Les mécanismes de fiabilisation (protocoles ARQ) et leur adaptation dans les réseaux radiomobiles de 3G

Robert Bestak

Thèse

présentée pour obtenir le grade de docteur de
l'Ecole Nationale Supérieure des Télécommunic ations

Spécialité : Informatique et Réseaux

Robert BESTAK

Les mécanismes de fiabilisation (protocoles ARQ) et leur adaptation dans les réseaux radiomobiles de 3G

Soutenue le 18 décembre 2003 devant le jury composé de

Xavier LAGRANGE	Président
Khaldoun AL AGHA	Rapporteurs
Laurent TOUTAIN	
Philippe MARTINS	Examinateurs
Boris SIMAK	
Jérôme BROUET	Invités
Rémy ROGACKI	
Philippe GODLEWSKI	Directeur de thèse

A mes parents

Remerciements

Tout d'abord, je tiens à remercier mon directeur de thèse le professeur Philippe Godlewski. Ses conseils et ses critiques constructives m'ont beaucoup apporté tout au long de ces années. Je lui suis très reconnaissant pour sa patience et ses efforts à me guider durant ma formation scientifique.

Un grand merci aux maîtres de conférences Philippe Martins et Elie Najm qui ont contribué largement à ma formation. J'ai beaucoup apprécié nos discussions techniques qui m'ont orienté vers le but de ma recherche.

Je remercie les rapporteurs de ma thèse, Khaldoun Al Agha et Laurent Toutain, d'avoir consacré de leur temps à ma thèse et d'avoir apporté des remarques constructives. Je tiens à remercier les autres membres du jury dont Xavier Lagrange, Boris Simak, Jérôme Brouet et Rémy Rogacki de leurs commentaires pertinentes.

Je remercie tous les enseignants et thésards du Département Infres de l'ENST pour avoir créé une ambiance amicale et sympathique. Je remercie tout particulièrement mes amis avec qui j'ai partagé le bureau durant la thèse Tatiane Aubonnet, Arnaud Bailly, et Alexandre Tauveron qui m'ont donné leur soutien durant cette thèse.

Un grand merci aux étudiants de l'ENST qui m'ont permis d'enrichir mes connaissances au cours de l'encadrement des projets.

Je voudrais exprimer ma gratitude à mon entourage personnel, mes parents et mes amis qui m'ont encouragé tout au long de mes études en France, sans lesquelles je n'aurai pas finalisé cette thèse.

Résumé

Les réseaux radiomobiles de 3ème génération (3G), tel que l'UMTS (*Universal Mobile Telecommunications System*), ont été prévus, dés le début de leur conception, pour offrir un mode paquet aussi bien qu'un mode circuit; le mode circuit étant le mode unique de la 2G (cas du système GSM, *Global System for Mobile communications*).

Pour fiabiliser les transmissions de données en mode paquet, il faut mettre en œuvre des protocoles disposant de la technique ARQ (*Automatic Repeat reQuest*). L'objet de cette thèse est l'étude de protocoles ARQ qui peuvent être activés dans l'architecture protocolaire globale de l'UMTS. Cela nous amène à étudier les couches contenant les protocoles ou entités suivantes : RLC (*Radio Link Control*), TCP (*Transmission Control Protocol*) et MAC-hs (*Medium Access Control-high speed*). L'étude des nouvelles caractéristiques des protocoles ARQ de l'UMTS ainsi que la possibilité de leur enrichissement est effectuée. L'empilement des mécanismes ARQ présente des risques d'interactions dont certains sont analysés.

Le chapitre 1 présente le contexte de notre étude. Le lecteur est familiarisé aux innovations et notations de l'UMTS qui sont reprises dans les chapitres suivants. L'architecture de l'interface radio de l'UMTS et ses différents aspects sont présentés dans le paragraphe 1.2; la présentation concerne principalement l'architecture d'un point de vue protocolaire. Suite à cette présentation de l'architecture de l'UMTS dans son ensemble, le paragraphe 1.3 introduit les trois protocoles ARQ ainsi que les éléments réseaux entre lesquels ces protocoles fonctionnent (figure 4). L'étude détaillée de ces protocoles est faite dans les chapitres suivants. Suivant l'exigence aux délais et la sensibilité aux erreurs, les services de l'UMTS sont divisés en quatre classes. Celles-ci sont présentées dans le paragraphe 1.4.

La transmission fiable de données sur l'interface radio de l'UMTS est principalement réalisée par le protocole RLC, plus précisément par les entités RLC qui fonctionnent en mode acquitté (AM, *Acknowledged mode*). Le protocole est largement adaptable et des primitives spécifiques permettent de l'ajuster conformément à la qualité de service requise. Dans le paragraphe 2.3, nous analysons différents aspects des mécanismes du protocole RLC, comme la fonction *SDU discard*. Le protocole RLC de l'UMTS a été amélioré et enrichi par rapport au protocole RLC du (E)GPRS (*Enhanced, General Packet Radio Service*). Certaines des différences entre elles sont mentionnées dans le paragraphe 2.5. Le chapitre 2 est terminé par des remarques concernant l'implementation des protocoles (le paragraphe 2.6).

Nombre de services prévus pour l'UMTS sont proches des services offerts par l'Internet fixe (comme le *web-browsing*, la messagerie électronique, etc.). Ces services sont fournis via la pile protocolaire TCP/IP, base traditionnelle et obligatoire du monde Internet. L'utilisation de TCP/IP sur une interface radio a donné lieu à de nombreuses études mais seulement une partie considère une couche liaison avec un ARQ. Les mécanismes TCP sont comparés avec ceux du RLC de l'UMTS dans le paragraphe 3.4. Le paragraphe 3.5 analyse l'utilité de certaines propositions du TCP pour les réseaux sans fil comme *Split-TCP*, *Wirless-TCP*, *Snoop*, etc., dans l'environnement de l'UMTS.

L'introduction du protocole RLC au-dessous de la pile protocolaire TCP/IP est utile pour la performance globale. La grande flexibilité du RLC conduit à une longue liste de paramètres ajustables, tels que les temporisateurs, la fenêtre d'émission, le nombre maximum de tentatives de réémissions d'un bloc RLC, etc. Un de ces paramètres spécifie la taille de la mémoire tampon (*buffer*) en entrée de la couche RLC. Nous étudions l'impact des différents *soft states* du protocole TCP sur l'occupation du tampon de l'entité RLC doit être pris en compte.

L'occupation de ce tampon affecte le TCP RTT (*Round Trip Time*). Il faut distinguer les transmissions de données : (i) en voie montante et (ii) en voie descendante. Dans le premier cas, l'occupation du tampon de l'entité RLC (situé dans l'UE, *User Equipment*) est déterminé par le système d'exploitation d'UE et le contrôle interne de flux entre les protocoles RLC et TCP en plus des *soft states* du TCP. Notre étude considère uniquement le deuxième cas (voie descendante) où le contrôle interne de flux entre les deux protocoles n'existe pas. L'occupation du tampon de l'entité RLC (RNC, *Radio Network Controller*), en considérant différentes valeurs du RTT dans le réseau fixe et du BlER (*Block Error Rate*, taux d'erreurs par bloc) sur la liaison radio sont analysées dans le paragraphe 4.3. De plus, les diverses tailles du tampon de l'entité RLC sur la performance du protocole TCP sont étudiées.

Pour surmonter quelques inconvénients de l'empilement TCP/RLC, dans le paragraphe 4.4.1, nous proposons d'introduire entre les deux protocoles un mécanisme spécifique appelé *Early Error Notification* (EEN). Celui-ci est basé sur la fonction *SDU discard* du RLC et sur la possibilité d'échanger des messages de contrôle entre les protocoles TCP et RLC. Les résultats de simulations sans et avec l'EEN sont présentés dans le paragraphe 4.4.2. Le mécanisme EEN est suggéré pour les transmissions de données en voie montante. La possibilité d'utilisation de l'EEN en voie descendante est étudiée dans le paragraphe 4.4.3.

Pour améliorer les performances des transmissions de données en voie descendante, la *Release 5* de l'UMTS propose un nouveau mode d'accès, HSDPA (*High Speed Downlink Packet Access*). Le mode HSDPA introduit un mécanisme hybride ARQ (HARQ) qui concerne les niveaux physique et MAC (localisé au *Node B*). Ce mécanisme, basé sur le *Stop&Wait* ainsi sur une numérotation spécifique, est analysé dans le paragraphe 5.2.

Le mode d'allocation du HSDPA est très flexible. Néanmoins, des services comme le *streaming* n'ont pas besoin d'une telle allocation dynamique. Les services de *streaming* génèrent un flux sortant régulier de données. Pour un tel flux de données, il est plutôt envisageable d'allouer périodiquement des ressources radio. L'impact de l'allocation périodique sur l'allocation dynamique est étudié dans le paragraphe 5.3.

La majorité des études sur l'HSDPA concerne l'adaptation rapide du lien radio, la performance d'*Error Correction Code* ou les politiques d'allocation de ressources radio. En ce moment, peu d'études s'intéressent aux mécanismes du protocole MAC-hs. Les mécanismes du protocole MAC-hs sont liés avec la procédure de l'adaptation rapide du lien radio à la couche physique. Par rapport au contexte traditionnel du CDMA, l'adaptation rapide du lien radio de l'HSDPA n'est pas faite par l'intermédiaire d'un contrôle de puissance. Au lieu de cela, une palette de schémas de modulation-codage (MCS, *Modulation Coding Scheme*) est introduite et le choix du schéma MCS est continuellement ajusté selon les conditions radio courantes. Dans le paragraphe 5.4, nous proposons d'enrichir la procédure d'adaptation de la liaison de l'HSDPA en introduisant un possible fractionnement des blocs lors d'une retransmission. L'impact de cet enrichissement aussi bien sur l'administration du processus HARQ que sur la numérotation spécifique utilisée par le protocole MAC-hs sont étudiés.

Le chapitre 6 présente nos conclusions et désigne quelques directions pour nos futurs travaux et recherches.

Abstract

The UMTS 3G system (3rd generation cellular system) is designed to offer a packet switched mode for data transfers in parallel to a circuit switched mode. This feature was planned since the beginning of 3G studies, contrary to 2G ones (such as the GSM system).

To provide reliable data transfers, protocols implementing ARQ (Automatic Repeat reQuest) techniques are needed. In UMTS, up to three ARQ protocols can simultaneously be active at different levels of the protocol architecture. In this thesis, we focus on RLC (Radio Link Control), TCP (Transmission Control Protocol) and MAC-hs (Medium Access Control-high speed) protocols. The stacking of ARQ mechanisms represents a risk of interactions between them. We study studies some of these interactions.

Reliable data transfers on the UMTS radio interface are provided by the RLC protocol, precisely through RLC entities that are configured to operate in an acknowledged mode (AM). The RLC protocol is highly adaptable and specific primitives allow the tuning of the protocol for different QoS requirements. We study different aspects of the RLC mechanisms such as a SDU discard function.

Number of UMTS services are expected to be close to Internet applications such as, web-browsing, e-mail, file transfer, etc. In Internet, these services are provided via the traditional and imperative TCP/IP stack. Protocol mechanisms of TCP are compared with RLC ones. Only a part of TCP extensions for wireless networks consider a radio link with an ARQ protocol at layer 2. We analyze the TCP behavior over RLC and we discuss the usefulness of some of the proposed TCP extensions when used in the UMTS environment.

The introduction of RLC below the TCP/IP stack is useful for the overall performance; however RLC parameters have to be carefully set up. High flexibility of RLC leads to a long list of adjustable RLC parameters. One of significant parameters is the buffer size of the considered RLC entity. Impact of different TCP protocol soft states on the RLC buffer occupancy has to be considered. We investigate how the RLC buffer occupancy changes for different values of wired RTT (Round Trip Time) and BlER (Block Error Rate) on the radio interface. In addition, we analyze the impact of different RLC buffer sizes on the TCP performance. To overcome some drawbacks of TCP/RLC stacking, we propose to introduce between RLC and TCP a specific mechanism called Early Error Notification (EEN). The EEN mechanism is based on the RLC SDU discard function and on a capability of exchanging control messages between TCP and RLC entities. The EEN mechanism is proposed for uplink data transfers. A possibility of using EEN for downlink data transfers is also discussed.

Release 5 of UMTS specifications introduces for downlink data services a new mode called HSDPA (High Speed Downlink Packet Access). When updating the UMTS radio interface with HSDPA, a hybrid ARQ mechanism is introduced at the physical and MAC-hs layers. The ARQ mechanism of HSDPA, which is based on the Stop&Wait method and a specific numbering, is analyzed. The allocation mode of HSDPA is very flexible. However, services such as streaming services do not need such dynamic allocation of HSDPA resources. We investigate impact of periodic allocation on the dynamic allocation mode. In comparison with the traditional CDMA context (IS95, WCDMA), the link adaptation in HSDPA is not provided through a fast power control. Instead of it, modulation and coding schemes are continuously adjusted according to currently known radio channel conditions. In our work, we propose to enrich the HSDPA link adaptation scheme by introducing fragmentation of blocks, if necessary, during their retransmission attempts. We study impact of this enrichment on the MAC-hs protocol.

Compared to 2G systems, UMTS reinforces the radio link with new technologies and features such as more elaborated ARQ protocols. However, this reinforcement goes with a risk of accumulation of mechanisms. Harmonization of these mechanisms are challenges for the next generation of cellular systems (beyond 3G, 4G, …).

Contents

Chapter 1
Introduction of ARQ protocols in UMTS

1.1 Introduction

A boom in cellular networks and Internet characterizes the last decade. Rapid advances in these two areas resulted in increasing demands of mobile users for Internet oriented applications. Thus, the provision of data packet transfer over the radio interface becomes more and more important for mobile operators. This trend has been taken into account while planning 3rd generation cellular systems (3G).

Compared to 2G, 3G systems are designed to offer a packet-switched mode for data transfer in parallel to a circuit switched mode since the beginning of 3G studies. This is one of significant features of 3G systems, besides others. In our work, we consider the Europe version of 3G that is denoted as Universal Mobile Telecommunication System (UMTS).

To provide reliable data transfers, protocols implementing ARQ (Automatic Repeat reQuest) techniques are needed. This thesis studies ARQ protocols that can be found at the global UMTS protocol architecture: RLC (Radio Link Control), TCP (Transmission Control Protocol) and MAC-hs (Medium Access Control-high speed). Compared to 2G systems, the UMTS ARQ mechanisms are more elaborated and they include new interesting features. It is worthwhile to analyze the corresponding ARQ protocols and their relation with the ARQ of TCP since number of UMTS services is expected to be provided via the Internet TCP/IP stack. Our work focuses on the flexibility of the protocols and interactions between them.

This introductory chapter familiarizes the reader with some UMTS innovations that are exploited in following chapters. Section 1.2 presents the architecture of UMTS radio access network. In section 1.3, we detail ARQ protocols in the global UMTS architecture. Different types of UMTS QoS classes are briefly described in section 1.4. Section 1.5 presents the context of the thesis and gives overview of the dissertation.

1.2 Architecture of UMTS radio access network

Evolution of GSM towards UMTS

One of the most successful and widely used a 2G system in Europe, as well as in other parts of the world, represents the GSM system (Global System for Mobile communication, see e.g., [LG99]). The first GSM network was launched in the first half of 1990s. By 2001, the GSM system represented 70% of the world's wireless market.

The main objective of GSM was to provide mobile users with voice telephony and international roaming services. Data rates in GSM were limited. The original GSM system was just offering a 9,6 kbit/s user data rate; later on, this rate was increased to 14,4 kbit/s. A growing interest of mobile users about data oriented services pushed mobile operators to introduce several new advanced technologies into the GSM architecture; such as HSCSD (High Speed Circuit Switched Data), GPRS (General Packet Radio Service) or EDGE (Enhanced Data Rates for GSM Evolution). These enhancements gradually allowed mobile operators to increase data rates from 14,4 kbit/s up to 384 kbit/s in EDGE ([ST01]). This data rate corresponds to the expected data rate in UMTS for wide area; it is assumed up to 2 Mbit/s data rates for micro-cells and indoor UMTS environments.

Major features of UMTS

In comparison with 2G systems, UMTS provides mobile users with several new features. We have already mentioned the introduction of the packet-switched mode at the initial studies of UMTS. Furthermore, the UMTS system offers a mobile user to establish and manage several radio connections (or radio bearers) at the same time. Thus, a mobile user can simultaneously make use several different services. For example, a user can rapidly download a file from Internet while running a video teleconference.

Another important UMTS feature is a high adaptability of radio bearers. Properties of a radio bearer can be negotiated and configured according to the service that the given radio bearer supports. UMTS will provide users with a wide range of applications that have different QoS (Quality of Services) requirements; different types of UMTS QoS classes are discussed in section 1.4. Since it is not possible nowadays to predict the nature and use of many of future applications, it is neither reasonable to optimize the UMTS architecture for only set of applications. Therefore, UMTS specifications are conceived to be generic to allow good support for existing applications and to facilitate the integration of new applications. The UMTS standardization body is called Third Generation Partnership Project (3GPP). So far, several releases of UMTS (or 3GPP) specifications have been published: Release 99, Release 4 and Release 5. Our studies are based on Release 5, which appeared in 2002.

A part of UMTS services are expected to be Internet applications such as, web-browsing, e-mail, file transfer, etc. In Internet, these services are provided via the traditional and imperative TCP/IP stack (Transmission Control Protocol/Internet Protocol). Thus, in addition to the UMTS ARQ technique at the 2^{nd} layer 2 (RLC) and/or at the $1^{st}/2^{nd}$ layer (MAC-hs), we find at the global UMTS protocol architecture as well the ARQ of TCP at the 4^{th} layer (transport layer). Before detailing these ARQ techniques, we will describe the UMTS radio access network (UTRAN). The description mainly focuses on the protocol layer architecture of this network.

1.2.1 Components and features of UTRAN

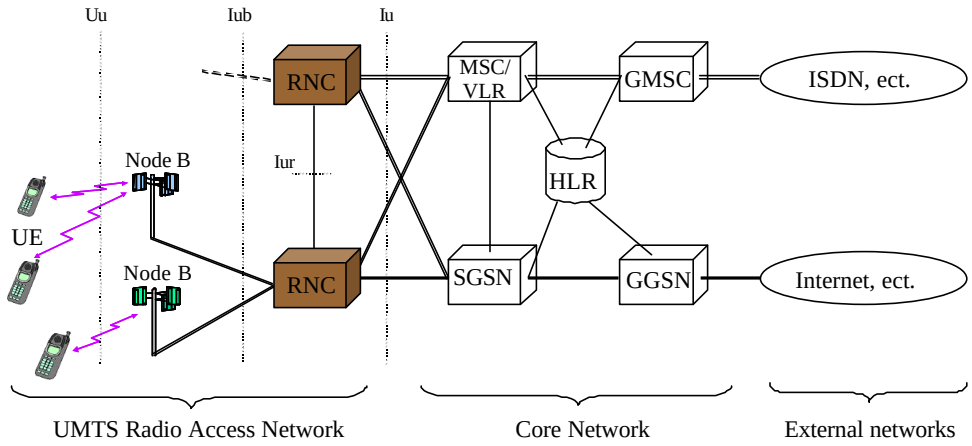
The UMTS architecture (see e.g., [HT00], [ST01] or [L01]) is decomposed into two major parts: Core Network or CN and UMTS Terrestrial Radio Access Network or UTRAN (figure 1). This thesis is interested in the radio access network.

The UTRAN consists of one or more Radio Networks Sub-system (RNS). A RNS is further composed of one Radio Network Controller entity (RNC) and of one or more entities called Node Bs. A RNC entity (respectively Node B) corresponds to the Base Station Controller entity or BSC (respectively Base Transceiver Station or BTS) in the GSM network.

A RNC entity controls functions related to the UMTS radio interface such as admission control, radio resource control, power control, etc. Studies concerning these issues have been subject of many works and can be found for example in [P01], [K01], [M00a] or [N01]. In addition, the RNC entity manages protocol exchanges on Iu, Iur, and Iub interfaces (see figure 1).

A Node B performs functions relevant to the physical layer processing; e.g., coding and interleaving, spreading/dispreading, modulation/demodulation, rate adaptation, physical measurements, etc. This network entity also performs some basic radio resources management operations such as softer handover or close loop power control.

The Time Division Multiple Access (TDMA) technique utilized in the GSM/GPRS system is replaced in UMTS by Wideband Code Division Multiple Access (WCDMA) technique. User information bits are spread over a 5 MHz wide bandwidth by multiplying the user data with quasi-random sequence bits (chips) that is derived from spreading codes.



UE	User Equipment	MSC	Mobile Switch Centre	GMSC	Gateway MSC
RNC	Radio Network Control	VLR	Visitor Location Register	SGSN	Serving GPRS Support Node
		HLR	Home Location Register	GGSN	Gateway GPRS Support Node

Figure 1. Network components of UMTS architecture ([ST01]).

Both Frequency Division Duplex (FDD) and Time Division Duplex (TDD) modes are supported by the WCDMA technique. The FDD mode uses different 5 MHz frequency bands for the downlink transmission (i.e. from a Node B to a UE entity; we also use a label $\downarrow$ for downlink) and for the uplink transmission (i.e. from a UE to a Node B; $\uparrow$). In a case of the TDD mode, the same frequency bandwidth is used for both directions (downlink, uplink). Our studies consider just the FDD mode.

1.2.2 Protocol architecture of UTRAN

The UTRAN layer architecture (see e.g., [3G25301] or [HT00]) consists of several layers and sub-layers (figure 2); in the rest of the thesis, we employ the term layer to mean as layer as sub-layer:

1. Physical layer (Phy)
2. Data link layer
   a. Medium Access Control (MAC)
   b. Radio Link Control (RLC)
   c. Packet Data Convergence Protocol (PDCP)
   d. Broadcast /Multicast Control (BMC)
3. Network layer
   a. Radio Resource Control (RRC)

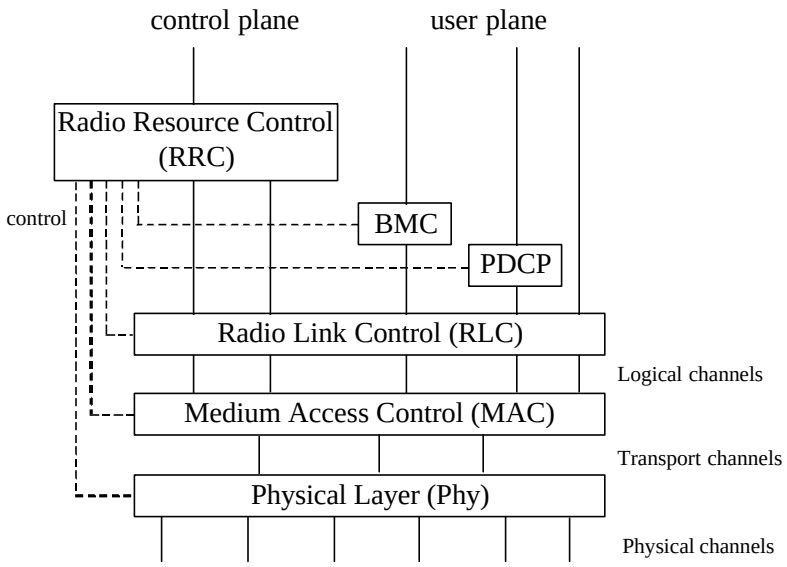


Figure 2. The UMTS radio protocol stack in a UE respectively in an RNC entity ([HT00]).

As in other telecommunication networks, the UTRAN architecture distinguishes two planes: user and control. The user plane handles user messages, whereas the control plane manages control (or signaling) messages.

Higher layer, such as Call Control (CC), Mobility Management (MM) or Session Management (SM) are transparent to the radio access network and they are not discussed in our next work.

Physical layer

The physical layer ([3G25212], [3G25302]), located in a Node B, supports the WCDMA technique. The layer offers services to the above MAC layer via radio transport channels. A transport channel is characterized by the size of transported blocks and by physical parameters. Transport channels are unidirectional.

A basic radio frame period of 10 ms (corresponding to 15 slots) is considered at the physical level. A radio transport channel can use a small multiple of the basic radio frame period duration to transmit a data unit of the MAC layer (transport block). The corresponding period is called Transmission Time Interval (TTI), where TTI $\in$ (10, 20, 40, or 80 ms). Small values of TTI are used for real time services (e.g., voice), whereas large values of TTI are configured for non-real time services (e.g., file transfers).

Medium Access Control

The MAC layer ([3G25321]) offers services to the RLC layer via logical channels that are classified into control and traffic channels. Logical control channels (respectively traffic channels) convey information originating from the control plane (respectively from the user plane). Parameters associated with a logical channel depend on the type of supported service; e.g., voice, video, signaling messages, etc.

One of MAC tasks is to schedule RLC blocks from different logical channels (i.e. from different RLC entities) according to physical constraints and QoS requirements. Furthermore, MAC is responsible for mapping data of logical channel(s) onto the appropriate transport channel(s) and vice versa. This means that the layer multiplexes (de-multiplexes) data to (from) transport channels.

