)*, volume 3, pages 1303–1312, February 2004.

[65] Young, D. J. and Beaulieu, N. C. ”The generation of correlated Rayleigh random variates by inverse discrete Fourier transform”. *IEEE Transactions on Communications*, 48:1114–1127, July 2002.