The MAC is constituted of several MAC entities; the initial release of 3GPP (Release 99) specified three types of MAC entities. Each MAC entity is designated to handle specific transport channel(s):

(i) MAC-b for broadcast channel,

(ii) MAC-d for dedicated channels,

(iii) MAC-c/sh for paging, access, shared channels and common packet channel.

To enhance the downlink packet data performance in UMTS, Release 5 of 3GPP introduced a new mode called High Speed Downlink Packet Access (HSDPA). With this mode, a forth type of MAC entity, called MAC-hs (MAC-high speed), was specified at the MAC layer. The MAC-hs entity handles a new transport channel of HSDPA called HS-DSCH (for more see section 1.2.5). To the HSDPA concept is dedicated chapter 5.

Radio Link Control

The RLC layer ([3G25322]) mainly provides segmentation of higher layer units into units (blocks) that are more suitable for transmission over the air interface. The receiver side then resembles radio blocks into original higher layer units and delivers them to the upper layer. If configured, RLC ensures flow control and reliable data transfer between the UE entity and the RNC entity. Retransmission scheme of RLC is based on the ARQ technique type Selective Repeat. The RLC layer is discussed in more details in chapter 2.

Packet Data Convergence Protocol

The PDCP layer ([3G25323]) just exists in the user plane. The role of PDCP is to provide an header compression (respectively decompression) of IP packets, e.g., the compression of the 40 bytes of TCP/IP headers. The header compression allows utilizing scarce radio resources more effectively when transmitting IP packets over the radio interface. The PDCP compression stems from a fact that few TCP/IP header fields changing from one IP packet to another and the rest of header fields remain more or less the same.

The PDCP standard specifies several compression algorithms that can be utilized. Performance studies of different header compression schemes over wireless links (respectively over the UMTS radio link) can be found in [CA99] (respectively in [RG01]).

Broadcast/Multicast Control

Likewise PDCP, the BMC layer ([3G25324]) only handles data originating from the user plane. The BMC controls transmissions of Cell Broadcast Center messages over the UMTS radio interface.

1.2.3 Radio Resource Control (RRC)

The RRC layer (see e.g., [3G25331], [JC01], [TC02]) is the most complex layer in the UMTS radio protocol stack. The RRC can be seen as a “chief” layer at the UTRAN architecture.

The principal function of RRC is management of connection(s) between the UE and RNC entity. Layer signaling messages of MM, CM, SM, etc. are encapsulated into RRC messages before transferring them over the UMTS radio interface.

The RRC establishes, modifies and releases protocol entities of the lower layers (Phy, MAC, RLC, etc.). This layer management is provided through control interfaces that are specified between RRC and every lower layer (figure 2). For example, RRC configures MAC transport formats, mode of an RLC entity, PDCP compression algorithm, etc.

Through the control interfaces, RRC also commands lower layers to perform certain types of measurements and to get measurement results back. For example, RRC may ask a lower layer to measure signal to interference ratio in a UE or to measure the Round Trip Time (RTT) between UE and RNC, etc. If an unexpected event occurs at any of the lower layer, the event is reported through the control interface to RRC, which takes the next steps. The RRC/RLC control interface is depicted in more details in section 2.4.4.

Figure 3 shows RRC states and transitions among them: (a) Paging, (b) Broadcast, (c) Dedicated control and (d) Active.

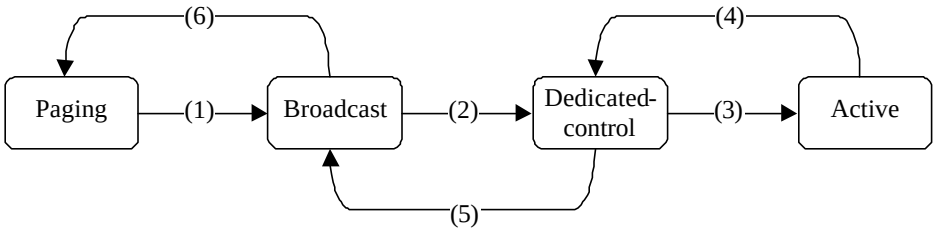


Figure 3. Different RRC states and transitions among them ([TC02]).

Paging: is a low-power standby state in which a UE periodically listens to a paging channel. In this state, no dedicated physical channel is allocated to the UE. Effectively, no uplink/downlink data transfers are possible in this state.

Broadcast: is a broadcast-only state in which a UE monitors common channels for downlink transfers and uses the random access channel for small uplink data transfers. Typically, the data transfers are limited to control information. The UE continuously monitors the common channel and hence, there is considerable power consumption.

Dedicated-control: is a dedicated state where a UE has acquired a dedicated physical channel. In this state, many of the physical layer maintenance processes are active. Typically, this state is limited to control or low rate data transfers. In addition, in this state the control overhead requires a noticeable fraction of system resources. Therefore, number of UEs occupying this state is limited.

Active: is an active state where a UE has been assigned significant radio resources for data transfers. Transitions from this state and the dedicated-control state may be made instantaneously depending on the specific network and mobile implementations.

1.2.4 Radio bearers

One of important features of UMTS is its possibility to dynamically adapt characteristics of a radio bearer according to supported service. The notation *radio bearer* can be seen as an association of functions that are located at different levels of the radio protocol stack. Characteristics (QoS characteristics) of a radio bearer are negotiated when the radio bearer is established. Within an ongoing connection, the initial proprieties of the radio bearer can be renegotiated and can be modified. The establishment, reconfiguration and release of a radio bearer are controlled by RRC.

The maximum number of established radio bearer (signaling and non-signaling) between the UE and the RNC entity is shown in the following table 1 ([3G25331]).

Maximum number of radio bearers per UE	32
- maximum number of signalling radio bearers	8
- maximum number of non-signalling radio bearers	27

Table 1. A maximum number of established radio bearers between the UE and RNC entity.

Examples of radio bearer configurations at the UE side can be found in [3G34108]. Furthermore, this reference describes number of testing scenarios.

1.2.5 Transport channels for packet transfers

To carry packet data traffics over the radio interface, UMTS disposes of several different transport channels that can be applied. These channels can be divided into the three groups:

(i) Random Access Channel (RACH, ↑), Forward Access Channel (FACH, ↓), and Common Packet Channel (CPCH, ↑);
(ii) dedicated channels (↑, ↓)
(iii) shared channels (↓)

The first two channels of the first group, RACH and FACH, are appropriate for transmission of small data amounts, such as a single request of web-page ([HT00]). The CPCH is suitable for small and medium data amounts. In case of large data amounts, the first group of transport channels suffers from their poor radio performance and other channels should be employed ([HT00]). Contrary to RACH and FACH, CPCH is fast power controlled. None of these three channels supports soft handovers.

Channels in the second group, as the name indicates, are dedicated to the UE. Every user can have established several dedicated transport channels. These channels are suitable for medium and large data transfers. Dedicated channels supports fast power control and soft hand-

overs. Since dedicated channels have to be set up at the beginning of transmissions (which takes certain time), the access time to them is longer than that of the channels in the first and third group.

To simplify our analysis concerning interactions between TCP and RLC protocol in chapter 4, these studies assume a TCP data transfer over the dedicated transport channel.

Shared channels dynamically share common radio resources among several users. The resources are assigned according to implemented packet scheduler; more about packet scheduling in UMTS can be found for example in [E03]. There are specified two shared transport channels. Both of them can transport as user data as signaling messages:

- Downlink Shared Channel (DSCH), introduced in Release 99 (1999),
- High Speed DSCH (HS-DSCH), introduced in Release 5 (2002).

The DSCH channel supports fast power control and variable spreading factor. The spreading factor of DSCH can vary from frame to frame, i.e. every 10 ms. To every UE, which uses DSCH, is assigned an associated dedicated downlink channel that carries relating DSCH information (the transport format, power control, pilot bits, etc).

In a case of the HS-DSCH channel, the link adaptation is not provided through the fast power control and variable spreading factor. Instead, the link adaptation is provided by continuously adjusting Modulation and Coding Schemes (MCSs) according to current radio channel conditions. The packet scheduling is moved from the RNC entity closer to the radio interface, i. e. to Node B. In addition, the allocation period is decreased from 10 ms to 2 ms (or three slot). Information about which UE(s) is scheduled in a given TTI (and as well other necessary information concerning HS-DSCH transmission) is transported through a downlink-signaling channel. We discuss more HSDPA signaling channels in section 5.2. A comparison of DSCH and HS-DSCH channels is provided in table 2.

	Downlink Shared Ch (Release 99, 1999)	High Speed Downlink Shared Ch (Release 5, 2002)
Variable spreading factor	Yes	No
Fast power control	Yes	No
Adaptive modulation and coding	No	Yes
HARQ	No	Yes
Transmission Time Interval (TTI)	10 ms	2 ms
Location of the packet scheduler	RNC	Node B

Table 2. A comparison of features ofDSCH and HS-DSCH (inspired by [H02]).

1.3 ARQ techniques in UMTS

The UMTS system reinforces the radio link with more elaborated ARQ mechanisms compared to 2G. The corresponding ARQ protocols can be found at the three different levels of the global UMTS protocol architecture: the 1st/2nd levels (physical/data link layers), the 2nd level (data link layer) and at the 4th level (transport layer). The ARQ protocols operate between User Equipment (i.e. a mobile) and different network elements such as server (or proxy server), RNC entity and Node B.

TCP

The highest operating reliable mechanism is the ARQ of TCP at the transport layer. The ARQ of TCP (type Go-Back-N) ensures end-to-end reliable data transfers, i.e. between the mobile and server or proxy server. Proxy servers are widely used in nowadays Internet. These servers allow (fix, mobile) users to get remote web-pages locally from the proxy cache instead of downloading them from their original distant sites (for more about proxy see e.g., [RFC2616]).

Notice that since the introduction of the TCP protocol, features of links in wired networks have largely evolved. Due to the vast utilization of optic fibers in wired networks, wired links can presently be assumed to be reliable. The principal source of losses in these networks is overflows in routers (i.e. router congestions). The TCP behavior upon detecting congestions is discussed in chapter 3.

RLC

Reliable data transfers on the radio interface, i.e. between the UE and RNC entity, are realized through the RLC protocol; precisely through RLC entities that are configured to operate in an acknowledged mode (AM). Different aspects of AM RLC entities are investigated in chapter 2 and chapter 3.

MAC-hs

When updating the UMTS radio interface with the HSDPA mode, a hybrid ARQ mechanism is introduced at the physical and MAC-hs layers (1st/2nd levels). The HARQ of MAC-hs operates between the UE and the Node B. Being closer to the air interface the MAC-hs protocol deals with erroneous data much faster than RLC does. The MAC-hs Round Trip Time (RTT) is about a dozen of ms contrary to several tens of ms in a case of RLC RTT. However, notice that the HSDPA mode was introduced for the downlink (HS-Downlink Packet Access). Therefore, contrary to the ARQ of RLC, the HARQ of MAC-hs is just applicable for downlink data transfers.

The HARQ of MAC-hs is based on the simplest ARQ technique of type Stop&Wait (S&W). Several S&W instances (or process) can simultaneously be activated per a radio bearer. Due to the multi-instance ARQ together with a specific numbering, the HARQ of MAC-hs behaves as if a Selective Repeat method would be applied. We will analyze this feature in more details in section 5.2 where we study the HSDPA mode.

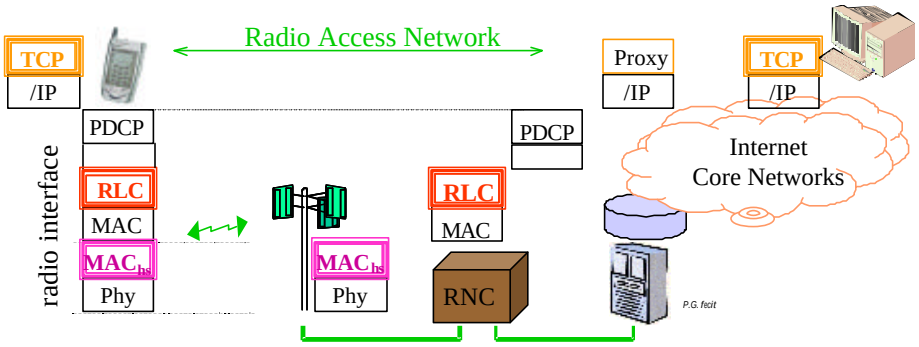


Figure 4. ARQ mechanisms in the global layer architecture of UMTS.

1.4 UMTS QoS Classes

Every UMTS service has specific requirements on wireless delay, bit error rate and eventually data rate. These attributes are negotiated when establishing radio bearers.

UMTS services are grouped into the UMTS QoS classes or traffic classes according to their characteristics. Reference [3G23107] specifies four types of UMTS QoS classes (table 3): (i) conversational, (ii) streaming, (iii) interactive and (iv) background. The main distinguishable factor among these classes is how sensitive they are to delay.

Conversational and Streaming classes

The conversational class is the most delay sensitive UMTS QoS class. The most known service of this class is the voice. An insufficiently low end-to-end delay results in unacceptably quality from the user point of view. Since this class has strict requirements on the end-to-end delay, the ARQ mechanisms are not used; received data which delay exceeds a certain threshold is useless for the receiver, even though the data would be error free. Studies of the human perception shown that the end-to-end delay for voice services has to be less than 400 ms ([3G22105]). The RLC entity is configured to operate in Transparent Mode (we discuss this RLC mode in section 2.2.1). Furthermore, the data processing delay at the UMTS radio protocol stack should be minimized as much as possible, i.e. TTI (Transmission Time Interval) is set to 10 ms.

Contrary to the conversational class, the streaming class is less strict to the delay. The quality of service is again determined by the human perception since the receiver is mostly a human. The interactive class represents services such as audio or video streaming. These services usually employ buffering at the receiver's side in order to smooth jitter of arrived data flows. This way, the reproduced flow is slightly put backed compare to the real time arrived flow. As shown in [M02], due to short MAC-hs RTT, the HARQ of MAC-hs with a restricted number of retransmission attempts is possible applied for streaming services. Compared to MAC-hs RTT, the RLC RTT is longer. The use of ARQ of RLC would introduce much more important radio delay than MAC-hs does. Therefore, the ARQ of RLC should be avoided for streaming services. We will return to streaming services when studying the HSDPA allocation mode (section 5.3).

Interactive and background classes

Interactive and background classes represent traditional Internet applications (e.g., web-browsing, telnet, etc.). Requirements on delay of these classes are looser than the previous ones. On the other hand, they require preserving data integrity. Therefore, radio bearers supporting these classes are configured to use either the ARQ of RLC or the HARQ of MAC-hs eventually both of them. The ARQ of RLC above the HARQ of MAC-hs could be used for example as "a safety against errors" that can pass through the MAC-hs layer.

Traffic class	Conversational	Streaming	Interactive	Background
Characteristics	Preserve time relation (variation) between information entities of the stream Conversational pattern (stringent and low delay)	Preserve time relation (variation) between information entities of the stream	Request response pattern Preserve data integrity	Destination is not expecting data within a certain time Preserve data integrity
Mode	Circuit	Circuit, Packet	Packet	Packet
ARQ		MAC-hs	MAC-hs, RLC TCP	MAC-hs, RLC TCP
Example	voice	audio, video streaming	web-browsing	SMS, MMS

Table 3. UMTS QoS classes (inspired by [3G23107]).

Interactive class corresponds to applications with the classical request response pattern such as web-browsing. In a case of web-browsing, the user expects a response in “reasonable” short time. The last class, background class, represents the less delay sensitive applications such as SMS (Short Message Service) or MMS (Multimedia Message Service). Since these applications do not require immediate action, they can run in background of other tasks that have higher priority. The delivery delay may range from few seconds up to few minutes.

1.5 Context and structure of the thesis

This thesis studies new features of the UMTS ARQ mechanisms and investigates possibilities of their enrichments. The accumulation of ARQ mechanisms at different levels of the UMTS global architecture goes with interactions between the ARQ protocols. Our work points some of TCP/RLC interactions and proposes a remedy to an adverse one.

A part of the thesis was realized in the framework of a RNRT project called MPRIM (Modules PRotocolaires Interchangeables pour Mobiles multimedia, [Mprim]). The project was realized in cooperation of Alcatel, ENST, I2E, SFR and Thomson CSF Communications. The project has investigated a possibility of dynamic reconfigurations of the lower three layers of UMTS. Due to the high complexity of this issue, we have focused our interest on a parameter setting of the RLC layer; precisely on a parameter setting of a RLC entity operating in an acknowledged mode (AM).

1.5.1 Studied features and related works

Parameters such as RLC timers, the RLC transmission window or the maximum number of transmission attempts per RLC block are analyzed in [ZL02], [XC02a] and [LV01]. One of significant RLC parameters is also the RLC buffer size. Impact of different type of MAC scheduling algorithms (respectively different TCP receiving window sizes) on the RLC buffer size is analyzed in [MH03] (respectively in [LV01]). Our work investigates how different TCP protocol soft states affect the RLC buffer occupancy. The buffer occupancy by return affects the TCP RTT (Round Trip Time).

In context of TCP data transfers over the radio interface, two different cases have to be distinguished: a TCP uplink and a TCP downlink transmission. In the first case, the RLC buffer (located in the UE) occupancy is determined by the operating system of UE and its internal flow control between RLC and TCP besides the TCP protocol soft states. The internal flow control is not standardized; its realization depends on the used operation system of UE. Our work considers the downlink case, where the internal flow between RLC and TCP control does not intervene and the RLC buffer (located in the RNC) occupancy is proportional to the TCP transmission window.

A classical problem of TCP on the radio interface represents the misinterpretation of wireless and congestions losses. This problem is handled by different TCP extensions such as Indirect-TCP ([BB95]), Snoop ([BS96]), Wireless-TCP ([RM98]), etc. However, only a part of the TCP extensions for wireless networks consider a radio link with an ARQ protocol at layer 2. In UMTS, RLC avoids wireless losses due to its ARQ mechanism, unless a specific RLC function called SDU discard is active. The RLC SDU discard mechanism allows the AM RLC entity to delimitate a number of retransmission attempts per RLC blocks. High number of retransmission attempts may result in TCP spurious timeouts. Two TCP extensions dealing with TCP spurious timeouts are called Eifel ([LK00]) and EBSN (Explicit Bad State Notification, [BK96]). Neither Eifel nor EBSN covers the RLC SDU discard issue. We propose to consider the RLC SDU discard function and to introduce between RLC and TCP a specific mechanism called Early Error Notification (EEN). The EEN mechanism is proposed for TCP uplink transmissions and its use for downlink transfers is also discussed.

The HSDPA mode introduces a third hybrid ARQ mechanism at the physical and MAC-hs layers. Besides the HARQ mechanism, HSDPA brings several other features such fast scheduling or fast link adaptation through MCSs (Modulation Coding Scheme). Majority of the HSDPA studies cover HSDPA link adaptation, performance of Error Correction Codes or HSDPA scheduling issues ([NA02], [DK02a], [KF02b]). HSDPA features make possible to it for streaming services; the HSDPA performance for streaming services is analyzed in [M02]. Nevertheless, streaming services do not need such dynamic allocation that HSDPA offers. This type of services generates regularly outgoing data and such data flows postulate a periodic allocation. Our work discusses impact of periodic allocation on the dynamic allocation mode and on the HSDPA signaling.

For the time being, few studies concern MAC-hs protocol features; e.g. impact of MAC-hs window size on delay is studied in [MR03]. We approach this domain in two steps. In the first step, we propose to enrich the HSDPA link adaptation by modifying the MAC-hs PDU size during retransmissions attempts. In the second step, we study the effect of this enrichment on the HARQ process management and on the specific MAC-hs numbering.

1.5.2 Structure of the thesis

Chapter 2

The next chapter is dedicated to the RLC protocol (Radio Link Control) of UMTS. Section 2.2 describes basic fact about RLC (RLC modes, type of RLC blocks, etc). The RLC QoS management is performed through a specific function called SDU discard that is explained in section 2.3.

Interlayer communication between RLC and its adjacent layers is investigated in section 2.4. In section 2.4.1, we briefly remind different types of interlayer primitives and their use.

Section 2.4.2 (respectively section 2.4.3) details layer interface between RLC and the adjacent upper layer (respectively lower layer). The control interface between the RLC and RRC layers is analyzed in section 2.4.4.

Some differences between RLC of (E)GPRS and RLC of UMTS are mentioned in section 2.5. Chapter 2 terminates remarks concerning development and implementation of protocols (section 2.6).

Chapter 3

In chapter 3, we turn our attention to the transport layer and we study TCP (Transmission Control Protocol) in cellular networks with a focus on the UMTS radio interface. Section 3.2 remains some basic facts about TCP (protocol soft states, behavior when detecting a loss, etc.). Section 3.3 briefly presents other reliable transport protocol called Stream Control Transmission Protocol. In section 3.4, protocol mechanisms of TCP are compared with the RLC ones. Section 3.5 analyzes the TCP behavior over RLC and we discuss usefulness of some proposed TCP extensions when used in the UMTS environment.

Chapter 4

Chapter 4 studies protocol interactions between TCP and RLC (of UMTS). This chapter is decomposed into three parts. The first part briefly describes two mechanisms that in a restricted way allow informing the TCP transmitter about specific events occurring in the connection path; these are Internet Control Message Protocol (section 4.2.1) and Explicit Congestion Notification (section 4.2.2).

The second part of chapter 4 (section 4.3) deals with the RLC buffer occupancy; we focus on TCP downlink transmissions. Section 4.3.1 describes our simulation environments. Section 4.3.2 investigates two of factors that influence the RLC buffer occupancy: (i) wired Round Trip Time (RTT_{wired}) and (ii) Block Error Rate (BlER) on the radio interface. In addition, impact of different RLC buffer sizes on the TCP performance is analyzed. The RLC buffer investigations are concluded in section 4.3.3.

The third part of chapter 4 (section 4.4) discusses a mechanism EEN (Early Error Notification) that we propose to introduce between TCP and RLC. Section 4.4.1 details the proposed mechanism for a TCP uplink data transmission. The simulation experiments and results are described in section 4.4.2. Contrary to the uplink case, in the downlink case a TCP transmitting entity and a RLC entity are each located in different equipments. Thus, EEN cannot be directly applied in downlink as described in section 4.4.1. A possibility of using EEN for downlink transmission is discussed in section 4.4.3. The EEN studies are concluded in section 4.4.4.

Chapter 5

Chapter 5 covers the HSDPA mode (High Speed Downlink Packet Access) for UMTS. Section 5.2 describes the HSDPA context (new channels, layer structure, protocol mechanisms, etc).

Section 5.3 is dedicated to the HSDPA allocation mode. Basic facts about the dynamic allocation mode are provided in section 5.3.1. The periodic allocation and its features are described in section 5.3.2. Simulation experiments concerning the dynamic and periodic allocation are depicted in section 5.3.3. Section 5.3.4 concludes studies about the allocation of radio resources in HSDPA.

The last part of chapter 5 (section 5.4) discusses the HSDPA link adaptation procedure and a possible enrichment of this procedure. The standard HSDPA link adaptation procedure is described in section 5.4.1. Section 5.4.2 provides more details about the enrichment that we propose introduce in the link adaptation procedure and we study its impact on Phy/MAC-hs layer. Section 5.4.3 depicts simulation environment and parameter setting. Performance of the standard and proposed adaptation schemes is compared in section 5.4.4. The HSDPA link adaptation analyses are concluded in section 5.4.5.

Chapter 6

In chapter 6, we provide main conclusions and we outline some possible future research directions.

Chapter 2
Radio Link Control

2.1 Introduction

Several substantial differences can be observed between wired and cellular networks. Cellular networks generally dispose of a lower bandwidth for transmission (meaning also lower rates) than wired networks do. In comparison with wired networks, cellular networks have to deal with user mobility, i.e. handoffs. Furthermore, the air represents a medium with high variability (fading, interferences, etc.). The variability of medium and the user mobility results in bit error rates that are much important in cellular networks (as high as 10^{-3}) than in wired networks (less than 10^{-7}). In order to detect and rapidly deal with errors on an air interface, the radio protocol stack of cellular systems implement reliable techniques. The next section reminds the basic reliable mechanisms and provides an overview of data link protocols.

2.1.1 Reliable mechanisms

There are two principal reliable techniques: ARQ (Automatic Repeat reQuest) and FEC (Forward Error Correction). The first reliable technique (ARQ) handles errors by retransmitting data according to receiver demands, whereas the FEC technique makes use of Error Correction Code (ECC) that allows the receiver to detect and correct the errors. The FEC receiver is capable to detect and correct a data unit that contains fewer than a predetermined number of errors in the unit. Our studies focus on the ARQ technique. More about ECC can be found in [CD92].

ARQ technique

Compared to FEC, the ARQ technique requires one bi-directional connection between the transmitter and the receiver. One direction is reserved for data transmission from the transmitter to the receiver. The opposite direction is used by the receiver to transfer status reports, i.e. Ack/NAck (positive/Negative Acknowledgement). Through status reports, the receiver informs the transmitter about data (blocks) that has to be retransmitted. To unambiguously distinguish blocks, the ARQ protocol has to effectuate their numbering. A block is retransmitted until the

block is either correctly received or a predetermined number of retransmissions is reached respectively a specific timer delimitating the maximum retransmission period expires. The ARQ protocol can manage either just one block (Stop and Wait) or several blocks at a time (Go-Back-N, Selective Repeat). Notice that erroneous data at the receiver side is thrown away; the receiver does not store it in its receiver buffer.

HARQ technique

By combining the ARQ and FEC techniques, we obtain a third technique called hybrid ARQ (HARQ). The ARQ part of HARQ can be provided by any of the earlier mentioned ARQ mechanisms (Stop and Wait, Go-Back-N, Selective Repeat). Compared to the ARQ receiver, the HARQ receiver may store received erroneous data to combine it with data that is received in the following retransmission attempts. There have been specified several HARQ schemes (HARQ-I, HARQ-II, etc.); more about them can be found for example in [M74], [LC84], or [DK02b]. In our work, we are just interested about the HARQ mechanism from the view of its protocol mechanisms, i.e. about the ARQ part of HARQ.

2.1.2 Overview of data link protocols providing reliable transfers

One of the best-known data link protocols for reliable transfers is the HDLC (High-Level Data Link Control) protocol. As a matter of fact, many other data link layer protocols are based on it; for example LAPB (Link Access Protocol Balanced) for X.25, LAPD (Link Access Protocol in the D channel) for ISDN (Integrated Services Digital Network), LAPM (Link Access Protocol for Modems) for V.42 modem, etc.

The data link layer of GSM includes a protocol called Radio Link Protocol (RLP). The RLP protocol is based on a simple protocol of type LAPDm (Link Access Protocol on the Dm channel), which is a modified version of LAPD. Notice that the cdma2000 system also specifies at the data link layer the RLP protocol (RLP-cdma2000), which however differs from the GSM version. From others reliable data link protocols, we may mention Service-Specific Connection Oriented Protocol (SSCOP) of ATM (Asynchronous Transfer Mode) or Error Control (EC) protocol that is utilized in HiperLAN/2.

In the next study, we restrict our view on the GSM/GPRS and UMTS environments and on a data link protocol of type RLC (Radio Link Control). A first simple version of RLC was introduced in the GPRS system (see e.g., [LG99]). The RLC version of GPRS operates between a mobile and BSS (Base Station Sub-system). The protocol provides either a reliable or an unreliable transfer of LLC messages (Logical Link Control) over the GPRS radio interface.

A more advance version of the RLC protocol has been introduced in the UMTS system. In comparison with RLC of GPRS, RLC of UMTS is more elaborated and offers more services to

the upper layer. This chapter provides more details about protocol mechanisms. There will explicitly be mentioned when considering a RLC version of different cellular systems.

This chapter is organized as follows. The next section discusses basic facts about the protocol (protocol modes, protocol units, etc.). Section 2.3 describes a specific RLC function called SDU discard. Interlayer communication between RLC and the upper and lower adjacent layers is studied in section 2.4; there is also detailed the control interface between RLC and RRC. Some of major differences between RLC of (E)GPRS and RLC of UMTS are provided in section 2.5. Chapter 2 is concluded by remarks concerning development and implementation of protocols (section 2.6).

2.2 Basic fact about RLC of UMTS

The RLC protocol operates between the UE (User Equipment, or mobile) and RNC entities (Radio Network Controller). The eponymous RLC layer can be seen as a composition of several RLC entities or RLC instances that operate independently. An establishment, (re)configuration and release of a RLC entity are controlled by the upper layer RRC (Radio Resource Control). There are established as many RLC entities per UE as many radio bearers are established between the UE and RNC entities.

One of RLC tasks is to segment upper layer protocol units, i.e. RLC SDUs (Service Data Unit), into RLC protocol units (RLC Blocks) that are more suitable for transmissions over the radio interface. A RLC SDU is shortly denoted as a SDU in the rest of the thesis. A SDU can contain either user data (when delivering from the user plane) or signaling messages (when delivering from the control plane). There is no implementation difference between RLC entities that handle user data and entities that handle signaling messages.

2.2.1 RLC Modes

Compared to RLC of GPRS, a RLC entity of UMTS can be configured to operate in one of the following three modes:

(i) Transparent Mode (or TM),
(ii) Unacknowledged Mode (or UM),
(iii) Acknowledged Mode (or AM).

Transparent mode

This mode is the simplest mode. A TM RLC entity does not add any RLC protocol overhead to an upper layer unit (i.e. a SDU). The entity is transparent to SDUs. This type of RLC mode is used for a service that tolerate higher bit error rate; however the end-to-end delay has to be

kept low. Such type of service is for example voice, where the maximum end-to-end delay is determined by the human perception.

Unacknowledged mode

Likewise a TM RLC entity, a UM RLC entity does not guarantee correct deliveries of RLC blocks to the peer entity. Contrary to a TM entity, an UM entity adds RLC overheads to data. The header of UM RLC block consists of one byte: Sequence Number (SN, 7 bits) and Extension bit (E, for more see the acknowledged mode). Since RLC blocks are numbered, an UM entity can provide segmentation of SDUs and their reassembling at the receiver side. If possible, the entity applies a concatenation, i.e. a RLC block can contain fragments of 2 SDUs. An example of the segmentation and concatenation of SDUs is illustrated in figure 5.

If a receiving UM entity detects a missing RLC block, the entity automatically discards all RLC blocks that are part of the same SDU(s); a missing block is detected by reception of a block with SN that is out of order. By this way, reassembled SDUs are always delivered to the upper layer in order (some of SDUs can be missing). The unacknowledged mode is utilized for transfers of cell broadcast messages or for transfers of certain RRC messages.

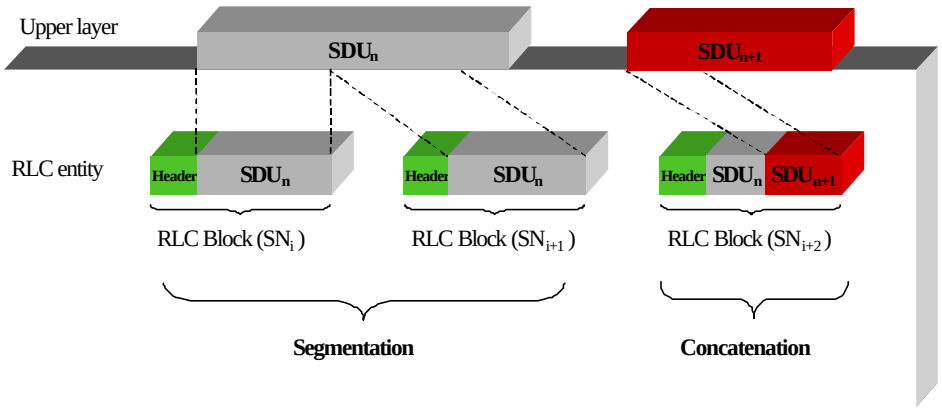


Figure 5. An example of segmentation and concatenation of RLC SDUs.

Acknowledged mode

This last mode is the most elaborated RLC mode. A RLC entity is configured to operate in AM if a reliable data transfer over the UMTS radio interface is required. The RLC protocol uses the ARQ technique of type Selective Repeat.

Compared to a UM RLC entity, an AM RLC entity can be configured to deliver reassembled SDUs to the upper layer either in sequence or out of sequence (i.e. a SDU is delivered to the upper layer as soon as is reassembled). When using TCP at the transport layer, changes of

SDUs order at the RLC level may result in the degradation of TCP performance; this is due to the TCP behavior on reordered segments (for more see section 3.2). Thus, if an AM RLC entity handles TCP segments, the entity has to deliver SDUs to the upper layer in-sequence.

An AM entity is utilized for services that tolerate higher wireless delays and require preserving data integrity (e.g., file transfer).

A TM or UM entity is unidirectional. This means that for a bi-directional connection (unreliable), two peer RLC entities need to be established; one peer controls transmission from the UE to RNC entity and the second peer controls transmission from the RNC to UE entity. An AM entity is bi-directional; one combined entity controls as transmission as reception.

2.2.2 Types of RLC blocks and their structures

There are two types of RLC blocks: (i) PDU (denoted as Data PDU in the rest of the thesis) and (ii) Status PDU. TM and UM RLC entities just manage Data PDUs.

Data and Status PDUs

A Data PDU can contain data or RRC signaling message(s), whereas a Status PDU carries RLC related control information. RLC control information can be intended either to the sender or to the receiver, i.e. as the sender as the receiver can transmit a Status PDU.

A Status PDU is composed of one or several fields called Super Fields or SuFi (figure 6). For example, a super field called ACK carries, as the name indicates, an acknowledgement. There are specified 8 types of SuFis (table 4). Some of them have a variable size that may change from Status PDU to Status PDU, whereas other SuFis have a fix size. The utilization of each SuFi will be presented through chapters 2 and 3.

Name of Super Field (SuFi)	Transmitted by	Size [bits]
NO_MORE (No More Data)	Sender or receiver	4
WINDOW (Window Size)	Receiver	16
ACK (Acknowledgment)	Receiver	16
BITMAP (Bitmap)	Receiver	28…148*
LIST (List)	Receiver	24…264*
RLIST (Relative List)	Receiver	24…80*
MRW (Move Receiving Window)	Sender	24…192*
MRW_ACK (Move Receiving Window Acknowledgement)	Receiver	20

Table 4. Name of super fields and its length; minimum …maximum size.*

The end of the super field part (i.e. the end of useful information) in a Status PDU is delimited by super fields ACK or NO_MORE. Bits following these SuFi are considered to be padding bits (Pad) and are ignored by the RLC receiver.

If an AM entity is connected to the MAC layer via one logical channel, the same size of Data PDUs and Status PDUs has to be used. A Status PDU carries in most cases a few bytes of useful information compared to a Data PDU (≈ 128, …, 640 bits, [3G34108]). To reduce the amount of useless padding bits that are transferred over the radio interface, an AM entity can be connected to the MAC layer through two logical channels. One logical channel transports Data PDUs, whereas the second channel transports Status PDUs. Thus, the Data PDU and the Status PDU can have a different size.

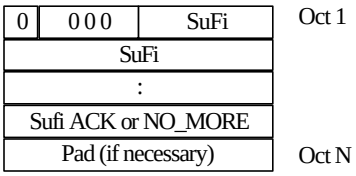


Figure 6. Structure of a Status PDU.

In a case of AM RLC entity, one RLC block can contain as data as RLC control information. As it well known, the technique, where data and control information are sent in the same protocol unit, is called piggybacking. The piggybacking control information form a RLC block called, in the RLC specification ([3G25322]), Piggybacked Status PDU.

Notwithstanding, the utilization of the RLC piggybacking mechanism is restricted. The mechanism can only be applied if a Data PDU contains the end of a SDU and the RLC entity would apply either padding or concatenation. If RLC control information cannot be postponed until the end of a SDU or if there is not enough space to insert the whole Piggybacked Status PDU into the Data PDU, a standalone Status PDU has to be employed.

Structure of Data PDU

Figure 7 shows an example of possible structures of Data PDUs that are managed by an AM entity. The first two bytes constitute the RLC header of a (AM) Data PDU. The first bit of the header, called D/C (Data/Control), specifies the type of PDU (Data/Status). The Polling bit (P) is used by the sender to demand the receiver to send a status report (i.e. Ack/NAck). More about the RLC acknowledgment policy provides section 3.3. The Header Extension (HE) flag indicates the meaning of the following byte, i.e. data or flags LI and E (see next).

If a Data PDU contains the end of a SDU, there are added flags LI and E (Length Indicator and Extension). The LI flag specifies the number of bytes between the end of the last LIE flag up to and including the byte at the end of an SDU in the RLC block. The rest of the block can be filled up either with data of the next SDU (concatenation) or with the padding bits (Pad) or with a Piggybacked Status PDU (piggybacking). If the padding or piggybacking is applied, other LIE flag is added; LI has specific values for padding and piggybacking (figure 7c, 7d).

There are used super fields NO_MORE or ACK to indicate the end of the Piggybacked Status PDU in the Data PDU. The rest of the Data PDU is filled up with Pad (figure 7d).

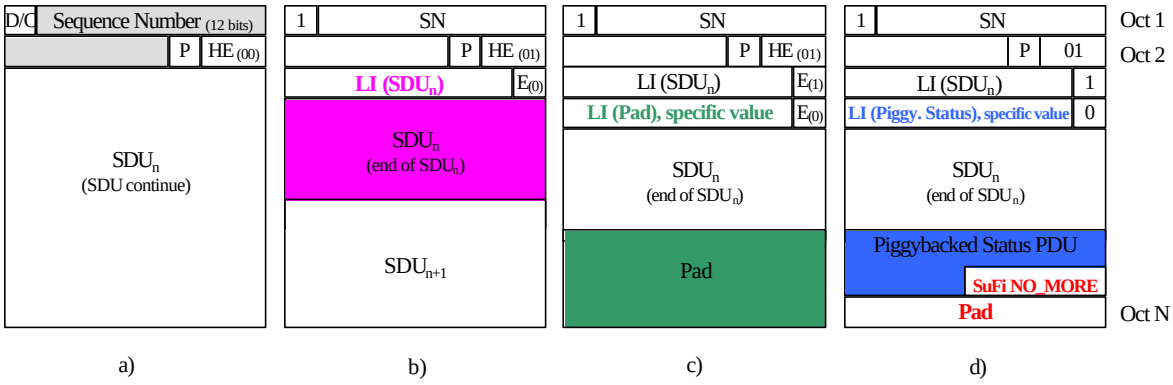


Figure 7. An example of Data PDU structures when using the acknowledged mode.

Remark: Notice that if the end of a SDU entirely fills up a Data PDU and there is not place to add the LIE flag, this flag is carried by the next transmitted Data PDU. In such case, the LI flag has a specific value. The LI flag does not designate the length of the rest SDU in the previous Data PDU since the receiver knows the Data PDU size.

2.3 Management of QoS at the RLC layer

The RLC layer manages QoS requirements concerning wireless delays and/or error rates of RLC blocks on the UMTS radio interface. The management of these two parameters is provided through a specific RLC function called SDU discard ([3G25322]).

The RLC sender utilizes the SDU discard function to discard SDUs from the sender buffer if a certain condition becomes fulfill. This condition can be represented by: (i) the attainment of the maximum sojourn time of a SDU in the sender buffer or (ii) the attainment of the maximum allowed number of retransmissions attempts for a given Data PDU. According to the followed target, an RLC entity is configured to utilize one of SDU discard modes indicated in table 5. This table also indicates the RLC mode for which the SDU discard mode is applicable. Next paragraphs will describe the SDU discard modes in more details; we focus on an AM entity.

Before analyzing SDU discard modes, notice that the SDU discard function (whatever RLC modes) is provided by managing of several RLC state variables and RLC timers. The upper bounds of state variables are delimitated by RLC protocol parameters. Values of these RLC parameters and RLC timers are signaled from the RRC layer when configuring the RLC entity. Our notations of state variables, timers and protocol parameters follow the prevalent notations used in the RLC specific ation ([3G25322]).

Name of SDU discard mode	Applicable for RLC mode
Timer based discard, with explicit signaling	Acknowledged
Timer based discard, without explicit signaling	Transparent, Unacknowledged
SDU discard after MaxDAT number of transmissions	Acknowledged
No discard after MaxDAT number of transmissions	Acknowledged

Table 5. Types of SDU discard modes and the RLC mode for which they are applicable.

Timer based discard, with explicit signaling

This type of SDU discard mode allows an AM RLC entity to delimit the maximum allowed sojourn time of a SDU in the sender buffer. This means to determine the maximum wireless delay of a SDU. Growing number of discarded SDUs increases the SDU loss rate.

The maximum allowed sojourn time of a SDU in the sender buffer is delimited trough a specific timer. This timer is called Timer_Discard. Values of the Timer_Discard timer can range from 100 ms up to 7,5 seconds ([3G25331]).

To every delivered SDU from the adjacent upper layer (e.g., PDCP) is associated and activated a new Timer_Discard timer (figure 8a). Thus, there are as many active Timer_Discard timers as many SDUs are currently stored in the sender buffer. As soon as a SDU is acknowledged to be correctly transferred over the radio interface, the SDU is removed from the RLC buffer and the associated timer is disabled. If a Timer_Discard timer expires, the corresponding SDU is discarded from the RLC buffer (figure 8b). This means that all Data PDUs that constitute the SDU (denoted as co-SDU blocks) are discarded from the RLC buffer. Notice that several Timer_Discard timers can simultaneously expire and thus several SDUs can be discarded from the RLC buffer at the same time.

Before continuing the next data transmission, the sender has to *explicitly* inform the receiver about all discarded Data PDUs. This notification is provided via a SuFi MRW (Move Receiving Window, figure 8b) carrying in a Status PDU. Through this super field, the sender informs

the receiver how to advance its receiving window to skip the discarded Data PDUs. Receipt of the SuFi MRW and the corresponding advancement of the receiving window have to confirm the sender. To do this, the SuFi MRW_ACK is sent to the sender. After receiving the confirmation super field, the sender resumes the next data transmission.

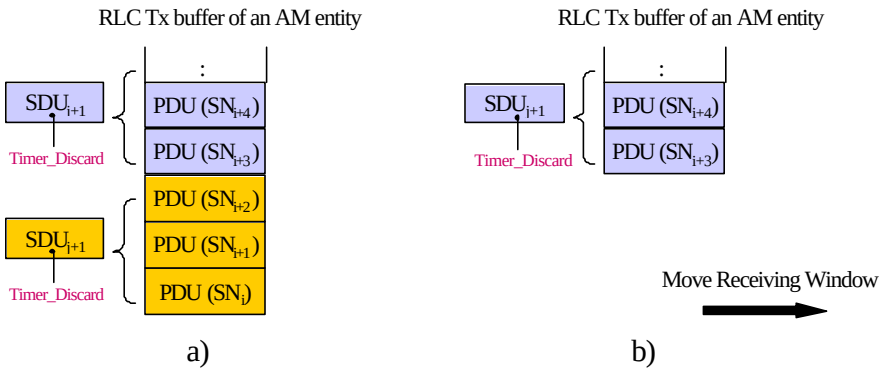


Figure 8. SDU discard mode: Timer based discard, with explicit signaling.

A question may arise: what happens if a Status PDU carrying SuFi MRW (or SuFi MRW_ACK) gets lost? To prevent a deadlock in such a case, a state variable VT(MRW) along with a timer Timer_MRW are utilized during the explicit signaling procedure. The upper bound of VT(MRW) designates a RLC protocol parameter referred as MaxMRW. When transmitting a Status PDU with SuFi MRW, the Timer_MRW timer is initialized and the variable VT(MRW), or transmission counter, is incremented. As soon as SuFi MRW is acknowledged, the state variable VT(MRW) and the timer Timer_MRW are disabled. If the Timer_MRW timer expires, SuFi MRW is retransmitted, the timer is reinitialized and the transmission counter is increased. If the transmission counter reaches the value of MaxMRW, the explicit SDU discard procedure is stopped. Instead of it, the RLC sender initializes a RLC reset procedure.

The RLC reset procedure follows the scheme of the just described explicit signaling procedure. Notwithstanding, instead of using Timer_MRW, VT(MRW), MaxMRW, there are used a timer Timer_RST, a state variable VT(RST) and a protocol parameter MaxRST. The Status PDU carrying SuFi MRW (respectively SuFi MRW_ACK) is replaced by a standalone RLC block called Reset PDU (respectively Reset PDU ACK). The reset procedure reinitializes RLC state variables and RLC timers. In addition, all SDUs that have begun to be transferred before the initialization of the reset procedure are discarded from the sender buffer. At the receiver side, all buffered Data PDUs are discarded. If even the RLC reset procedure fails, i.e. VT(RST) = MaxRST, the RLC sender indicates RRC an unrecoverable error.

Values of Timer_MRW (50ms, …, 900ms), Timer_RST (50ms, …, 1s), MaxMRW (1, …, 32), MaxRST (1, …, 32) are signalled by RRC via the RRC/RLC control interface.

Remark: A TM or UM RLC entity can be configured to utilize a very similar SDU discard mode called *Timer based discard, without explicit signaling*. As the name of the discard mode indicates, the explicit signaling procedure is not activated. After discarding co-SDU(s) blocks, the sender immediately continues with the transmission of next blocks without notifying the receiver about the discard event. The sender informs the receiver about the discard event *implicitly* by setting SN+2 in the next transmitted Data PDU after the discard event. This skip in SN introduces a missing Data PDU; from the receiver view a Data PDU that is out of order. As it has been mentioned in section 2.2.1, this "reordering" provokes a discard of all non-reassembled SDUs at the receiver side.

SDU discard after MaxDAT number of retransmissions

This SDU discard mode tries to keep the SDU loss rate constant on the cost of variable wireless delay of SDUs.

To a Data PDU is associated a RLC transmission counter alias a state variable VT(DAT), figure 9a. This variable is increased each time the corresponding Data PDU is scheduled to be (re)transmitted. A Data PDU can be transmitted up to a maximum number of times. This number corresponds to a RLC protocol parameter referred as MaxDAT. Value of this parameter can range from 1 up to 40 ([3G25331]); its value is signaled by RRC.

If a RLC transmission counter reaches the value of MaxDAT, the corresponding Data PDU is discarded from the buffer. Besides this Data PDU, all co-SDU blocks are as well discarded from the buffer (figure 9b). Subsequently, the sender initializes an explicit signaling procedure (and afterwards eventually RLC reset procedure) that has been detailed for the first SDU discard mode (*Timer based discard, with explicit signaling*).

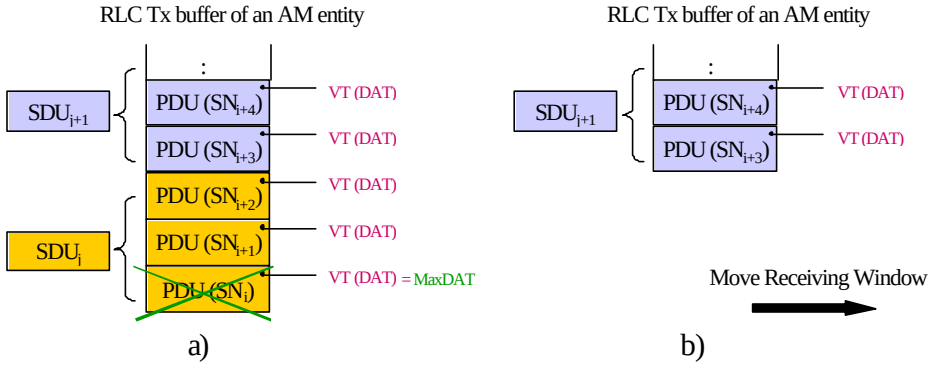


Figure 9. SDU discard mode: SDU discard after MaxDAT number of retransmissions.

No discard after MaxDAT number of retransmissions

This SDU discard mode is derived from the previous one (*SDU discard after MaxDAT number of retransmissions*). The only difference here is that the sender directly initiates the RLC reset procedure (for more see *Timer based discard, with explicit signaling*). The explicit signaling procedure is skipped.

Notice that the timer Timer_Discard (respectively the parameter MaxDAT) can be set to rather high values; 7,5s (respectively 40). If the RLC entity supports a service that does not have strict restrictions on the wireless delay (such a case of the UMTS background class), it can be assumed that the RLC entity sooner or later succeeds to get all Data PDUs correctly over the radio interface.

We will return to the SDU discard modes in chapter 4. In this chapter, we propose to introduce between the RLC and TCP protocol a specific mechanism that is based on SDU discard modes: *SDU discard after MaxDAT number of retransmissions* respectively *Timer based discard, with explicit signaling*.

2.4 Interlayer interfaces between RLC and adjacent layers

A large flexibility of the UMTS radio stack is made possible thanks to specifying number of messages that are exchange among the layers. This section studies in more details messages and information that are exchanged between the RLC layer and its upper and lower adjacent layer. The interlayer communication on the control interface between RLC and RRC is investigated as well as. Before detailing the interlayer communication, we remain different types of primitives that can be exchanged between layers.

2.4.1 Different types of primitives

We have seen in figure 2 (in section 1.2.2) that the UMTS radio stack is composed of several layers. Each layer offers specific services to the upper adjacent layer. The interlayer communication is described in terms of primitives. A primitive can be regarded as an conveyer that transports information between two adjacent layers. Besides data, a primitive can contain several parameters. Through these parameters, the source layer informs the destination layer about specific requirements to fulfill or about specific events; e.g. a SDU has been correctly transferred over the radio interface, an unrecoverable error occurs, etc. There are distinguished four types of primitives:

(i) Request (*Req*),
(ii) Indication (*Ind*),
(iii) Confirm (*Conf*),
(iv) Response (*Resp*).

The first primitive, *Req*, is employed by a layer to request a service from the lower adjacent layer such as the transfer of a SDU. The primitive *Ind* is utilized by a layer to indicate the upper adjacent layer of activities that concern this upper layer, e.g., reception of a SDU that is intended for this layer. With regard of the utilization and content of these primitives, *Req* and *Ind* are the principal primitives exchange between layers.

The remaining two primitives *Conf* and *Resp* are employed for interlayer communication that does not include SDUs exchanged. Through the primitive *Conf*, a layer confirms the upper adjacent layer an event that the upper layer has previously required to confirm, e.g., that a SDU has been correctly transferred over the radio interface. Contrary to *Conf*, the primitive *Resp* is used by a layer to inform the lower adjacent layer about a specific event, e.g., that the layer is currently in the suspend state.

2.4.2 Upper layer/RLC (AM entity)

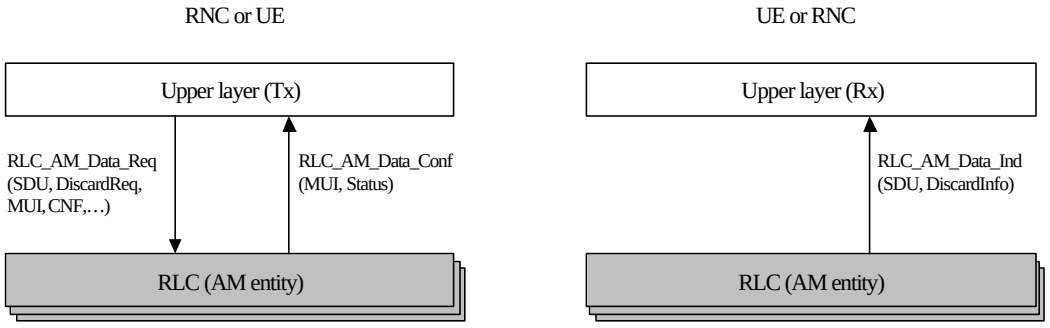
This section studies an interlayer interface between the upper layer and RLC (precisely an AM RLC entity). Our interest focus on primitives that convey data (SDUs) and information concerning the RLC SDU discard function. The primitive notation used in our text follows the notation used in the 3GPP specifications ([3G25321] and [3G25322]).

Figure 10 shows data and discard related information that is exchanged between an AM RLC entity and its upper adjacent layer. The upper layer can be either the RRC layer (if SDUs carry signaling messages) or a layer such as IP (or PDCP if the IP header compression is applied).

Data, i.e. a SDU, from the adjacent upper layer is delivered to the AM RLC entity via a primitive of type *Req*, called RLC_AM_Data_Req. In addition to the SDU, the primitive carries several parameters (figure 10a). Among other parameters, the primitive contains parameters called Unit Identifier (MUI), DiscardReq and Confirmation Request (CNF). Setting of these three parameters is just related to the accompanied SDU. In the next primitive RLC_AM_Data_Req (delivering the next SDU), these parameters can be set up differently. Contrary to MUI, parameters DiscardReq and CNF can be implemented by one bit (yes/no).

The parameter MUI identifies SDUs in a context of the interlayer communication; the MUI's value is assigned by the upper layer. The interlayer SDU "numbering" is not specified in [3G25322]. It depends on a concrete implementation of the radio protocol stack. By the parameter DiscardReq, the RLC entity is informed, whether the upper layer do or does not demand to be notified when the SDU is discarded. If so and the SDU is discarded, the RLC entity sets up a parameter Status, which is delivered to the upper layer via a primitive *Conf* called RLC_AM_Data_Conf. To identify the SDU at the upper layer, the primitive also carries MUI of the discarded SDU.

Besides the discard notification, the upper layer can as well demand the RLC entity to confirm the successful reception of a SDU by the peer RLC entity. The requirement indicates a parameter CNF. Once the SDU is successfully transferred (acknowledged), the upper layer is notified. This notification is again provided through the parameter Status carried in the primitive RLC_AM_Data_Conf. The parameter Status requires just one bit for its implementation (discard/successful reception).



a) transmitter side b) receiver side

Figure 10. Primitives containing data and SDU discard related information that are exchanged between an AM RLC entity and the upper layer (RRC, PDCP, etc.).

Setting up as CNF as DiscardReq, the upper layer knows a transmission status of every delivered SDU to the AM RLC entity. This means, whether a SDU has been transmitted over the radio interface successfully or unsuccessfully (i.e. discarded). Notice that the upper layer also implicitly learns an instance of the successful transmission of a SDU over the radio interface.

At the receiver side, a reassembled SDU is delivered to the upper layer via a primitive *Ind*, called RLC_AM_Data_Ind (figure 10b). If a discard event occurs at the RLC sender side, the RLC receiver is explicitly informed about the discarded SDU (via SuFi MRW, see section 2.2). If configured, the RLC receiver informs the upper layer about the discarded event. This notification is provided through a parameter called DiscardInfo, which is carried in the primitive RLC_AM_Data_Ind.

2.4.3 RLC/MAC

In this section, we turn our attention to the interface between RLC and the lower adjacent layers, i.e. the MAC layer. The description is done for RLC entities in general, without a respect if the RLC entity operates in TM, UM or AM.

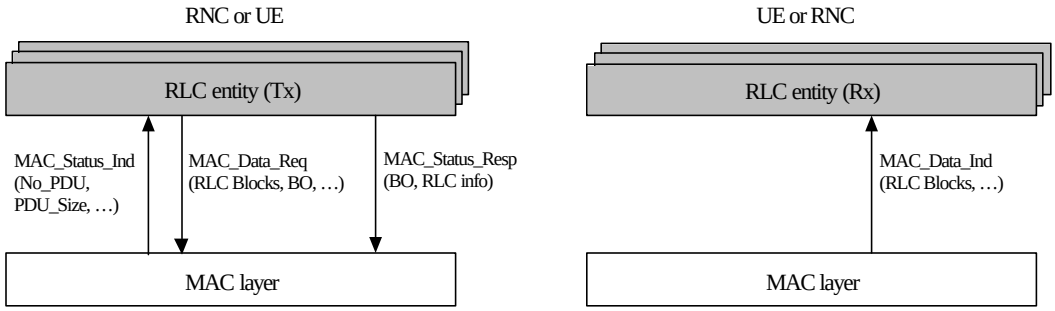
The MAC layer offers services to the RLC layer via logical channels. According to information from the RLC layer, QoS requirements and physical constraints, the MAC layer schedules

RLC blocks from different logical channels, i.e. from different currently established RLC entities.

Primitives that are exchanged between the RLC and MAC layers are shown in figure 11. Through a primitive MAC_Status_Ind, MAC indicates a rate by which a RLC entity may transfer data to the MAC layer. This rate is designated by two parameters: (i) No_PDU and (ii) PDU_size. The parameter No_PDU (respectively PDU_size) specifies the number of RLC blocks (respectively the size of RLC block) that can be delivered to MAC in a given TTI (Transmission Time Interval).

A RLC entity delivers MAC the required number of RLC blocks via a primitive called MAC_Data_Req. This primitive represents a response to the previously received primitive MAC_Status_Ind. Besides the RLC blocks, the primitive MAC_Data_Req also contains, among other parameters, a parameter called Buffer Occupancy (denoted as BO in figure 11a). This parameter indicates an amount of data (in bytes) that is available for transmission (first transmission and retransmission). If the MAC layer is connected to an AM RLC entity, Data PDUs outside the RLC transmission window and Status PDUs are as well included in the buffer occupancy ([3G25321]).

It can occur that a RLC entity needs to communicate the MAC layer other information then data or the primitive MAC_Data_Req cannot currently be used (e.g., because the RLC buffer is empty). In such case, the RLC entity utilizes a primitive MAC_Status_Resp. Through this primitive, a RLC entity may notify MAC for example that it has nothing to send or that it is in a suspended state or the RLC entity just informs MAC about its current buffer occupancy.



a) transmitter side b) receiver side

Figure 11. Primitives and parameters that are exchanged between the RLC and MAC layer.

At the receiver side, the MAC layer extracts RLC blocks from received MAC PDUs. Extracted RLC blocks are delivered to the appropriate RLC entities via primitives MAC_Data_Ind (figure 11b).

2.4.4 RLC/RRC (control interface)

There can be distinguished three groups of primitives on the RLC/RRC controls interface (figure 12): (i) Config, (ii) Suspend/Resume and (iii) Status. In case of a UM or AM entity, primitives in all three groups can be used. For a TM entity, just the first and third groups are applicable.

The first group is represented by a primitive called CRLC_Config_Reg. The RRC layer uses this primitive to establish, reestablish and release a RLC entity. A parameter E/R signals a concrete event. Besides the parameter E/R, the primitive contain several other parameters that depend on the RLC mode. For example, when establishing an AM RLC entity, the primitive CRLC_Config_Reg carries AM parameters such as timers (Timer_Discard, Timer_MRW, etc.), protocol parameters (MaxDAT, MaxMRW, etc.), ciphering elements (ciphering mode, ciphering key, etc.), type of the SDU discard mode, etc.

The second group includes three primitives: CRLC_Suspend_Req, CRLC_Suspend_Conf and CRLC_Resume_Req. The first primitive is used to suspend an RLC entity. The entity resumes the transmission via the primitive CRLC_Resume_Req. After receiving the primitive CRLC_Suspend_Req, the RLC entity enters to the suspend state. Nevertheless, the entity does not immediately suspend the transmission; it can still transmit several blocks in this state. An AM entity is allowed to send blocks up to a Data PDU with SN = VT(S) + N. The parameter N is indicated in the primitive CRLC_Suspend_Req. The state variable VT(S) is handled by the RLC entity; VT(S) indicates SN of the next fresh transmitted Data PDU. The RLC entity confirms the primitive *Req* via the primitive *Conf* (CRLC_Suspend_Conf) that contains current value of VT(S). In a case of a UM entity, the variable VT(S) is replaced by a variable VT(US).

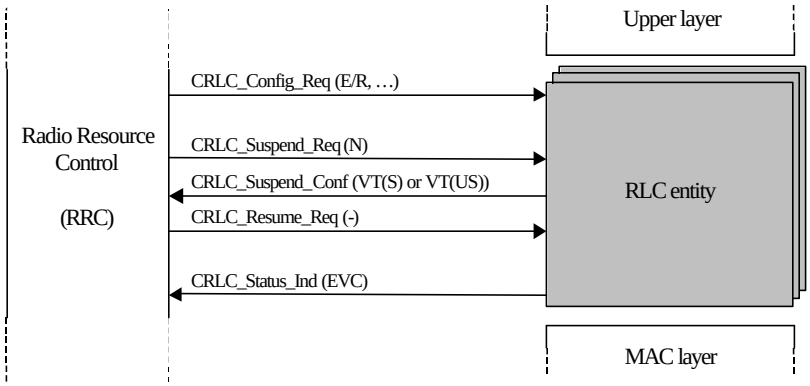


Figure 12. Primitives exchange on the control interface between the RLC and RRC layer.

The last primitive, CRLC_Status_Ind, allows a RLC entity to inform the RRC layer about the progression of communication or about specific events. The primitive contains a parameter called Event Code (EVC) that indicates a reason of sending the primitive; e.g., a variable VT(RST) reached MaxRST or a SDU discard event occurred, etc.

2.5 Some of differences between RLC of (E)GPRS and UMTS

Perceptible signs of gradual evolution of (E)GPRS towards the UMTS system can be observed at the RLC layer. The RLC of UMTS takes up RLC of (E)GPRS and further it ameliorates. The RLC version of UMTS offers more functions and is more adjustable than its predecessor in (E)GPRS.

The UMTS version of the RLC protocol mainly unifies header structures of RLC blocks for downlink and uplink directions. There is no difference between the header structures of a downlink and uplink Data PDU (respectively a control PDU).

To the exiting (E)GPRS RLC modes, AM and UM, the UMTS system introduces a third mode, transparent mode. Compared to GPRS, UMTS uses for each mode a specific Data PDU (Data PDUs defer in their header structures).

The AM of UMTS has considerably been enhanced in comparison with the AM of (E)GPRS. The acknowledgment policy has largely been enriched (see section 3.3) and several new features have been added such as SDU discards, detection of duplicated blocks or ciphering. In addition, an AM entity of UMTS can be set up to deliver reassembled SDUs to the upper adjacent layer either in-sequence or out of sequence. In (E)GPRS, received SDUs are always delivered in-sequence.

The behavior of a UM RLC entity slightly differs in both systems. In a case of UMTS, a missing RLC block provokes a discard of all RLC blocks that are part of the same SDU(s), i.e. co-SDU blocks. This way, reassembled SDUs are delivered to the upper layer in-sequence on the cost of discarding any out of sequence data. In (E)GPRS, a UM entity delivers reassembled SDUs as they arrive at the receiver, i.e. even out of sequence.

A number of bits that are reserved for the block numbering differ in both systems. UMTS uses modulo 2^{12} for AM and 2^{7} for UM. In GPRS (respectively EGPRS, Enhanced GPRS), blocks are numbered with modulo 2^{7} (respectively 2^{11}), regardless of the used mode.

2.6 Remarks concerning protocol implementations

Before terminating our study about the RLC protocol, we mention few remarks concerning a development and implementation of protocols in general.

2.6.1 Specification and development of protocols

For the specification and development phase of a protocol, the International Telecommunication Union (ITU) introduced and defined a formal language called Specification and Description Language (SDL, see e.g., [ITUa] or [M00b]). Besides protocols and the telecommunication domain, SDL is also applied in other areas such as aircraft, train or medical systems.

SDL is a graphical language that is formal and an object-oriented (version SDL-2000). The semantics behind each SDL symbol and concept are precisely defined. The graphic syntax of SDL is extremely intuitive. Thus, even non-protocol designers can quickly obtain from a graphical SDL script the overview of system structure and its behavior. Other formal techniques (e.g., LOTOS or ESTELLE) are less intuitive compared to SDL.

A SDL methodology used for specifying, analyzing, simulating, validation and implementing radio protocols of UMTS are discussed in [BB94]. A design and implementation of the RRC layer of UMTS with the use of SDL is described in [JC01].

The most commonly known (and used) tool that supports the SDL language is a commercial designer tool from a society Telelogic. As a matter of fact, as far as we know, there is presently no free software supporting the SDL language.

The Telelogic tool is composed of several modules. One of these modules also makes possible to generate an execution code in C from a SDL script. However, this module is not a standard component of the Telelogic package. If it is required, the module has to be paid extra.

Compared to big companies that are engaged in developments of protocols, the exploitation of the commercial software at universities can bring a problem in form of its license policy. After expiration of the license, the software become non-operational until the license is re-paid. Consequently, earlier developed SDL scripts and projects become unusable unless the license is extended. Thus, a long-term academic work with the commercial software may be a little bit tricky.

In the framework of the RNRT project MPRIM (introduced in section 1.5), we have developed a simple version of the RLC protocol with the use of SDL ([BC01]). However, not to dispose of the module that generates the execution C code from the SDL script, we have turned our attention to an implementation of the protocol directly in the language C. The C version of the RLC protocol subsequently became a part of a UMTS test-bed developed at ENST. The test-bed allows simulating a TCP data transfer over the RLC protocol of UMTS; the test-bed is detailed in section 4.3.1.

2.6.2 C versus Java

In spite of the existence of the formal language SDL, number of companies prefers to develop protocols by using directly standard programming languages. Recent years, a question is

arising: what type of a programming language should be selected for the implementation of lower/higher layer protocols? Is it better a higher programming language such as Java or a lower programming language such as C?

When developing and implementing a protocol, there have to be taken into account several different aspects such as the rapidity of a development process, the portability on different platforms, the rapidity of code execution, requirements on the memory size, etc.

Contrary to C, the object orienting Java allows accelerating the development process of protocols. At a low cost, Java makes possible to easily reutilize a developed protocol on different platforms. However, Java is more exigent on memory then C. Furthermore, a Java code is much slower than that one in C. There can be considered about a factor of 10 between the execution rapidity of the C code and the Java code.

For the time being, C remains a standard implementation programming language as for lower layer protocols (e.g., RLC) as for higher layer protocols (e.g., IP/TCP) of nowadays wired and wireless networks.

Chapter 3
Transmission Control Protocol in cellular networks

3.1 Introduction

The Transmission Control Protocol (TCP, see e.g., [S94], [T96] or [T99]) is a connection-oriented reliable transport layer protocol. Protocol mechanisms of TCP have been designed and optimized for utilization in wired networks with data rates from a few kbit/s up to hundreds of Mbit/s. In this type of networks, the protocol behavior adapts to fluctuating network conditions and makes possible to face congestion events in routers that are located in the connection path.

The protocol provides four major functions: (i) segmentation of data, (ii) flow control, (iii) error control and (iv) congestion control. The first three functions have been a part of the protocol since its first version ([RFC793]). The fourth function, the congestion control ([J88]), has been added later on as a result of poor performance of the original TCP with an increasing traffic in wired networks.

It might be pointed out that there are other transport protocols used in Internet besides TCP. For example, UDP (User Datagram Protocol, [RFC768], [T96]) or RTP (Real-Time Transport Protocol, [RFC1889]). However, we are interesting about reliable data transfers. Furthermore, recent studies show that more than 90% of transiting IP packets in nowadays Internet contains TCP segments ([L01]). Therefore, we consider in our studies TCP at the transport layer.

The next section discusses basic facts about TCP such as protocol soft states, ways of detecting a loss, etc. Section 3.3 briefly presents another reliable transport protocol called Stream Control Transmission Protocol (SCTP). Section 3.4 compares protocol mechanisms of RLC of UMTS (ITU protocol) and TCP (IETF protocol). Few TCP extensions for wireless networks consider a radio link with an ARQ protocol at data link layer. Section 3.5 analyzes the TCP behavior over RLC and the usefulness of some proposed TCP extensions when used in the UMTS environment is discussed.

3.2 Basic facts about TCP

In this section, we study TCP protocol mechanisms that are utilized during a data transmission itself. Studies concerning the establishing phase (known as the three-way handshake) and the closing phase of a TCP connection are left out. A detail description of these two imperative TCP phases can be found for example in [T96] or [C00]. Furthermore, if it is not explicitly

mentioned in the text, the study is provided for a TCP version called Reno. This is presently the most often implemented TCP version in Internet ([PF01]).

TCP segment and transmission window

Data arriving from an application layer are segmented into unit called segments or TCP segments. A TCP segment consists of a header (20 bytes, eventually up to 40 bytes if TCP options are used) and a variable data block size. The size of the largest data block that can be sent is called Maximum Segment Size (MSS). MSS is negotiated between the peer TCP entities during the establishing phase of a TCP connection. The negotiation procedure of MSS is provided in order to avoid segmentation/reassembling of IP packets within the connection path. The upper size of MSS is delimited by the size of Maximum Transfer Unit (MTU) at the IP level; the MSS and TCP/IP headers forms one MTU.

The TCP sender controls the emission rate of segments into the network through its transmission window. The current transmission window size (*twnd*) is given by the following expression:

$$twnd = \min(rwnd, cwnd) \quad (1)$$

where variables *rwnd* represents the current receiver window size and *cwnd* denotes the current congestion window size. Through *rwnd* (carried in feedback segments), the receiver informs the sender about the current available receiver buffer size (in bytes). In other words, the variable *rwnd* is used for the flow control function of TCP. The variable *cwnd* is handled by the sender. The growth of *cwnd* depends on the current protocol soft state: (i) slow start, (ii) congestion avoidance.

3.2.1 TCP soft states: slow start and congestion avoidance

The slow start soft state is utilized at the beginning of a TCP connection or after a congestion event occurs. The variable *cwnd* is initiated to a small value, typically to 1 or to 2 MSS; the initial value of *cwnd* indicates a TCP parameter called Initial Window (IW). Every received fresh Ack leads to increase of *cwnd* by one MSS ([RFC2581]):

$$cwnd_{i+1} = cwnd_i + \mathrm{MSS} \quad (2)$$

This growth results in an exponential increase of *cwnd* and consequently to the exponential increase of the TCP transmission rate. Analyses concerning the effectiveness of setting the initial congestion window size to a higher value than 1 or 2 MSS can be found for example in [RFC2757] or [IM02].

As soon as *cwnd* reaches a threshold called *ssthresh* (slow start threshold), the slow start soft state is replaced by the congestion avoidance soft state. In the congestion avoidance state, the growth of *cwnd* slows down. The variable *cwnd* is increased by one MSS per RTT (Round Trip Time). Thus, instead of the exponential increase, *cwnd* increases linearly. The sender remains in the congestion avoidance soft state until a congestion event occurs.

3.2.2 Detection of a segment loss

When detecting a segment loss, the sender automatically assumes congestion in the network and reduces its *cwnd*. This step results in a reduction of the TCP transmission rate and thereby controlling the congestion in the connection path. A TCP sender can detect a segment loss by two ways:

(i) timeout, i.e. by expiration of the TCP retransmission timer
(ii) reception of three duplicated Acks, i.e. by reception of the fourth same Ack

The duration of the retransmission timer is referred as RTO (Retransmission Time-Out, [RFC2988]). Each time a new segment (or a new sequence of segments) is sent, a retransmission timer is initialized. The quantity of RTO indicates the maximum delay for transmission of the segment (to the receiver) and transmission of the corresponding Ack (from the receiver to the transmitter). The management of the retransmission timer is depicted in [RFC2988]. This reference also includes expressions that are used to compute RTO.

A segment loss: timeout

If a segment loss is detected via the retransmission timer, the sender considers that the congestion is severe since no segments (i.e. Ack) have been received at the sender side.

The first non-acknowledged segment in the transmission window, i.e. the assumed lost segment, is retransmitted (fast retransmission algorithm). Subsequently, the protocol reinitializes the slow start soft state. The variable *cwnd* is set to the value of a parameter called Loss Window (LW, LW = MSS). The variable *ssthresh* is updated according to the following expression ([RFC2581]):

$$ssthresh = \max(FlightSize / 2; 2 * MSS) \quad (3)$$

where *FlightSize* represents an amount of outstanding data in the network, i.e. data that has been sent but not yet acknowledged. Furthermore, the retransmission timer is restarted with a new value of RTO_{new}; this update is known as “back off the timer” ([RFC2988]):

$$RTO_{new} = 2 * RTO_{old} \quad (4)$$

The previously discuss progression of the transmission window size during the slow start and congestion avoidance soft states is summarized in figure 13. The figure also shows the *twnd* and *ssthresh* update, when detecting a segment loss via the TCP retransmission timer.

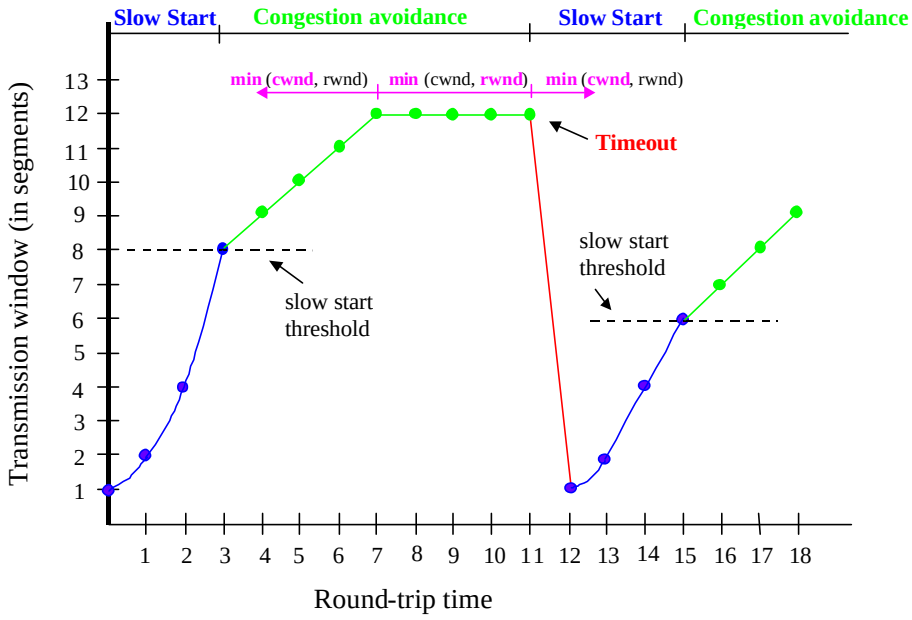


Figure 13. An increase of the transmission window size during different TCP soft states. The figure also illustrates the update of cwnd and ssthresh when detecting a segment loss via the timeout.

A segment loss: three duplicated Acks

If a loss is detected by reception of three DupAcks (duplicated Acks), the congestion is considered to be less severe since the sender still receives segments. As in the timeout event, the lost segment is retransmitted (fast retransmission algorithm). The next steps of TCP depends on the TCP version.

In a version Tahoe, the protocol resumes with the slow start soft state; *ssthresh* and RTO are updated according to expressions (3) and (4). The variable *cwnd* is set to the value of the parameter LW.

In a version Reno, the protocol enters into the fast recovery soft state. The TCP sender remains in this protocol soft state until a fresh Ack is received. When re-transmitting the lost segment, the retransmission timer is restarted (with the same value of RTO, [GL02]). The *ssthresh* is decreased according to expression (3). The varaible *cwnd* is updated ([RFC2581]):

$$cwnd = ssthresh + 3 \text{ MSS} \quad (5)$$

This update artificially "inflates" *cwnd* by the number of segments that have left the network (three) and that the receiver has already stored in its buffer. Every subsuequently received DupAck indicates that a segment has been removed from the network and is stored in the receiver buffer. The sender waits till a certain number of DupAck are received before resuming the transmission with new segments.

Once the TCP sender receives a fresh Ack, *cwnd* is set to value of *ssthresh* and the sender resumes the next transmission with the congestion avoidance soft state. The fresh Ack also acknowledges all segments sent between the lost segment and its retransmission version, if none of these segments have been lost.

Notice that neither the first nor the second DupAck produces increase of *cwnd,* i.e. new segments are only sent if the current *twnd* allows it. Generally, the transmission window is full and thus no segments are sent after receiving the first and second DupAck.

The TCP references [RFC2581] and [RFC2988] do not clearly state how to manage the TCP retransmission timer when receiving a DupAck. Reference [GL02] suggests restarting the timer with the current value of RTO when receiving the first or the second or the third DupAcks (this DupAck invokes the retransmission of a segment). According to [GL02], these suggestions are allowed by the TCP references [RFC2581] and [RFC2988], though they are not explicitly mentioned. The restart of the retransmission timer for a DupAck that arrives after the third one remains a research issue.

A detail simulation-based comparison of different TCP versions (Tahoe, Reno, New Reno, and TCP with SACK) for one and for several successive segments losses can be found in [FF96]; a wired network environment is considered during simulations.

A segment loss: use of a TCP extension called Limited Transmit

If there is few outstanding data in the network (small *cwnd*) or a large number of segments get lost, the TCP receiver may miss segments to be able to generate three DupAcks. In such situations, the sender cannot do anything else then to wait for an expiration of the retransmission timer to detect the loss and to react. A solution to this problem has been proposed in a TCP extension called Limited Transmit ([RFC 3042]).

Limited Transmit suggests transmitting new segments already upon receiving the first or second DupAck; the reception of the first or the second DupAck does not lead to any modification of *cwnd*. The sender can only send new segments if the following two conditions are satisfied: (i) the advertised receiver window size (*rwnd*) allows the sender to transmit new segments and (ii) the amount of outstanding segments remains less than or equal to *cwnd* + 2 segments.

In other words, the sender can send at most two new segments beyond the current congestion window size.

An advantage of this TCP extension (in addition to faster reaction of the sender to a loss) is less bursty transmissions. The emission of segments is clocked out by incoming Acks as well as by incoming DupAcks.

3.3 Stream Control Transmission Protocol

The TCP provides a single stream of data and ensures that data delivery of this stream takes place with the perfect sequence preservation. The protocol is not aware of the inter structure of transported information. If a loss occurs, the TCP receiver delays data delivery to the application until the correct sequencing is restored.

However, such strict TCP sequence preservation is not really necessary for applications such as web-browsing or SS7 (Common Channel Signaling System No. 7). If using SS7 over TCP, it is only necessary to maintain sequencing of signaling messages that affect the same resource (e.g., the same call or the same channel). Other signaling messages are just loosely correlated and can be delivered without having to maintain the overall sequence integrity.

In the case of web-browsing, it is generally not necessary to maintain sequence preservation between displayed objects of a web page. A user usually prefer to see something on the display rather than waiting long time for all objects of web-page to be received before displaying them. The ability to deliver and display web-page objects out of sequence result in better perceived performance from the view of users.

To provide data delivery at the application level without the sequence preservation, a reliable transport protocol called Stream Control Transmission Protocol (SCTP, [RFC2960]) was specified. Likewise TCP, the SCTP protocol is designed at the ground of IETF. SCTP is a connection-oriented protocol. Compared to TCP, SCTP is message-oriented protocol and supports framing of individual message boundaries; TCP is stream oriented and does not preserve any implicit structure within a transmitted byte stream.

The multi-streaming feature of SCTP is accomplished by the independence between the data transmission and the data delivery to the application level. Data are portioned into multiple streams that are independently transfers; e.g., a stream can be associated to one web-page object. A message loss in any of the streams will only affect delivery in that stream and not in other streams. Performance evaluation of the SCTP protocol can be found in [JS00].

3.4 Comparison of TCP and RLC (AM entity)

The comparison of protocol mechanisms is provided for TCP Reno and a RLC entity operating in the acknowledged mode.

Both protocols provide some similar functions such as numbering, segmentation/reassembling, flow control, error control, etc. However, their different history and backgrounds led to different implementation of their protocol mechanisms. The RLC protocol of UMTS is normalized by ITU. Its protocol mechanisms have been optimized for the air interface. The TCP protocol mechanisms have been optimized for the wired environment, without considering other types of environments in connection paths. The normalization of Internet protocols (TCP, UDP, IP, etc.) takes place on the ground of IETF.

Protocol units

Protocol data units and their numbering differ in both protocols. A TCP entity transmits segments and numbers bytes. The TCP Sequence Number (SN, modulo 2^{32}) indicates a sequence number of the first byte that is carried in the data field of a given segment; SN is one of flags in the TCP header. As to RLC, a RLC entity transmits and numbers Data PDUs (modulo 2^{12} for AM, 2^{7} for UM); Status PDUs are not numbered.

Notice that the denotation for the sequence number is same in both protocols, i.e. SN. Using the same notation at two different data structures and each time numbering different units could be confusing. At this point, the used notation in RLC of GPRS is clearer since GPRS denotes the sequence number of RLC blocks as BSN (Block SN, [LG99]).

A RLC block size (Data PDU, Status PDU) is steady and configured when establishing the RLC entity. If necessary, the RLC block size can be reconfigured during the establishing phase of the RLC entity.

In TCP, the data part of TCP segment may vary from segment to segment. The upper size of a segment is delimited by MSS (Maximum Segment Size). The MSS is negotiated between the peer TCP entity when establishing the TCP connection (par default MSS = 536, [RFC1191]).

Studies analyzing Internet traffic shows that more than 40% of all transiting IP packets in Internet are 40 bytes IP packets ([L01]). Such size of IP packet corresponds either to control packets that are exchanged during the establishment and closing phases of a TCP connection or to IP packets conveying just a TCP Ack. To avoid segmentation of this type of IP packets on the radio interface, the Data PDU size has to be set to 43 bytes (40 bytes for user data and 3 bytes for the AM RLC header). If the AM RLC just handles TCP Ack segments and the PDCP compression is applied, the Data PDU size should be set in regard to the compression ratio of PDCP. Otherwise, number of idle padding bits in the RLC blocks may be uselessly too high.

TCP data segments have generally much bigger sizes than are commonly employed sizes of Data PDUs (see table 6). Therefore, the RLC entity nearly always effectuates the segmentation of TCP data segments to get them over the radio interface.

ARQ mechanisms

Both protocols provide a reliable data transfer by using the ARQ technique. The RLC protocol implements Selective Repeat while TCP utilizes an "impure" Go-back-N method. By the term impure, we mean that the TCP sender does not necessarily retransmit all segments since the demanded one. Retransmitted segments are clocked by received Acks and the TCP sender retransmits segments according to these Acks.

The cumulative TCP Ack scheme raises a problem. The scheme does not provide the TCP sender with sufficient information to recover quickly from multiple losses within a single transmission window. To partly overcome this restraint, Selective Acknowledgment was introduced in TCP (SACK, [RFC2018]). The SACK scheme is defined as one of TCP options (for more about TCP options see e.g., [RFC1323] or [T99]). Its activation is negotiated between the peer TCP entities when establishing a TCP connection. If no other TCP options are utilized (e.g., Timestamps), the SACK scheme makes possible to inform the sender about maximally 4 losses in a single transmission window. The SACK scheme still preserves the imperative TCP retransmission concept, i.e. receiving three DupAck before re-transmitting a required segment. In other words, the TCP sender has to receive in the SACK filed three times the same Ack (SackDupAck) in order to retransmit the required segment.

Acknowledgment policy

The RLC acknowledgement policy is much more flexible compared to the TCP ones. The RLC protocol can employ two types of acknowledgement policies:

(i) the sender explicitly demands the receiver to send an Ack (this is provided by setting the Polling bit P in the RLC header), or

(ii) the receiver sends an Ack without being asked.

In the case (i), the sender may set the P bit by several ways: every N Data PDUs, every M SDUs, periodically (by using a periodic timer), each time a part of the transmission window (x %) is filled up, etc. In the case (ii), the receiver can send an Ack periodically and/or when detecting a missing Data PDU. The Ack triggers can be differently combined; e.g., the sender set the P bit up every N Data PDUs plus the receiver sends Acks periodically. The applied acknowledgement policy of the RLC entity is configured by RRC. Impact of two time RLC triggers on the RLC throughput and the wireless delay are investigated in [XC02a].

In TCP, the transmission of TCP Acks is fully under control of the receiver. The sender has no way how to demand the receiver to send him an Ack. After receiving a segment, the receiver can delay transmission of the Ack about a certain period (typically 200 ms) and to piggyback the Ack with eventual data that would be transferred in the opposite direction. How-

ever, an Ack is transmitted as soon as either a segment out-of-order or two segments in order are received.

The triggering of Ack (DupAck) for each TCP segment out-of-order imposes configuring the AM RLC entity to deliver reassembled SDUs (TCP data segments) to the upper layer in-sequence. Any change of SDUs order at the RLC level can produce the transmission of DupAck. These DupAcks can subsequently result in a useless retransmission at the TCP level and thus a degradation of the TCP performance.

Transfer of an Ack (NAck)

RLC Acks (or NAcks) are carried in Status PDUs through SuFi (Super Field). There are 4 types of SuFi that can be employed to indicate correctly/incorrectly received Data PDUs. These SuFis are: SuFi ACK, SuFi BITMAP, SuFi LIST and SuFi RLIST. The first super field, SuFi ACK, has the simplest structure. The SuFi just specifies the sequence number of Data PDU up to which, the receiver correctly received Data PDUs. The remaining three super fields have more elaborated structure. They contain a list of correctly and/or incorrectly received Data PDUs. The size of SuFi BITMAP, SuFi LIST, SuFi RLIST varies according to how many correct and/or incorrect Data PDUs they indicate.

If a RLC data transfer takes place in both directions (from the UE to the RNC and from the RNC to the UE), Ack/NAcks can be carried in the Piggybacked Status PDU. However, the use of the RLC piggybacked mechanism has restrictions, which are explained in section 2.2.2.

In TCP, the Ack field (4 bytes) is an integral part of the TCP header, i.e. every TCP segment carries an Ack. The validity of the Ack field is confirmed through a bit in the TCP header that is called Ack bit. This validation feature is necessary. For example, it can happen that in a TCP bi-directional data transmission one of the connection paths can be much faster (e.g., the data path) than in the other one (e.g., the Ack path). Thus, one of TCP senders can transmits several data segments carrying the same Ack before receiving a data segment from the slower direction. To avoid misinterpretation of these DupAcks as an indication of congestion, the faster TCP sender has to invalidate the Ack field of DupAcks by the Ack bit.

Detection of a loss

The RLC protocol detects a loss differently than TCP does. The RLC receiver exactly informs the sender about Data PDUs that has to be retransmitted; way of triggering RLC Ack/NAcks reports is discussed in paragraphs *Acknowledgment policy.*

The TCP protocol utilizes just positive Acks. A TCP NAck is represented through reception of four same Acks. Besides these duplicated Acks, the TCP sender uses the retransmission timer to detect a segment loss.

Transmission window

The initial size of RLC transmission (and reception) window is set up by RRC when configuring the RLC entity. Anytime during the connection, the receiver can demand the sender to modify its transmission window size. This modification is provided through SuFi WINDOW that is carried in a Status PDU. The super field indicates a new transmission window size. The required size of the window has to respect the maximum allowed value, which is specified by RRC (the minimum value is 1). Study analyzing impact of different RLC transmission window sizes on the TCP throughput can be found in [XC02b] or [ZL02].

The TCP transmission window size dynamically changes during a connection. The current size of the TCP transmission window is given by expression (1), which is discussed in section 3.2. The following table 6 summarizes some of the previously mentioned features of TCP and RLC.

	RLC	TCP
Concerned link	UE – RNC	End-to-end
ARQ mechanism	Selective Repeat	Go-Back-N SACK (TCP option)
SN modulo	Data PDUs	Bytes
Numbering (Sequence Number)	AM: $2^{12} - 1$ UM: $2^{7} - 1$	$2^{32} - 1$
Mode	Tr: unidirectional UM: unidirectional AM: bi-directional	Bi-directional
Management of Acks	Sender or/and receiver	Receiver
How to send an Ack	Status PDU Piggybacked Status PDU	TCP segment
Aver. size of data unit [bytes]	Tr: ≈ 0 to 80 ([3G34108]) UM: ≈ 20 AM: ≈ 16 to 80	≈ 500 to 1500

Table 6. A comparison of RLC and TCP features.

3.5 TCP over the UMTS radio interface

A typical TCP connection between a UE and a server via the radio interface of UMTS is shown in figure 14. A TCP transfer on the radio interface can be provided by one or by two peers AM RLC entities, i.e. one or two radio bearers. In a case of one radio bearer, the AM RLC entity manages as TCP data segments as feedback TCP Ack segments. If using two radio bearers, one AM RLC entity handles TCP data segments, whereas the second one manages TCP Acks. This second configuration makes possible to tune the setting of RLC parameters according to TCP segments that the RLC entity supports, i.e. the TCP data or TCP Ack.

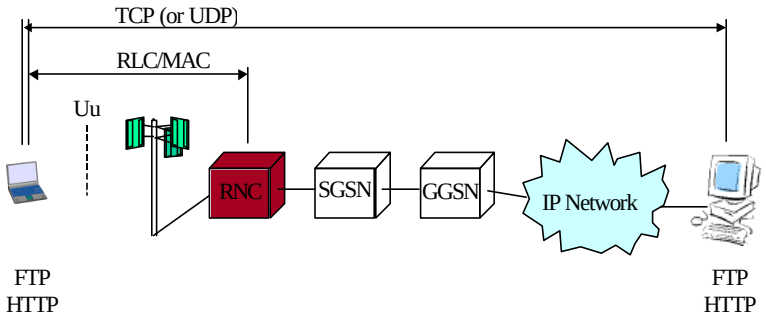


Figure 14. A TCP connection over the RLC protocol of UMTS.

The TCP error control and congestion control mechanisms have been developed with assumption that the data flow traverses networks with low Bit Error Rates (BER), less than 10^{-7}. With this hypothesis, a TCP sender presumes that a segment loss (detected either by DupAcks or by the retransmission timer) is an indication of congestion in the connection path. The sender reacts to the loss by invoking the congestion control mechanisms.

However, in wireless networks segments losses are produced for other reason than congestions. The losses are mainly due to high bit error rates (as high as 10^{-3}) and handoff procedures. These losses are interpreted by TCP as network congestion events. This misinterpretation results in degradation of the TCP performance.

3.5.1 Overview of TCP extensions for wireless networks

In the last decade, a great number of TCP extensions have been proposed in order to improve the TCP performance in wireless networks. This section discusses the usefulness of some proposed TCP extensions when used in the UMTS environment.

Split TCP

Different characteristics of wired and wireless environments inspired several authors to split a TCP connection on the wired-wireless border, e.g., Indirect TCP (I-TCP, [BB95]), Mobile

TCP (M-TCP, [BS96]), etc. In the split TCP, an end-to-end TCP connection is split into two independent connections: one connection between a mobile and a "base station" (or the RAN serving node) and another one between the base station and a fixed host. By this way, different characteristics of a wireless and a wired network are separated. In addition, instead of using a normal TCP, an adjusted transport layer protocol to the wireless environment has been suggested to be use over the wireless link. For example, authors in [YB94] propose using Selective Repeat Protocol (SRP) on top of UDP over wireless links. Such solution is a maladroit in the UMTS environment. The UMTS radio stack already implements its own ARQ protocols (RLC, MAC-hs) that are well optimized for the radio interface.

The split TCP include some drawbacks. One of them consists of breaking the so called "end-to-end semantic" of TCP ([RFC2757]). A client on the wired-wireless border sends an Ack to the sender (e.g., the fixed host) as soon as it receives data. A received Ack, by the sender (the fixed host), does not mean that the corresponding segment(s) has been successfully delivered to the intended receiver (mobile). Thus, the sender is informed about a successful transfer before data really reaches the receiver.

Extensions introducing an agent in a base station and/or modifying the TCP code

Another group of TCP propositions suggests to introduce at the 4th level an agent in the base station or/and modify the TCP code. Propositions belonging to this group are for example TCP extensions called Snoop Protocol ([BS95]), Wireless TCP (WTCP, [RM98]), Explicit Bad State Notification (EBSN, [BK96]), etc. Number of these TCP extensions is based on local retransmissions of TCP segments through the transport layer since they do not consider a radio link with an ARQ protocol (e.g., Snoop, WTCP, etc.). In UMTS, the use of a transport layer mechanism for local retransmissions is redundant.

Notice that extensions modifying the TCP code are not very well seen by the Internet community. This community rather prefers to modify the protocol stack of radio interface than the TCP/IP stack.

When using TCP on a radio interface, there can be considered two main issues. The first one is that the protocol does not dispose of a protocol mechanism that would allow the sender to distinguish congestion losses from wireless losses. If a wireless lost occurs, the sender assumes congestion and reduces the TCP transmission rate.

The second issue represents a high variability of wireless delay when using the ARQ technique on the radio interface. The high variability of wireless delay of SDUs may produce spurious TCP timeouts leading to unnecessary retransmissions of segments (e.g., [LK00]); a TCP spurious timeout is a timeout that would not have occurred if the TCP sender had waited "long enough". Such type of timeout occurs if the round trip time suddenly increase and the affected Ack arrived to late, i.e. after the timeout is already triggered. Besides the ARQ technique, other

factors affecting the SDU wireless delay are scheduling at the MAC layer, interleaving and coding at the physical layer.

Philosophy behind TCP extensions handling the first issue is that if the sender learns about a non-congestion loss, it can invoke a "non-congestion" control mechanism. TCP extensions dealing with the first TCP issue are for example Explicit Loss Notification (ELN, [BK98]) or Explicit Loss Notification with Acknowledgement (ELN-Ack, [DJ01]). The first extension (ELN) is proposed for an uplink data transfer, whereas the ELN-Ack extension is used for a downlink data transfer. In both extensions, there are introduced an agent in the base station and a new flag in the TCP header. Through the new TCP flag, the agent informs the sender if a lost occurred in the wireless (ELN) or wired part (ELN-Ack) of the connection path.

An example of TCP extension that deals with the first issue by just modifying the TCP code (i.e. without using an agent in the base station) is called Loss predictor ([BV98]). In this TCP extension, the sender utilizes simple statistics such as round trip time and/or throughput to estimate whether a next loss will be a congestion or wireless loss. Such estimation algorithm is utilized in a TCP version called Vegas ([BO94]).

Loss predictor may be implemented together with Congestion Avoidance Techniques (CAT, [BP95]). These techniques monitor a level of congestion in the connection path and propose to decrease or increase *cwnd*. If the congestion avoidance technique proposes to decrease (respectively increase) *cwnd*, the loss predictor would predict a next loss as a congestion loss (respectively a wireless loss). However, in order to correctly estimate the level of congestion, CATs require fulfilling several assumptions (e.g., congestion losses have to be preceded by a long queue build up at a router; a queue build up has to typically result in congestion losses, etc). If these assumptions are not fulfilled, the estimation of the cause of a loss becomes quite random ([BV98]).

In UMTS, the RLC protocol avoids wireless losses (thanks to its ARQ mechanism) unless the RLC SDU discard mechanism is activated. Contrary to a UM RLC entity, an AM RLC entity always applies the SDU discard mechanism. In section 2.3, we have noticed that the RLC parameters MaxDAT respectively Timer_Discard (which are related to the SDU discard mechanism) can be set to rather high values. By setting these parameters to high values, we may expect to get over the UMTS radio interface all TCP segments sooner or later. On the other hand, high values of MaxDAT respectively Timer_Discard may results in TCP spurious timeouts.

A TCP extension dealing with spurious timeouts is called Eifel algorithm ([LK00], [LM02]). This algorithm tries to eliminate the TCP retransmission ambiguity, i.e. the TCP sender inability to distinguish an Ack for an original transmitted segment and an Ack that corresponds to its retransmission version. To do this, the Eifel algorithm uses the TCP option called Timestamps ([RFC1323]).

Each time a TCP retransmission occurs, the sender stores a timestamp value of the retransmitted TCP segment. Besides the timestamp value, the current values of *ssthresh* and *cwnd* are as well saved. A timestamp value of the first received Ack (after the retransmission event) is compared with the stored timestamp value. If the timestamp value in the received Ack is smaller than the stored one, the retransmission was spurious. In such a case, the TCP sender restores values of *ssthresh* and *cwnd*; the sender resumes the transmission with the next fresh segment.

Instead of using Timestamps, the Eifel algorithm could use a new specific bit in the TCP header. Through this bit, the sender would differently mark an original transmitted segment and its retransmission version. However, the Timestamps option has an advantage of being already defined as a standard, which is widely deployed in Internet. On the other hand, the Timestamps filed occupies in each segment 10 bytes in the TCP option part. Thus, if the SACK option simultaneously is employed, the SACK field can maximally indicate 3 segment losses in a single transmission window (the use of other TCP options further decrease this number).

A disadvantage of the Eifel algorithm can be seen in a fact that the TCP sender detects a spurious timeout after retransmitting a segment. This retransmitted segment is uselessly transferred over the radio interface. A TCP extension that tries to prevent spurious timeouts is called EBSN (Explicit Bad State Notification, [BK96]). The extension is just proposed for TCP downlink data transfers. The EBSN introduces a TCP agent in the base station. The agent sends the TCP transmitter (fix host) an EBSN message whenever the base station fails to transmit a TCP segment over the wireless link; the agent locally retransmits discarded TCP segments. Upon receipt of an EBSN message, the TCP transmitter resets the TCP retransmission timer.

To reduce unnecessary TCP retransmissions during periods of high BlERs on the UMTS radio interface, we propose to introduce between TCP and RLC a specific mechanism called Early Error Notification (EEN). We detail the EEN mechanism in section 4.4.

Extensions for transaction oriented services

Large number of established TCP connections support transaction-oriented services (e.g., web-browsing). The term “transaction” refers to a request sent by a client to a server along with the server response. The request/response paradigm is used by many application protocols such as HTTP (Hypertext Transfer Protocol) as by a number of network control and management protocols such as DNS (Domain Name System). A transaction can be characterized as an unsymmetrical connection; the client requests data and the server responds.

Internet traffic surveys show that a large part of the web traffic consists of short HTTP transactions. About 40% of web transfers can fit into a one 1500 byte segment and about 63% fit into two 1500 byte segments ([CG99]). The mean size (respectively the median size) of a web document was found to be 4,4 Kbytes (respectively 2 Kbytes, [SN99]).

Due to the setting and closing TCP phases, a TCP connection is poorly amortized for transactions with a few data segment that is exchanged. Many web servers employ HTTP redirections, where a redirection operation itself represents a short TCP connection. All these initial and closing non-data TCP segments have to be as well transferred over the radio interface via the scarce resources. Furthermore, the TCP protocol mechanisms (congestion control, ARQ, etc.) introduce extra data processing and they are excessive for few segments that are finally transferred.

To reduce a number of exchanged segments in the initial phase of a TCP connection, there was proposed an extension called TCP for Transaction (T/TCP, [RFC1644], [SN99]). This extension is only useful for small amount of transferred data. As the amount of transferred data increases the effectiveness of T/TCP decreases ([SN99]). In addition, the effectiveness of T/TCP depends on the utilized version of HTTP at the application level. If using HTTP/1.0 ([RFC1945]), a new TCP connection has to be opened for every transaction, i.e. the three-way handshake takes place. However, in a case of the advance version of HTTP (HTTP/1.1, [RFC2616]), one TCP connection can be reutilized for more transactions. Thus, with HTTP/1.1 at the application level, T/TCP loses its benefit. This is one of reasons why T/TCP is not recommended to be deployed in Internet; another reason represents its higher vulnerability to attacks compared to the normal TCP ([RFC2757]).

The HTTP/1.1 and T/TCP are an example of this that an application enhancement may be sometime more effective than a transport level enhancement. Other example of application enhancement is the use of proxy servers in Internet. A user downloading a web page that is not in the proxy cache, the proxy server does not bring any benefit. However, a user demanding a web page already downloaded in the proxy cache takes advantage of a lower TCP RTT. If a dedicated (radio) channel is used for the TCP transfer, the channel is better utilized (for more see section 4.3.2).

Another extension dealing with transaction services, called Hybrid TCP-UDP ([CG99]), suggests a dynamic transport protocol selection between TCP and UDP. The hybrid scheme combines the best of both protocols. For short transactions, the schema profits stateless UDP low overheads. For large data transfers, there is taken advantage of TCP mechanisms (congestion control, ARQ, etc.)

In the hybrid TCP-UDP scheme, a client firstly attempts to use UDP as the transport protocol. If UDP appears to be a “wrong choice”, the client retries the transmission with TCP. The notation wrong choice means for example that a response is too large to fit into a single UDP segment or a UDP segments get lost or the contacted server does not implement the hybrid scheme, etc.

The hybrid scheme reduces load for servers because fewer TCP connections are set-up, maintained, torn down and thus fewer segments are processed. Since fewer TCP control seg-

ments are needed, the network traffic is reduced and the radio resources better utilized; there are less “non-data” segments carried over the radio interface. The scheme is incrementally deployable in Internet since it requires only changes at the application level; the operating system kernel code remains unchanged.

A more detail discussion about others TCP extensions and about issues concerning TCP on wireless links can be found for example in [BP97], [TM02], or [RFC2757]. Furthermore, the third reference mentions if a given TCP extension should be or should not be deployed in nowadays Internet.

Chapter 4
Study of interactions between TCP and RLC in UMTS

4.1 Introduction

In the previous two chapters, we have analyzed protocol mechanisms of RLC (UMTS) and TCP. We have observed the RLC protocol is highly adaptable. Specific primitives allow tailoring the protocol for different QoS requirements. The ARQ mechanism of RLC (Selective Repeat) together with its very flexible acknowledgment policy allows an AM RLC entity to deal with erroneous data much faster than TCP would do itself with out using the RLC protocol. Notwithstanding, the introduction of RLC below TCP goes with interactions between the protocols. This chapter points some of TCP/RLC interactions out and proposes a remedy to one of them.

Chapter 4 is decomposed into three major parts. The first part (section 4.2) briefly describes Internet Control Message Protocol (section 4.2.1) and Explicit Congestion Notification (section 4.2.2). These two IP/TCP level mechanisms allow informing TCP transmitting entities about specific events that occur in TCP connection paths (e.g., invalid destination address, router congestions, etc.).

The second part of the chapter (section 4.3) deals with issues concerning the RLC buffer occupancy when using at the transport layer TCP. Our studies focus on TCP downlink data transfers and on an AM RLC buffer that is located in the RNC entity. Section 4.3.1 describes a simulation environment. Section 4.3.2 studies two of factors affecting the RLC buffer occupancy: (i) wired Round Trip Time (RTT_{wired}) and (ii) Block Error Rate (BlER) on the radio interface. In addition, impact of different RLC buffer sizes on the TCP performance is analyzed. The second part of chapter 4 is concluded in section 4.3.3.

The third part of the chapter (section 4.4) studies a specific mechanism called Early Error Notification (EEN) that we propose to introduce between RLC and TCP. The EEN mechanism is based on a RLC SDU discard procedure that is depicted in chapter 2. Section 4.4.1 provides more details about the proposed EEN mechanism; a TCP uplink data transmission is assumed. Simulation experiments and results are described in section 4.4.2. Contrary to the uplink data transmission, in the downlink case, the TCP transmitting entity and the corresponding RLC entity are each located in different network elements. Therefore, the proposed EEN mechanism cannot be directly applied for TCP downlink data transfers as it is described in section 4.4.1.

A possibility of using EEN for downlink transmissions is discussed in section 4.4.3. The EEN analyses are concluded in section 4.4.4.

4.2 Congestion and error notification in IP networks

As discussed in section 3.1, the TCP sender detects congestions (segment losses) in the connection path through two methods: (i) by expiration of the TCP retransmission timer or (ii) by reception of three duplicated Acks. Besides these purely TCP mechanisms, TCP can also learn about congestions by exploiting congestion and error notification mechanisms that are implemented at the IP level. Two of these mechanisms are Internet Control Message Protocol and Explicit Congestion Notification.

4.2.1 Internet Control Message Protocol

To provide feedback about communication in IP networks, there was specified (in 1981) a protocol called Internet Control Message Protocol (ICMP, see e.g., [RFC792] or [C00]). The protocol is an integral part of the IP layer and must be implemented by every IP module in networks ([RFC1122]).

ICMP reports problems concerning a delivery or handling of IP packets in networks. For example, if a router discards an IP packet (i.e. a congestion event occurs), the router may send the TCP source an ICMP message called Source Quench. Through this message, the overflowed router demands the TCP source to slow down its transmission rate. When a TCP sender receives Source Quench, a recommended procedure ([RFC1122]) is to resume the next transmission with the TCP slow start soft state, i.e. as if a TCP timeout event occurred.

ICMP messages are convey by IP packets. Therefore, there is not any guarantee that the ICMP Source Quench message (as well as any other ICMP message) reaches its destination. Thus, there is not a guarantee that the TCP sender reduces its transmission rate.

Several types of ICMP messages are specified (e.g., Destination Unreachable, Echo, Parameter Problem, etc.); a complete list of ICMP messages can be found for example in [C00]. In the framework of the EEN mechanism, we propose to introduce a new ICMP message denoted as Virtual Discard. The generation and use of this ICMP message is detailed in 4.4.3.

4.2.2 Explicit Congestion Notification

The second TCP/IP level mechanism is called Explicit Congestion Notification (ECN, see e.g., [RFC3168], [RFC2481] or [F94]). In comparison with the “multi-functional” ICMP, the ECN mechanism is “mono-functional”. The only function of ECN is to warn a TCP sender about an oncoming congestion in the TCP connection path. Knowing in advance about an approaching congestion, the sender can take necessary steps to avoid segment loss(es). Oncoming

congestions are detected in routers through a technique called Active Queue Management technique (for more about them see e.g., [RFC2309] or [BP95]).

If a TCP sender receives a notification about an oncoming congestion (by setting a new flag in the TCP header), the sender reacts to this notification as if a congestion loss occurred. This means, the sender halves the *cwnd* (congestion window) and *ssthresh* (slow start threshold). However, no segment is retransmitted; the notification informs about an oncoming congestion, not about an occurred congestion.

Since the ECN mechanism introduces few new flags in the IP and TCP headers, there have to be modified source code as IP as TCP. Discussions about the usefulness of implementing the ECN mechanism in IP networks can be found for example in [TM02] or [IM02]. According to reference [IM02], the potential improvement of the TCP performance can only be achieved if a majority of network routers implement the ECN mechanism.

In our simulation scenarios in section 4.3, we consider that neither the ECN mechanism nor the ICMP message Source Quench is used to inform the TCP sender about the RLC buffer state.

4.3 RLC buffer in a TCP connection path

One of significant RLC parameters is the buffer size of the considered RLC entity. An interesting question is how different TCP protocol soft states affect the RLC buffer occupancy. The buffer occupancy by return affects the TCP RTT (Round Trip Time). Two different cases have to be distinguished; a TCP uplink and a TCP downlink transmission. In the uplink case, the RLC (UE) buffer occupancy is determined by the operating system of UE and its internal flow control besides the TCP soft states. Our studies consider the downlink case, where the internal UE flow control does not intervene to the RLC (RNC) buffer occupancy.

4.3.1 Simulation environment

This section details a simulation environment that is utilized in our studies concerning interactions between TCP and RLC.

The RLC buffer experiments as well as the EEN experiments are realized on a UMTS test-bed that is developed at ENST. The test-bed makes possible to simulate a TCP connection over the UMTS radio interface. The test-bed development has mainly focalized on protocol mechanisms of RLC and TCP. Other protocol layers are modeled in a simple way.

Platform of the test-bed

The test-bed platform consists of two computers that are connected via Ethernet (figure 15). The first computer models a UE entity (User Equipment), whereas the second one models the

Node B, RNC entity (Radio Network Control), Internet and Server. In the next description, we shortly denote the first computer (respectively the second one) as a UE module (respectively a RNCIS module). The radio interface is modeled at the transmitting side, i.e. ether by UE or by RNCIS module (for more see next text). There can be considered a third computer that may be used to observe the communication between the first and second computers.

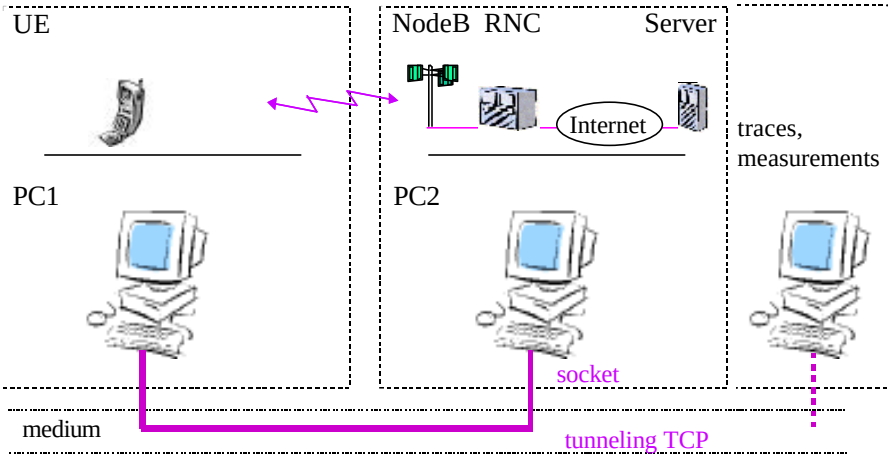


Figure 15. Platform of the UMTS test-bed.

The modules UE and RNCIS communicate via Ethernet by using a socket interface. Of course, the socket interface can be set in such a way that just one PC is employed for both UE and RNCIS modules. Two PCs are used in order to better observe simulation runs in every module and to debug the code.

Data exchange in Ethernet is provided through a tunneling TCP. The tunneling TCP has nothing to do with the simulated TCP in the test-bed. The tunneling TCP is assumed to be error free. The test-bed simulates errors itself.

Layer architecture and features of the test-bed

The overall layer architecture of the test-bed is shown in figure 16. The test-bed is decomposed into several layers or entities; every layer represents a standalone file (C file). Data processing in modules is controlled by entities called Supervisors; each module contains its own Supervisor.

The test-bed does not use the internal clock of the computers. Time is simulated and is managed by UE Supervisor. Simulated time (T_i) is proportional to the value of Simulated Frame Number (SFN), i.e. $T_i = SFN_i * T_{SRF}$, where the parameter T_{SRF} is a duration of one Simulated Radio Frame (SRF). The T_{SRF} is set to the basic radio frame period, i.e. 10 ms (the initial values of SFN and T_0 are zero). The simulated time granularity corresponds to the duration of T_{SRF}.

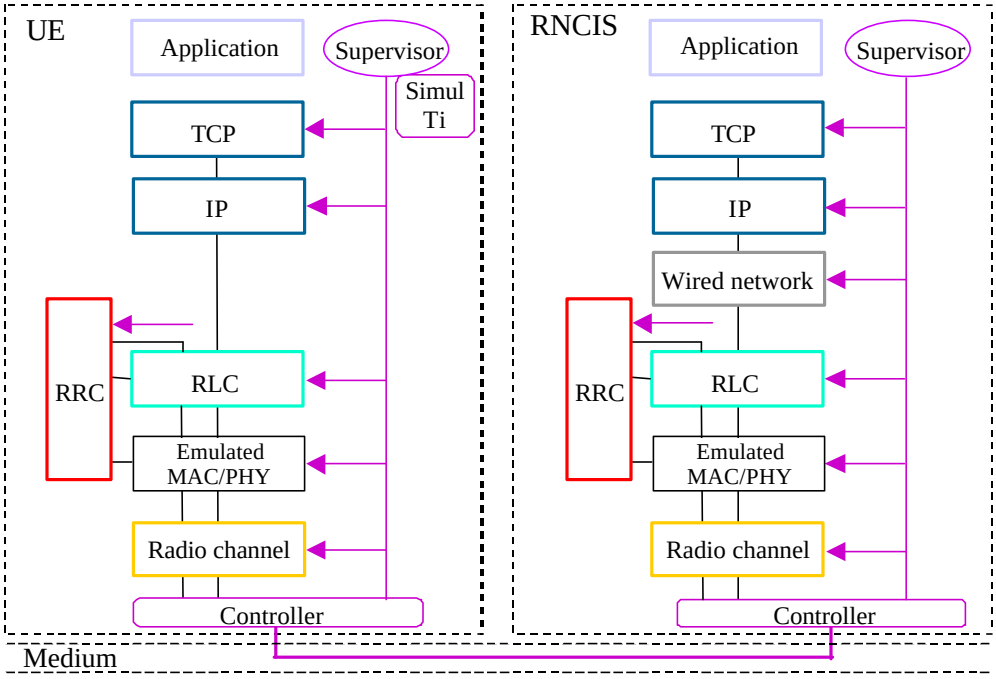


Figure 16. Overall layer architecture of the UMTS test-bed.

During one SRF, data is sent from the UE to the RNCIS module and vice versa (figure 17). At first, the UE module represents the transmitter and sends the RNCIS module (receiver) data. Subsequently, the modules exchange their roles; the RNCIS module becomes the transmitter sends the UE module (receiver) its data. The data processing in the currently transmitting module passes through the Application entity to the Controller entity, i.e. down in the protocol stack. In the receiving module, the data processing is up in the protocol stack, i.e. through the Controller to Application entity. The order in which entities are passed through remains same in every radio frame. An entity is passed the control although if there is no data processing that has to be performed by the entity. After exchanging data by both modules, the time T_i is increased and the above-descried procedure is repeated.

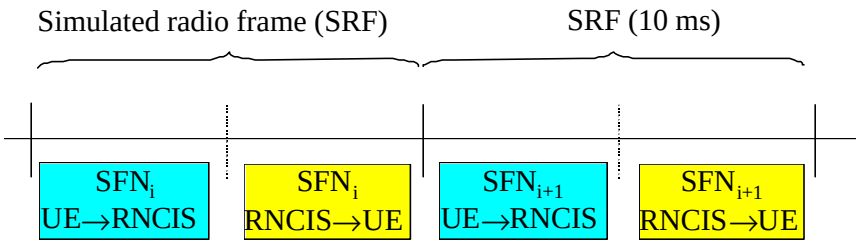


Figure 17. Principle of data exchange between the UE and RNCIS module in radio frames.

An entity (C file) expects as the input parameter a pointer to data structure that the entity will process. As the output parameter, the entity provides a pointer to data structure that will be processed by the following entity. Besides the pointer to input data, an entity is also passed pointers to: the entity parameters, the supervisor parameters and the trace parameters.

The access to the socket interface (read/write procedure from/to a socket) is managed through Controller. Furthermore, Controller adjusts internal data structures utilized in modules to data structures utilized in radio frames.

Functions provided by the test-bed layers

As it has been mentioned, the development of the test-bed mainly focalized on the TCP and RLC protocols. Implemented mechanisms of these protocols closely follow recommendations in [RFC2581], [RFC2988] and [3G25322],

The RLC protocol implements one RLC entity that operates in the acknowledge mode, i.e. there is one combined RLC entity for the transmission and reception. The AM entity provides the following functions: segmentation/reassembling, concatenation/separation, flow control, error control (ARQ, Selective Repeat) and in-sequence delivery of received SDUs to the upper layer. If concatenation, padding or piggybacking takes place, the LIE flag is added (for more about this flag see section 2.2.2). Furthermore, the AM entity is configured to employ the SDU discard mode *SDU discard after MaxDAT number of transmissions*; which is explained in section 2.3. Possible structures of Data PDUs and Status PDUs are shown in figures 6 and 7 (section 2.2.2).

The test-bed implements TCP version Reno; i.e. the following protocol mechanisms are implemented: Slow Start, Congestion Avoidance, Fast Retransmits and Fast Recover (section 3.2). The establishing phase (three-way handshake) and the closing phase of a TCP connection are not implemented. A TCP transmission directly starts with the data transfer itself. TCP uses during a connection the TCP option Timestamp (10 bytes). Therefore, the TCP header size is considered in our simulation scenarios to be 30 bytes instead of 20 bytes. The protocol can activate or deactivate the delayed Ack function; i.e. the receiver acknowledges without any delay every received segment. A received in-sequence segment is released (read) from the TCP receiver buffer as soon as the receiver processes its TCP header. The SACK option (section 3.4) is not implemented.

When receiving an Ack out of order, the TCP retransmission timer is managed according to suggestions in [GL02]: the first and second DupAck restart of the retransmission timer, the timer is restarted with the current value of RTO (section 3.2.2). DupAcks received beyond the third one do not have any influence on the running timer.

The remaining layers of the test-bed are modeled in a simple way. The IP layer adds the IP header (20 bytes) to a TCP segment; respectively removes the IP header.

The wired entity (located in the RNICS module) represents the wired network. The entity simulates wired delays and introduces congestion events (discards) in the wired network. The wired network buffer is managed according to the FIFO policy (i.e. first in, first out). This means that the wired network does not reorder IP packets.

The MAC/Phy entity emulates in a simple way the MAC and physical layers. At the beginning of a radio frame, MAC informs the RLC entity how many RLC blocks should be delivered to the MAC/PHY entity. When delivering blocks, the RLC entity also informs the MAC/Phy entity about the number of RLC blocks that are available for transmission (first transmission and retransmission).

The Radio Channel entity simulates wireless delay and errors on the radio interface. Likewise in the case of the wired entity, the wireless delay is simulated with help of a buffer that is managed according to the FIFO policy.

4.3.2 Simulation and discussion

RLC buffer analyzes consider four simulation scenarios. The first scenario (scenario A) studies a progression of the RLC buffer occupancy and of the SDU wireless delay during different TCP soft states. Scenarios B and C investigate how the RLC buffer occupancy and the SDU wireless delay change for different values of RTT_{wired} (scenario B) and BlER on the radio interface (scenario C). The last scenario, scenario D, analyzes the impact of different RLC buffer sizes on the TCP performance.

Parameter setting

Simulations are realized on the UMTS test-bed that is described in section 4.3.1. The simulation model and protocol architecture are shown in figure 18.

All four scenarios assume TCP downlink file transfers. Size of the transmitted file is 1 Mbytes. The RTT_{wired} (i.e. the RTT between the server and the RNC entity) follows a normal random variable $N(m,d^2)$. The standard deviation d is set to 20 ms ([KB98]), whereas the mean value m varies according to scenarios. The IP packets are delivered to the RNC entity (respectively the server) in the same order as they arrive from the server (respectively from the RNC entity).

The $RTT_{wireless}$ (i.e. the RTT between the UE and the RNC entity) equals to 60 ms (or 6 SRFs), if no RLC retransmission occurs. Time processing of received data in the UE and RNC entities is considered to be zero second.

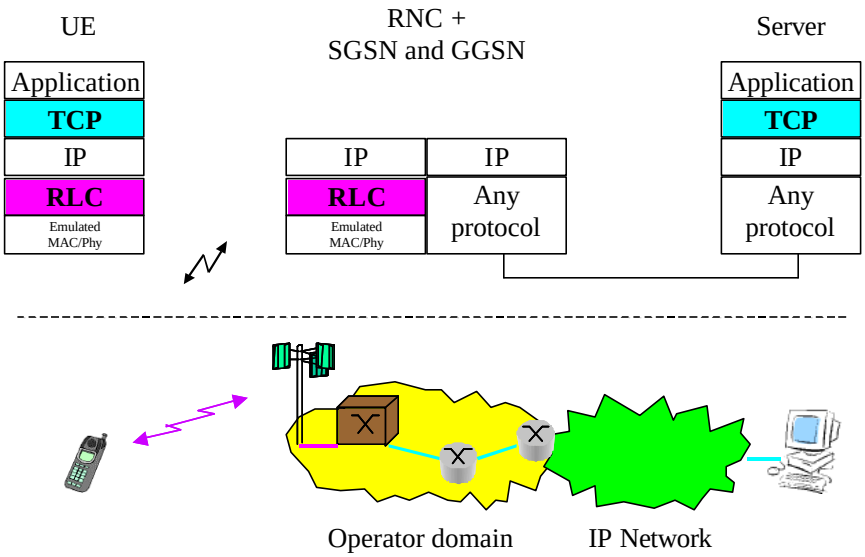


Figure 18. The simulation model and layer architecture.

The TCP (Reno) transmission directly starts with the data transfer. We set the MSS (Maximum Segment Size) to be 600 bytes. The initial *cwnd* (congestion window size), i.e. the parameter Initial Window (IW), is set to one MSS. The parameter Loss Window (LW) is set as well one MSS. The initial value of *ssthresh* (slow start threshold) is set to 32 MSS (or 19,2 kbytes). The maximum value of the receiver's window size (*rwnd*), i.e. the maximum receiver buffer size, is 40 MSS (or 24 kbytes). The delay Ack function is activated; the receiver delays an Ack at maximum about 200 ms ([RFC2757]).

The RLC entity operates in the acknowledged mode. The entity provides segmentation; if possible, the entity performs concatenation. The reassembled SDUs are delivered the upper layer (i.e. the IP layer) in sequence. The Data PDU size is set to 80 bytes. Possible structures of Data PDUs are shown in figure 7 (in section 2.2.2).

The RLC windows of transmitter and receiver have the same sizes. The window size is set to sufficiently large values that the RLC transmission is not stalled due to the window size (the RLC transmission window size = 2047 Data PDUs, which is the maximum size [3G25331]). Impact of different values of RLC transmission window size on the TCP throughput is investigated in [XC02b] and [ZL02].

The RLC parameter MaxDAT, which delimitates the maximum number of transmission attempts per Data PDU, is set to 32 (for more about the MaxDAT parameter see section 2.3). The effect of different MaxDAT values on TCP performance is studied in [LV01].

The AM entity uses an acknowledgment policy, where the transmitter explicitly (through the polling bit P) demands the receiver to send Ack/NAck reports (section 3.2). The transmitter

sets the P bit up in one of Data PDUs carried in each radio frame. A radio frame can transmit up to six Data PDUs. This corresponds to the 384 kbit/s user data rate. One IP packet carrying a TCP data segment is transmitted at least in two radio frames; one IP packet is segmented into 9 Data PDUs (650bytes / 80 bytes). The duration of a radio frame and TTI is set to 10 ms. The RLC Ack path is assumed to be error free.

On the radio interface, a dedicated transport channel is used for the TCP file transfer, i.e. the radio channel is dedicated to the TCP connection during the whole data transfer. Handoffs are not considered in our experiments. The error process on the radio channel is supposed to be memory less. The parameter setting is summarized in table 7.

Application (file transfer)	Size of the transmitted file	1 Mbytes
TCP (Reno)	MSS (Maximum Segment Size) *ssthresh* *rwnd* (max. TCP receiver's buffer size) Initial Window (IW) Lost Window (LW) Delay Ack	600 bytes 32 MSS (or 19,2 kbytes) 40 MSS (or 24 kbytes) MSS MSS 200 ms
Wired network	RTT_{wired} (RNC-Server) Errors	see scenarios see scenarios
RLC (AM entity)	Size of Data PDU Max. number of retransmissions (MaxDAT)	80 bytes 32 (…unlimited)
Wireless network	$RTT_{wireless}$ (UE-RNC) Radio frame duration and TTI Maximum number of Data PDUs in one radio frame BlER (Block Error Rate)	60 ms (if no retransmission) 10 ms 6 Data PDUs (…384 kbit/s) see scenarios

Table 7. Parameter setting during the RLC buffer analyses.

In scenarios A, B and C, there is introduced one congestion event in the wired network (denoted as Er in figures 19). The congestion event is realized by introducing a single TCP segment loss with the same sequence number in all three scenarios. The data transfer always starts with the same TCP sequence number. The TCP sender detects the segment loss by receiving three duplicated Acks.

Scenario A

The mean value m of RTT_{wired} is set to 200 ms (standard deviation d = 20 ms). The radio interface is error free (i.e. BlER = 0%) and the RLC buffer size is infinite. The radio link is considered to be bottleneck in the TCP connection path.

Figure 19 shows a number of SDUs (TCP segments) in the RLC buffer as function of time. The introduced loss in the wired network can be observed at time around t = 10s (denoted as Er in the figure).

When comparing figures 13 (section 3.2.2) and 19, we can observe that the RLC buffer occupancy reflects the TCP soft states slow start (SlSt) and congestion avoidance (CgAv).

The TCP sender transmits segments in bursts. The burst size exponentially increases during the slow start soft state. The RLC buffer occupancy starts growing exponentially (phase SlSt in figure 19). For small values of *twnd* (= *cwnd*), the RLC transmitter processes all segments in a burst before a new burst of segments arrives. The RLC buffer gets empty between two segment bursts.

As soon as the burst size (*cwnd*) exceeds a bandwidth-delay product, the RLC transmitter is no more capable to treat all segments in the burst before receiving the next new burst of segments. The new arrived TCP segments are added to the RLC buffer after currently stored segments from the previous burst. In our example, this event happens at time around t = 2s (see the detail in figure 19). The buffer does not get empty any more. The bandwidth-delay product *BD* can be computed according to:

$$BD = C * RTT_{TCP} = C * (RTT_{wireless} + RTT_{wired}) \qquad (6)$$

where C denotes the bandwidth of the radio link and RTT_{TCP} represents the TCP's RTT, i.e. the end-to-end round trip delay. In our example (C = 384 kbit/s and RTT_{wired} = 200 ms, $RTT_{wireless}$ = 60 ms), the bandwidth-delay product is BD = 12480 bytes or 19,2 IP packets (BD / *IPSz*, where *IPSz* is the IP packet size, i.e. 650 bytes).

Detailed analytical analyzes that describe progression of the congestion window size during the slow start soft state can be found in [LM97] and [AW02]. The second reference also derives transmission cycle for which the radio link reaches 100% utilization.

As soon as *twnd* reaches value of *ssthresh*, the soft state changes from the slow start to congestion avoidance soft state. In this soft state, the growth of *cwnd* slows down; *cwnd* increases linearly. The RLC buffer occupancy as well begins to grow linearly (phase CgAv1 in figure 19).

The *twnd* linearly increases till *twnd* becomes limited by the receiver buffer size (i.e. by *rwnd*). When *twnd* reaches the *rwnd*, it becomes constant (see expression 1 in section 3.2). Likewise, the RLC buffer occupancy stops increasing linearly and oscillates around a mean

value (phase CgAv2 in figure 19). In the figure, the mean value is about 23 SDUs. The oscillations around the mean value are due to various RTT_{wired} for every TCP segment.

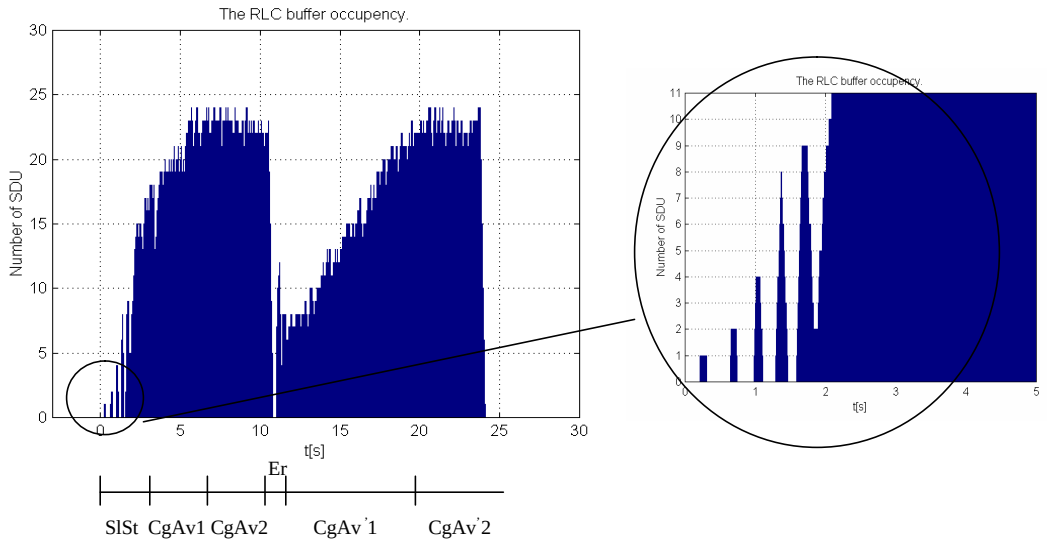


Figure 19. Number of SDUs in the RLC buffer as function of time; mean RTT_{wired} = 200 ms, BlER = 0%. One TCP segment loss is introduced in the wired network (at time around 10 s); SlSt: Slow Start soft state, CgAv: Congestion Avoidance soft state.

The congestion event, introduced by a segment loss in the wired network occurs at time around t = 10s (denoted as Er in figure 19). The TCP sender detects the lost by reception of three DupAcks. When receiving the third DupAck, the sender retransmits the lost segment and reduces its *ssthresh* and *cwnd* (expressions 3 and 5 in section 3.2.2). Subsequently, the TCP sender enters into the fast recovery state (section 3.2.2). In this soft state, the sender inflates its *cwnd* for every received DupAcks. Each DupAck indicates that a segment has been removed from the network and is cached at the TCP receiver buffer. The sender waits until a certain number of DupAck are received without transmitting any new segments. During this period, the RLC buffer occupancy dramatically decreases because: (i) there are no new delivered SDUs from TCP and (ii) the RLC entity continues to serve queued SDUs in the buffer.

Once the TCP sender receives a fresh Ack, it exits the fast recovery state and resumes the next transmission with the congestion avoidance soft state. The fresh Ack also acknowledges all segments sent between the lost segment and its retransmission version. The RLC buffer occupancy starts growing linearly (phase CgAv`1 in figure 19) till the *twnd* becomes again limited by *rwnd* (phase CgAv`2 in figure 19).

From the figure can be observed that if a dedicated transport channel is employed, the channel is poorly utilized during the TCP slow start soft state. Setting the initial *cwnd*, i.e. the pa-

rameter IW, to a higher value or using a higher value of MSS improve the radio link utilization in this early TCP phase ([AW02]).

Our analyses have not considered the mandatory establishing (three-way handshake) and closing TCP phases. These phases would further decrease the utilization of the dedicated channel. To avoid this, transport channels RACH, FACH, CPCH can be used for the establishing and closing TCP packets; for more about these transport channels see section 1.2.5.

Let's now look at the *wireless transmission delay* of SDU (TCP segment). In our studies, the wireless transmission delay of a SDU begins to be measured at the moment the SDU is placed in the RLC buffer. The measurement is stopped as soon as the RLC transmitter (in RNC) receives the Ack for this SDU from the UE entity (figure 20).

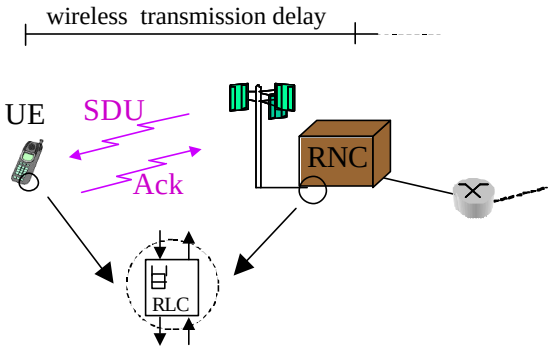


Figure 20. Measurement of the wireless transmission delay of a SDU.

A trace of the wireless transmission delay for each SDU is shown in figure 21. The SDU wireless transmission delay again reflects different TCP protocol soft states that have been discussed previously.

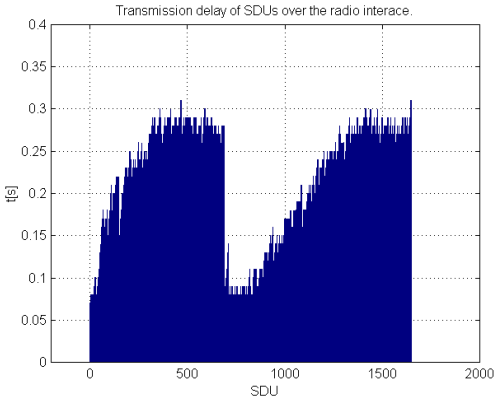


Figure 21. Wireless transmission delay for each SDU; mean RTT_{wired} = 200 ms, BlER=0%. As in figure 19, one TCP segment loss is introduced in the wired network (the segment has the same sequence number as in figure 19).

Scenario B - different wired network delays

This scenario investigates impact of different wired delays RTT_{wired} on the RLC buffer occupancy. There are considered three mean values m of RTT_{wired}; $m \in$ (50, 100, 200 ms). For all three values of RTT_{wired}, the standard deviation d is again set to 20 ms. The BlER on the radio interface is set to 10%. The RLC buffer size is again assumed to be infinite.

Traces of RLC buffer occupancies as function of time for different mean values m of RTT_{wired} are shown in figure 22. As in scenario A, there is introduced one congestion event in the wired network; the lost segment has the same sequence number as in the scenario A.

The RLC buffer occupancy again reflects the different TCP soft states that are discussed in the previous scenario. For smaller values of RTT_{wired}, the bandwidth delay product is lower and its value is reached faster by the TCP transmission window. Lower values of RTT_{wired} means faster emission of TCP segment bursts and thus providing the RLC entity with less time to process a segment burst before receiving a new burst. The event of adding a new TCP segment burst after the currently stored segments from the previous TCP burst in the RLC buffer occurs sooner (see the detail in figure 22). The RLC buffer occupancy grows faster for lower values of RTT_{wired}.

When changing from the TCP slow start to congestion avoidance protocol soft state, the RLC buffer occupancy is higher for smaller values of RTT_{wired}. This is due to the faster growth of the buffer occupancy in the TCP slow start soft phase.

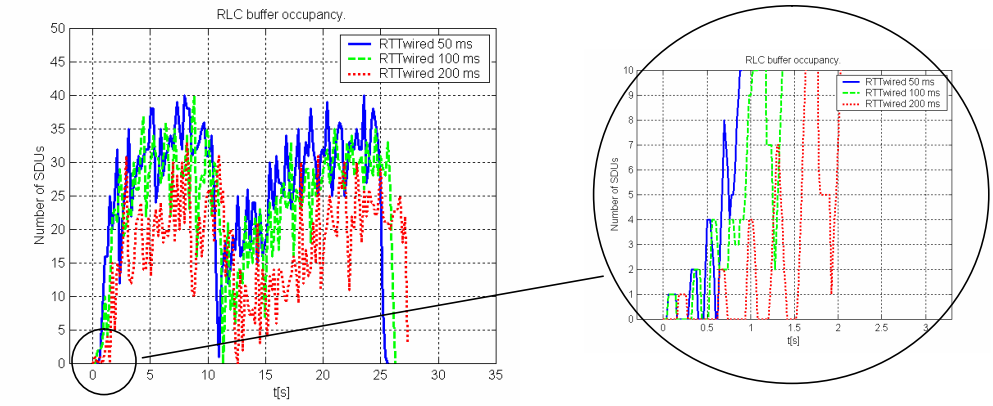


Figure 22. Number of SDUs in the RLC buffer as function of time for different values of mean RTT_{wired}; BlER = 10 %. The solid line is for a low value of RTT_{wired} (50 ms), whereas the dotted line is for a high value of RTT_{wired} (200 ms).
As in figure 19, one TCP segment loss is introduced in the wired network at time around 10s (the segment has the same sequence number as in figure 19).

Notice that on the one hand, the lower values of RTT_{wired} increase the RLC buffer occupancy; however on the other hand the utilization of the dedicated link during the TCP slow start phase is improved. In section 1.3, we have mentioned that proxy servers can decrease TCP RTTs. Besides shorter TCP RTT, the use of proxy servers also result in better utilization of dedicated channels.

Figure 23 illustrates how different values of RTT_{wired} affect the wireless transmission delay of SDUs. There are more stored SDUs in the RLC buffer for lower values of RTT_{wired}. Therefore, the wireless transmission delay is higher for lower values of RTT_{wired}.

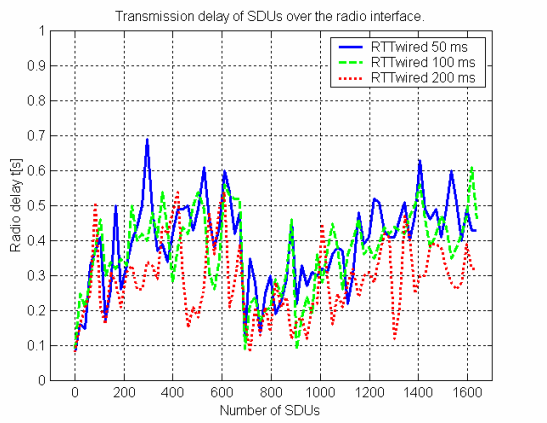


Figure 23. Wireless transmission delay for each SDU and for different mean values of RTT_{wired}; BlER=10 %. The solid line is again for a low value of RTT_{wired} (50 ms), whereas the dotted line is for a high value of RTT_{wired} (200 ms).
As in figure 19, one TCP segment loss is introduced in the wired network (the segment has the same sequence number as in figure 19).

Scenario C - different BlERs on the radio interface

Let's now analyze how different BlERs (Block Error Rates) on the radio interface affect the RLC buffer occupancy. The mean value m of RTT_{wired} is set to 100 ms (d = 20 ms). The buffer size is again infinite. As in scenarios A and B, there is introduced one congestion event in the wired network.

Curves of the RLC buffer occupancy as function of time for different BlERs on the radio interface (0%, 10% and 20%) are shown in figure 24. Different BlERs have different impact on the RLC level retransmission and thus on the TCP end-to-end delay. Each SDU needs a different amount of RLC retransmission attempts to get it correctly over the radio interface. The higher BlER is, the longer time correct transferred SDUs over the radio interface takes.

Wireless transmission delays for each SDU and for different BlERs are shown in figure 24. By comparing figures 22 and 24, we observe that higher values of BlER mainly increase the SDU wireless transmission delay then the RLC buffer occupancy.

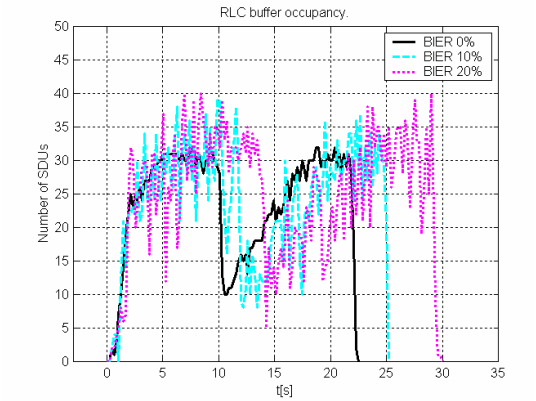


Figure 24. Number of SDUs in the RLC buffer as function of time for different BlERs on the radio interface; mean RTT_{wired} = 100 ms. As in figure 19, one TCP segment loss is introduced in the wired network (the segment has the same sequence number as in figure 19).

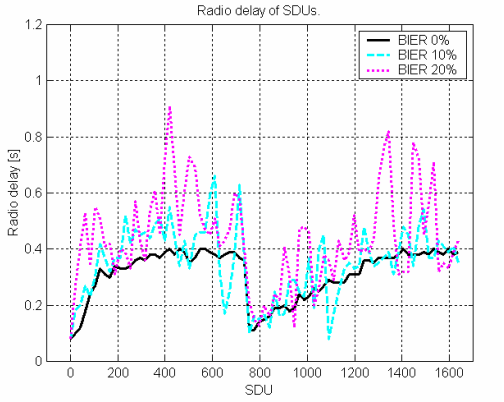


Figure 25. Wireless transmission delay for each SDU and for different BlERs; mean RTT_{wired} = 100 ms. As in figure 19, one TCP segment loss is introduced in the wired network (the segment has the same sequence number as in figure 19).

Scenario D - different RLC buffer sizes

This scenario analyzes the impact of different RLC buffer sizes on the TCP performance. Contrary to scenarios A, B and C, TCP segments are only lost due to the limited RLC buffer size, i.e. the wired part of the TCP connection is now error free. The buffer stores SDUs (TCP

segments) that arrive from the server. If a TCP segment arrives at the moment the RLC buffer is full, the TCP segment is dropped.

Neither the ICMP message Source Quench nor the ECN mechanism is used to inform the TCP transmitter about the segment discard or about the RLC buffer state. The RLC parameter MaxDAT is still set to 32. The mean value m of RTT_{wired} is set to 100 ms (d = 20 ms).

Figure 26 shows the TCP throughput versus BlER (on the radio interface) for different sizes of the RLC buffer; the size is indicated in a number of SDUs. The throughput is computed as a ratio of the size of transmitted file (1 Mbytes) to the whole transmission time. The value of the throughput is normalized to the wireless link bandwidth (i.e. 384 kbit/s). The upper bound of the TCP throughput is lower than 1 due to the protocol headers (TCP, IP, RLC).

The first curve in figure 26 (40 SDUs) shows a case where the RLC buffer size is sufficiently big that no TCP segment is dropped from the buffer during the whole file transfer. The decreasing performance of TCP with increasing BlERs is due to the retransmission delay at the RLC level.

The second curve (30 SDUs) is for a case where the RLC buffer size is not big enough to store all arrived TCP segments during the file transfer. For low BlERs, the buffer is able to store arrived SDUs. Nevertheless, for higher values of BlER, the buffer runs out of the buffer space and it starts occasionally dropping SDUs.

The last curve in the figure (20 SDUs) is for a case where the RLC buffer size is too small and SDUs are dropped even for very low BlERs. As BlER increases, there are more and more SDUs (TCP segments) that are dropped from the RLC buffer.

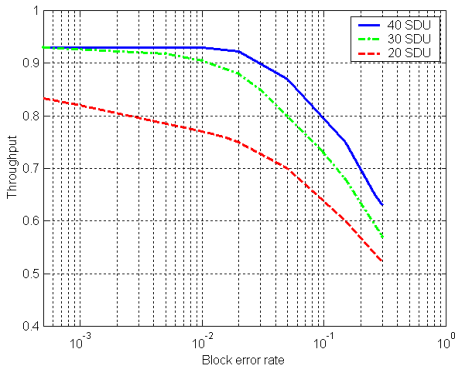


Figure 26. The TCP throughput as function of BlER for different RLC buffer sizes (40, 30 and 20 SDUs or IP packets), mean RTT_{wired} = 100 ms.
Contrary to scenarios A, B and C, there is no TCP segment loss in the wired network.

4.3.3 RLC buffer in a TCP connection path – conclusions

The second part of chapter 4 analyzes a TCP downlink data transfer via the UMTS radio. In our analyses, we study the buffer occupancy of AM RLC entity that manages transmissions of TCP data segments, i.e. the RLC entity located at the RNC entity.

We observe that the RLC buffer occupancy reflects the TCP soft states: slows start (exponential growth of the RLC buffer occupancy) and congestion avoidance (linear growth of the RLC buffer occupancy). Higher values of RTT_{wired} decrease: (i) the RLC buffer occupancy, (ii) the SDU wireless transmission delay (and thus the TCP RTT) and (iii) the utilization of the dedicated channel. Higher values of BlER increase the wireless transmission delay of SDUs and the necessary time to transfer the whole file.

In a case of the uplink TCP data transfer, the transmission of TCP data segments is controlled by the RLC entity that is located in the UE protocol stack. The RLC (UE) buffer occupancy is determined by a UE operating system and its internal TCP/RLC flow control besides the TCP soft states. This flow control is not standardized in 3GPP specifications and it entirely depends on the UE operating system.

Through the internal flow control between RLC and TCP, the number of TCP segments in the RLC buffer can be minimized. The buffer can store just TCP segments that the AM RLC entity can currently serve. After finishing serving a SDU, the RLC entity can ask the TCP sender to deliver the next TCP segment; the segment delivery is still provided with respect of the current TCP transmission window. This internal buffering optimization decreases memory requirements on the RLC buffer size. In addition, the increase of TCP RTT due to the RLC buffering (that we can see for the downlink case) is avoided.

4.4 Cooperation of RLC and TCP

A classical problem of TCP on a radio interface represents a misinterpretation of wireless and congestions losses. This problem is handled by different TCP extensions such as ELN (Explicit Loss Notification [BK98]), Loss predictor ([BV98]), etc. However, only a part of TCP extensions for wireless networks consider a radio link with an ARQ protocol at layer 2. In UMTS, there are no wireless losses a priori unless using the RLC SDU discard function. The RLC SDU discard delimitate the maximum RLC delay for each SDU (TCP segment). If setting the maximum RLC delay to high values, TCP spurious timeouts may occur. Two TCP extensions dealing with TCP spurious timeouts are called Eifel ([LK00]) or EBSN (Explicit Bad State Notification, [BK96]); the principle of these TCP extensions is explained in section 3.5.1. Neither Eifel nor EBSN covers RLC SDU discard issue.

We suggest to introduce between RLC and TCP a specific mechanism called Early Error Notification (EEN). The EBSN extension is based on the TCP agent in the base station and is

just proposed for TCP downlink transfers. The EEN mechanism is based on the RLC SDU discard function and is proposed for TCP uplink data transfers. Compared to Eifel, the EEN mechanism (and EBSN) tries to prevent triggering of TCP spurious timeouts.

In a case of an uplink data transfer, the TCP transmitting entity and the RLC entity are both located in the UE protocol stack. In such case, a direct communication can easily be implemented between the TCP and RLC transmitting entity. Section 4.4.1 details the proposed EEN mechanism for TCP uplink data transfers. Simulation experiments and their results are described in section 4.4.2.

In a case of a TCP downlink transmission, the TCP transmitting entity and the RLC entity are each located in different equipments. Therefore, the proposed EEN mechanism cannot be directly applied in the downlink case as described in section 4.4.1. A possibility of using EEN for downlink transmissions is discussed in section 4.4.3.

4.4.1 Early Error Notification (EEN) - uplink

This section analyzes two simulation scenarios. The first one analyzes a standard uplink TCP connection over RLC, i.e. the proposed EEN mechanism is not used (*TCP without the EEN mechanism*). The second scenario analyzes a case where the EEN mechanism is applied (*TCP with the EEN mechanism*). In both scenarios, the AM RLC entity is configured to use the SDU discard mode: *SDU discard after MaxDAT number of retransmissions* (explained in section 2.3).

TCP without the EEN mechanism

At the sender side (i.e. in UE), a TCP sender delivers segments to a lower layer as long as the TCP transmission window size (*twnd*) allows it. Through the internal TCP/RLC flow control, the number of stored SDUs (TCP segments) at the RLC buffer can be minimized. The RLC buffer could store only as many SDUs as the RLC entity can currently serve. By this way, there would be avoided an unnecessary queuing of many SDUs at the RLC level and thus the reserved memory for the AM RLC buffer could be minimized.

Erroneous Data PDUs are signaled by the RLC receiver to the RLC sender that provides its retransmission. A Data PDU can be (re)transmitted up to a maximum number of times. This number corresponds to the RLC parameter that is referred as MaxDAT (this parameter is introduced in section 2.3).

To a Data PDU is associated a RLC transmission counter that is represented by the state variable VT(DAT); see figure 9a in section 2.3. If a RLC transmission counter reaches the value of MaxDAT, the AM RLC entity discards the corresponding Data PDU from its buffer. Besides this RLC block, all blocks that constitute the same SDU (co-SDU blocks) are as well discarded see figure 9b in section 2.3. Before continuing the next data transmission, the RLC

sender has to inform the RLC receiver about all discarded Data PDUs. This notification is provided via the explicit signaling procedure that is described in more details in section 2.3.

At the TCP level, the TCP sender detects the discarded segment either by receipt of three DupAcks or by expiration of the TCP retransmission timer. The sender reacts to this internal segment loss by the same way as it would react to a congestion event in a distant router, i.e. by invoking its congestion avoidance mechanisms.

TCP with the EEN mechanism

Let's now turn our attention to the second scenario with the active EEN mechanism. Through the EEN mechanism, a "pseudo-loss" event occurring at the RLC level is reported to the TCP sender.

Until a RLC transmission counter, i.e. VT(DAT), does not reach the value of MaxDAT, the RLC processing in the UE entity follows the RLC processing described in the first scenario (*TCP without the EEN mechanism*). As soon as a VT(DAT) reaches the value of MaxDAT, the following procedure is initialized (we denote the procedure as EEN discard procedure).

The RLC entity sets the expired RLC transmission counter and transmission counters of all co-SDU blocks to zero. Neither the concerned Data PDU nor co-SDU blocks are discarded. The RLC sender just discards the SDU "virtually" and it continues with the transmission of the SDU over the radio interface. The RRC layer is informed about the virtual discard event, for this can be used the primitive CRLC_Status_Ind (for more see section 2.4.4). The explicit signaling procedure is not activated since no RLC blocks are discarded by the entity. The RLC virtual discard procedure could be considered as a new SDU discard mode; for example denoted as *Virtual SDU discard after MaxDAT number of retransmissions without explicit signaling*.

In addition, the RLC entity sends the TCP transmitter an EEN message containing a "virtual discard" warning. The TCP sender reacts to this message by updating the retransmission timer. The value by which is updated the TCP retransmission timer is carried in the EEN message, for example a parameter denoted as UpDateTimer. The value of the UpDateTimer parameter equals to MaxDAT * $RTT_{wireless}$, where $RTT_{wireless}$ is the Round Trip Time between the UE and the RNC entity (if no RLC retransmission occurs).

If a RLC transmission counter reaches again the value of MaxDAT, the EEN discard procedure is repeated. To prevent a RLC deadlock due to the virtual discard events, the number of repetitions of the EEN discard procedure per SDU is monitored. If a certain limit is reached, the RLC reset procedure is activated instead of the EEN discard procedure; for more about the RLC reset procedure see section 2.3.

In section 2.4.2, we have observed that an AM RLC entity disposes a primitive called RLC_AM_Data_Conf to inform the upper adjacent layer about a discard event. Thus, the EEN

mechanism can employ this already specified primitive for conveying the UpDateTimer parameter.

If a RLC entity has difficulties to transmit SDU(s) from the UE to RNC entity, we can expect similar transmission difficulties on the opposite link. The RLC entity located in the RNC entity may have problems due bad channel conditions to transmit over the radio interface a SDU that contains an Ack. In order not to lose SDUs containing TCP Acks, the RLC (RNC) entity employs similar virtual SDU discard procedure except the generation of the EEN message. Triggering of the message is disabled.

A question can arise: How to trigger the EEN message, if there is no Data PDU in the RLC (UE) buffer and the returning TCP Acks are "blocked" in the RNC entity due to the bad channel conditions? Such event may occur at the beginning of a TCP connection during the TCP slow start soft state; as we have seen in section 4.2 (figures 19 or 22). Or, if all TCP data segments have been sent (and transferred by the RLC entity over the radio interface) and the TCP sender just waits for Acks. This issue can be overcome by introducing at the RLC transmitting entity a new RLC timer; for example denoted as Timer_Virtual_Discard. This timer is activated if the following two conditions simultaneously become fulfilled: (i) the transmitting AM RLC buffer (in UE) is empty and (ii) the UE receives erroneous RLC blocks from the RNC entity. The reception of erroneous RLC blocks in the UE is an indication that the RNC entity has some data for the UE entity. If one of the previous two conditions becomes invalid, the timer is deactivated. When expiring, the timer Timer_Virtual_Discard is restarted and the EEN message is sent the TCP sender. The value of the timer Timer_Virtual_Discard can be set to MaxDAT * $RTT_{wireless}$. We propose to employ the timer Timer_Virtual_Discard as in UE as in RNC entity in order to unify the EEN mechanism at both sides. Notwithstanding, the EEN message is not triggered at the RNC side.

Similar mechanisms to the EEN mechanism

There can be considered few analogue mechanisms (or implementations) to the EEN mechanism.

The description of the EEN mechanism is provided for the SDU discard mode *SDU discard after MaxDAT number of retransmissions.* Instead of this discard mode, the RLC entity can be configured to use the SDU discard mode *Timer based discard, with explicit signaling*; this SDU discard mode is detailed in section 2.3. In such a case, the EEN discard procedure is triggered based on the expiration of the Timer_Discard timer. The time based virtual discard procedure could be seen again as a new RLC SDU discard mode: *Virtual timer based discard without explicit signaling*. The value of the UpDateTimer parameter directly equals to a value of the Timer_Discard timer.

In section 2.4.2, we have noticed that the upper layer may require the transmitting RLC entity to confirm the successful reception of SDU by the receiving RLC entity. This requirement is indicated through the CNF parameter (Confirmation Request), which is carried in the primitive RLC_AM_DATA_Req. The confirmation of a successful SDU transfer over the radio interface is provided via the Status parameter (carried in the primitive RLC_AM_DATA_Conf). Thus, the TCP sender may know for each transmitted segment a moment when the segment gets correctly over the radio interface. Knowing this instant, the initialization of the TCP retransmission timer can be postponed until the confirmation of successful transfer of the segment is obtained from the RLC entity.

4.4.2 Simulation and discussion - uplink

Experiments concerning the EEN mechanism are again realized on the UMTS test-bed (for the test-bed description see section 4.3.1). The simulation model and layer architecture correspond to the model that we use for the RLC buffer analyses (see figure 18 in section 4.3.2).

Parameter setting

The RTT_{wired} is set to 300 ms. The IP packets are delivered to the RNC entity (respectively the server) in the same order as they arrive from the server (respectively the RNC entity). The $RTT_{wireless}$ is set to 40 ms (or 4 radio frames), if no RLC retransmission occurs. Time processing of received data in the UE and RNC entities is considered to be zero second.

The TCP (Reno) transmission directly starts with data transfer, the three-way handshake and TCP closing phase are not simulated. The MSS (Maximum Segment Size) equals to 600 bytes. The initial *cwnd* (congestion window size), i.e. the parameter IW, is set to one MSS. The parameter Loss Window (LW) is also set to one MSS. The delay Ack function is deactivated, i.e. the TCP receiver acknowledges without delay every received segment.

The RLC entity operates in the acknowledged mode. The transmitting RLC entity provides the segmentation; if possible, the entity performs the concatenation. The reassembled SDUs are delivered the upper layer (i.e. the IP layer) in-sequence. The Data PDU size is 80 bytes. The RLC entity is configured to use the SDU discard mode: *SDU discard after MaxDAT number of transmissions*. The MaxDAT parameter is set to 4.

The RLC transmitter and receiver windows have the same size. Both windows are set to sufficiently large values that the RLC transmission is not stalled due to the window size (the RLC transmission window size = 2047 Data PDUs, which is the maximum value [3G25331]).

The AM entity uses an acknowledgment policy, where the transmitter explicitly (through the polling bit P) demands the receiver to send Ack/NAck reports. The transmitter sets the P bit up in one of Data PDUs carried in each radio frame. A radio frame can transmit up to six Data PDUs. This corresponds to the 384 kbit/s user data rate. An IP packet is transmitted at least in

two radio frames; one IP packet is segmented into 9 Data PDUs (650bytes / 80 bytes). The duration of a radio frame and TTI is set to 10 ms.

A dedicated transport channel is used for the whole file transfer. Handoffs are not considered in our experiments. The error process on the radio channel is supposed to be memory less. A transmission of Status PDUs carrying SuFi ACK, SuFi BITMAP, SuFi MRW and SuFi MRW_ACK is assumed to be error free. The parameter setting of the test-bed for the EEN experiments is summarized in table 8.

TCP (Reno)	Maximum Segment Size (MSS)	600 bytes
	Initial Window (IW)	MSS
	Lost Window (LW)	MSS
	Delay Ack	0 s
Wired network	RTT_{wired} (RNC-Server)	300 ms
	Errors	None
RLC (AM entity)	Size of a Data PDU	80 bytes
	Max. number of retransmissions (MaxDAT)	4
Wireless network	$RTT_{wireless}$ (UE-RNC)	40 ms (if no retransmission)
	Radio frame duration and TTI	10 ms
	Maximum number of Data PDUs in one radio frame	6 Data PDUs (…384 kbit/s)
	BlER (Block Error Rate)	see scenarios

Table 8. Parameter setting during the EEN experiments.

There are considered two scenarios: 1) without the EEN mechanism and 2) with the EEN mechanism. In both scenarios, there are introduced two periods during which the radio channel is in a "bad state" (denoted as Bad in figures 27 and 28). During a bad state period, the BlER is high (BlER = 20%), whereas it is negligible in a good state period. The duration of the bad state period is set to 600 ms. The first bad period starts at $t = 0{,}4$s and the second one at $t = 2{,}7$s.

To keep similar TCP soft states in both scenarios, the *ssthresh* (respectively *rwnd*) is set to a sufficiently high value that during the whole transmission the TCP sender remains in the slow start soft state; i.e. *twnd* is just function of *cwnd* (see expression 1 in section 3.2).

Scenario 1 - without the EEN mechanism

Figure 27 shows the sequence number trace of IP packets (TCP segments) for the case when the EEN mechanism is not utilized.

The first TCP segment is correctly transmitted and acknowledged at t = 0,4s. Next two segments are delivered to the RLC entity at instance the first period of high BlER starts. After a while, a RLC transmission counter, VT(DAT), reaches the value of the MaxDAT parameter. The concerned Data PDU and all co-SDU blocks are discarded. The discarded blocks correspond to the second TCP segment. The transmission of the next SDU (the third segment) is also unsuccessful and the SDU is as well discarded. The TCP sender is not aware of the (internal) discarded event at the RLC level.

The TCP sender detects the segment loss via expiration of the TCP retransmission timer at t = 0,9s. To decrease the "inactive" period while waiting for the expiration of the TCP retransmission timer, the initial value of RTO is set in both scenarios to 0,5s (instead of 3s). This modification allows us to better observe and compare the traces of the sequence number of TCP segments in both scenarios.

The TCP sender retransmits the second segments (arrow 1 in figure 27) and resumes the next transmission with the slow start soft state. The retransmitted segment is correctly transferred to the TCP receiver and is acknowledged at t = 1,3s. Thereupon, the TCP sender retransmits the third segment (arrow 2 in figure 27) and sends a next new segment. The transmission subsequently continues without any problem until the next period of high BlER.

The second period of high BlER starts at t = 2,7s. Segments *a-c* (segments are labeled by using alphabet letters, c.f. arrows *a-f* in figure 27) are correctly transferred by the RLC entity over the radio interface. A segment *d* is delayed due to RLC retransmissions until it is correctly transferred. The transmission of segments *e, f* is unsuccessful and they are discarded at the RLC level.

The received TCP Acks for segments *a-c* are discarded by RLC in the RNC entity (due to unsuccessful transmissions). The TCP retransmission timer expires at t = 3,2s. The TCP sender retransmits the first unacknowledged segment in the transmission window, i.e. the segment labeled as *a* (arrow 3 in figure 27). The next transmission is resumed with the slow start soft state. Short time after the retransmission of the segment *a*, the TCP sender receives the Ack for segment *d*. This Ack provokes retransmission of segments *e, f* (arrows 4 and 5 in figure 27). After receiving the next fresh Ack (at t = 3,6s), the TCP sender goes on with transmission of next segments. The rest of transmission continues without any specific problem.

From the first scenario, we can observe that during the first bad state period, the TCP sender is not aware of the losses occurring at the RLC level. After a while, the TCP retransmission timer expires due to discarded segments at the RLC level: no protocol's cooperation is done to face those losses.

One-way how to reduce the number of discarded SDUs at the RLC level is to increase a value of the parameter MaxDAT. Studies investigating the TCP throughput versus BlER for different values of MaxDAT can be found in [LV01]. However, a high value of the MaxDAT parameter can considerably delay some TCP segments on the radio interface. During the second period of high BlER, the segment *d* and its Ack are so delayed on the radio interface that the TCP sender receives the Ack too late and a TCP spurious timeout occurs. The TCP sender unnecessary retransmits a segment and resumes the transmission with the slow start soft state.

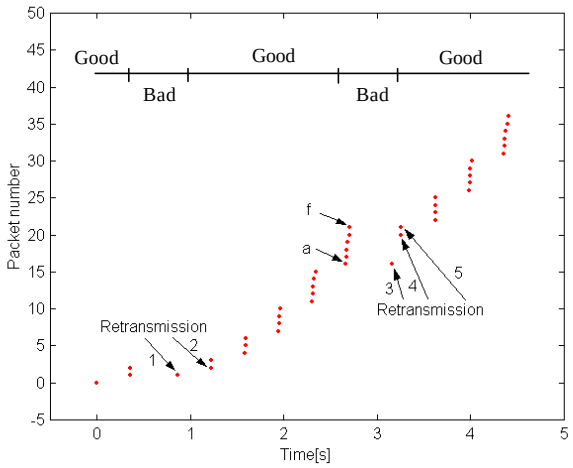


Figure 27. The sequence number of IP packets (TCP segments) as function of time; the EEN mechanism is not activated. The figure indicates the good and bad state periods on the radio interface.

Scenario 2 - with the EEN mechanism

Let's now analyze the second experiment, where the EEN mechanism is active. Beginnings and duration of the bad state periods are the same as in scenario 1. Figure 28 shows the sequence number of IP packets (TCP segments) as function of time during the similar transmission scenario as in the previous one.

Likewise in the first scenario, the first TCP segment is transferred over the radio interface without any problem. The second and third TCP segments are delivered to the RLC entity at the moment the first period of high BlER starts. During the bad state period a RLC transmission counter reaches the value of MaxDAT. The SDU discard mode *Virtual SDU discard after MaxDAT number of retransmissions without explicit signaling* is initialized and a virtual discard warning is sent the TCP sender. The TCP retransmission timer is incremented by the indicated value in the EEN message. The second TCP segment is acknowledged at t = 1s. Thereupon, two new segments are delivered to the RLC level. After receiving the TCP Ack for the

third segment (at t = 1,2s), two new TCP segments are sent. The transmission subsequently continues without any problem until the next period of high BlER.

The second period of high BlER starts at t = 2,7s. Within this period, the TCP retransmission timer is again updated according to the value indicating in the EEN message. This update prevents the expiration of the TCP retransmission timer and no segment is uselessly retransmitted by the TCP sender.

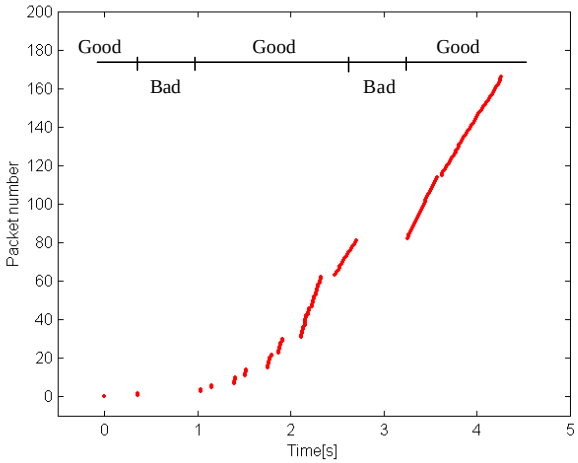


Figure 28. The sequence number of IP packets (TCP segments) as function of time; the EEN mechanism is activated. The figure indicates the good and bad state periods on the radio interface.

4.4.3 The EEN mechanism - downlink

The EEN mechanism takes advantage of the fact that the transmitting entities RLC and TCP are located in the same equipment. In such situation, the RLC entity can easily inform the TCP transmitter about virtual discard events at the RLC level. This is not a case of a TCP downlink data transfer; the TCP transmitter is located in a server (or proxy server), whereas the RLC entity is located in a RNC entity. The "direct" interlayer communication between RLC and TCP cannot be applied for the downlink case. Thus, a question can arise: Can the EEN mechanism also be employed for TCP downlink data transfers? The response to this question is yes. This section looks at EEN modifications that have to be introduced in order to use it for TCP downlink data transfers.

Likewise in the TCP downlink data transfers, in the uplink case the EEN mechanism is activated at both RNC and UE to prevent discards of TCP segments containing data and Acks. We firstly discuss modific ations of the EEN mechanism at the UE side and then at the RNC side.

UE side

At the UE side, the EEN mechanism closely follows the EEN discard procedure in the UE that is described in section 4.4.1. In the UE, the RLC transmitting entity handles transmission of segments containing TCP Acks. As soon as a RLC transmission counter reaches the value of MaxDAT, the virtual SDU discard is initialized. The TCP (UE) entity represents the TCP receiver. Since the TCP retransmission timer is managed by the TCP sender, the EEN message is not necessary to trigger and the RLC timer Timer_Virtual_Discard (introduced in section 4.4.1) does not have to be used. Notwithstanding, in order to unify the EEN mechanism at the UE side for both directions of TCP data transfers (uplink, downlink), we propose to trigger the EEN message and to use the RLC timer even for the TCP downlink transfer. In the downlink case, the TCP receiver simply does not react to the EEN messages; the UpDateTimer parameter can be set to 0. This "modification" represents the only difference in comparison with the uplink case.

To prevent a deadlock at the RLC level, the number of repetitions of the EEN discard procedure is monitored; as it is done for uplink data transfers. If a certain threshold is reached, the RLC reset procedure is initiated.

RNC side

Let's now analyze the RNC side. The RLC transmitting entity in RNC manages transmission of TCP data segments. Compared to the uplink case, the EEN mechanism in the RNC entity is slightly modified in the downlink case.

If a RLC transmission counter reaches the value of MaxDAT, the EEN discard procedure is initialized and the EEN message is sent the TCP sender; in the uplink case this message is not triggered. The message is also triggered whenever the RLC timer Timer_Virtual_Discard expires. Contrary to the uplink case, the RLC timer has to be now employed.

For TCP downlink data transfers, the direct interlayer communication between TCP and RLC cannot be used. We propose to replace the "direct" communication by an "indirect" communication by means of ICMP (the protocol is discussed in section 4.2.1). We suggest introducing a new ICMP message denoted as Virtual Discard. Whenever, RLC (RNC) entity triggers the EEN message, the message is enveloped into the ICMP message Virtual Discard and forwarded to the TCP sender (server). The IP address can be determined from the concerned SDU (IP packet). Upon receipt the ICMP message Virtual Discard, the TCP transmitter updates its retransmission timer. An advantage of using ICMP is that this protocol is defined as standard in IP networks. A disadvantage represents the necessity to modify the TCP and ICMP codes to deal with the new ICMP message. The necessary modifications can be incrementally deployed in networks.

Notice that the TCP retransmission timer update approach is also employed in the TCP extension EBSN (Explicit Bad State Notification), which is depicted in section 3.5.1. Compared to EEN, the EBSN mechanism is based on a TCP agent and the TCP retransmission timer update information is carried via a new TCP segment.

An objection might be made that the EEN message is conveyed by the IP packet and there is no guarantee that the message reach the TCP sender. This issue can partly be overcome by simply transmitting the ICMP message several times. At the server side, either the IP layer or the TCP sender itself would handle several copies of the ICMP message. Notice that a seemingly better solution corresponding to the EBSN mechanisms would not help either; i.e. using a TCP agent and utilize a TCP segment for delivery of the EEN message. The introduced TCP segment (carrying the EEN message) could still be lost on its way from the RNC to the server since no reliable mechanism is provided between them.

Not delivering the ICMP message Virtual Discard at the server is not necessary a drawback or an unfavorable effect of the ICMP proposition. Not receiving the Virtual Discard message can simply indicate congestion between the RNC entity and the server. Even the ordinary TCP segment(s) would get lost and the TCP sender would reduce its transmission rate anyway.

Remark: In case of TCP downlink data transfers, any sort of mechanism informing the TCP sender to update its retransmission timer has to be implemented in a RNC entity. An implementation of such mechanism in a UE entity goes with risk of blocking the notification message in the UE entity due to high BlER on the radio interface.

4.4.4 Cooperation of RLC and TCP – conclusions

The section 4.4 proposes to introduce between TCP and RLC (AM entity) a specific mechanism called EEN (Early Error Notification). Through this mechanism, the TCP sender is notified about a virtual SDU discarded event occurring at the RLC level.

If a virtual event occurs, the RLC entity sends the corresponding TCP transmitting entity a virtual discard warning in the form of EEN message. The TCP transmitter reacts to this message by updating the TCP retransmission timer. The value by which is updated the retransmission timer is carried in the EEN message. The TCP segment is not necessary retransmitted since it is still stored in the RLC buffer; the TCP segment is just virtually discarded by RLC. Triggering of TCP spurious timeouts due to a long period of high BlER on the radio interface is avoided. The EEN mechanism implicitly helps a TCP sender to differentiate a congestion and wireless loss; by discarding SDU virtually, segment losses detected by the TCP sender are congestion losses.

The EEN mechanism can be employed as for TCP uplink data transfers as for TCP downlink data transfers. In the uplink case, the mechanism takes advantage that the RLC entity and the corresponding TCP transmitting entity are situated in the same equipment (i.e. in the

UE). Thus, the protocol entities can easily communicated between them. In the downlink case, the “direct” interlayer communication is replaced by an “indirect” communication. The EEN message is transported to the distant TCP transmitting entity (server) through a new ICMP message denoted as Virtual Discard.

The EEN mechanism is based on the RLC SDU discard function. The AM entity can be configured to utilize either the SDU discard mode *SDU discard after MaxDAT number of retransmission with explicit signaling* or the SDU discard mode *Timer based discard, with explicit signaling*. The EEN mechanism requires slightly modified the previous two modes. The modified discard modes could be considered as two new SDU discard modes denoted as *Virtual SDU discard after MaxDAT number of retransmissions without explicit signaling* respectively *Virtual timer based discard, without explicit signaling*. Compared to the original SDU modes, the new modes discard SDUs virtually; the explicit signaling procedure described in section 4.4.1 is not activated.

Chapter 5
High Speed Downlink Packet Access in UMTS

5.1 Introduction

To enhance the packet data performance on the UMTS radio interface, a new concept called High Speed Downlink Packet Access (HSDPA, see e.g., [3G25848], [3G25855] or [PP01]) has been introduced in Release 5 of 3GPP specifications (in 2002). HSDPA includes several enhanced technique such as fast link adaptation, hybrid ARQ (HARQ), higher order modulation (e.g., 16-QAM), fast scheduling of users, etc. The HSDPA enhancements enrich the adaptation of downlink radio bearers in UMTS and allow reaching on the air interface data rates up to 10 Mbit/s ([H02]) within a 5 MHz bandwidth. This data rate is significantly higher than the maximum previous rate (i.e. 2 Mbit/s, Release 4).

When updating the UMTS radio interface with HSDPA, a hybrid ARQ mechanism is introduced, bellow RLC, at the physical/MAC-hs layers. The ARQ of MAC-hs operates between a UE entity and the Node B. This chapter analyzes MAC-hs protocol features. Besides them, issues concerning the HSDPA link adaptation procedure and HSDPA allocation mode are studied. The chapter is organized as follows.

The HSDPA architecture (new channels, layer structure, etc.) and protocol mechanisms are described in section 5.2.

The HSDPA allocation mode makes possible to reallocate radio resources very fast among the users; basic facts about the mode is depicted in section 5.3.1. However, services such as streaming services do not need such dynamic allocation of HSDPA resources. Section 5.3.2 discusses periodic allocation and its impact on the dynamic allocation. Simulation experiments concerning the dynamic and periodic allocation are described in section 5.3.3. The HSDPA allocation studies are concluded in section 5.3.4.

Contrary to the EGPRS system, the HSDPA link adaptation procedure does not consider a possibility of modifying the MAC-hs PDU size during different retransmissions attempts. To this issue is dedicated the third part of chapter 5 (section 5.4). The standard HSDPA link adaptation procedure is described in section 5.4.1. Section 5.4.2 details modification that we propose to introduce in the link adaptation procedure. Section 5.4.3 depicts a simulation environment and parameter setting. The performance of the standard and proposed schemes is compared in section 5.4.4. The HSDPA link adaptation conclusions are presented in the last section 5.4.5.

5.2 Architecture of HSDPA

5.2.1 Impact of HSDPA on the radio protocol layers

The HSDPA enhancements rely on a rapid adaptation of transmission parameters to the instantaneous channel conditions. Therefore, they are implemented as close as possible to the air interface, i.e. in Node B. The assignment of radio resources (i.e. scheduling) and HARQ (Hybrid ARQ) functionalities are implemented in a new MAC (Medium Access Control) entity called MAC-hs (MAC-high speed, [3G25321]). The new MAC-hs entity is located in Node B.

The HSDPA functionalities are implemented at the physical and MAC-hs layers. Layers situated above the MAC-hs layer (i.e. MAC-d, RLC, PDCP) are not modified; these layers remain unchanged from the previous Release 4 ([PP01]). The layer upgrade when introducing HSDPA on the UMTS radio interface is shown in figure 29 (the lower Node B in the figure).

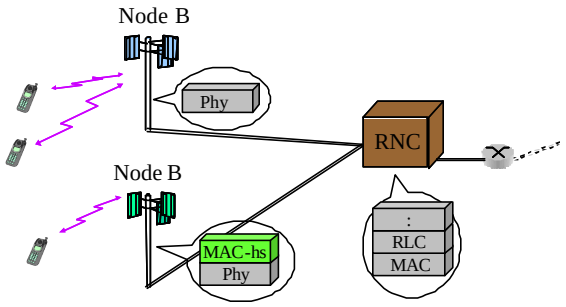


Figure 29. Layer modification when upgrading a Node B with HSDPA functionalities (the lower Node B in the figure).

5.2.2 New channels introduced in HSDPA

The HSDPA mode brings several new channels. The utilized channel abbreviations in 3GPP specifications are sometime awkward. In order to facilitate the decoding of channel abbreviations, our work slightly modifies the 3GPP channel abbreviations. A 3GPP channel abbreviation is mentioned when initiating the channel abbreviation.

Transport channel

A new downlink transport channel is introduced between the MAC-hs and physical layer. The channel is called High Speed Downlink Shared Channel (HS-DoShCH, or HS-DSCH in 3GPP specifications). As the name indicates, the transport channel is shared among several HSDPA users; a brief comparison of HS-DoShCH (Release 5) and DoSHCH (Release 4) is provided in section 1.2.5. The MAC-hs scheduler reallocates radio resources (i.e. channelization codes) with a period called HS-DoShCH TTI (Transmission Time Interval). For the FDD

mode, HS-DoShCH TTI is set to 2 ms ([3G25308]). In the rest of this chapter, we simply call HS-DoShCH TTI as TTI since there is no confusion with the conventional duration of TTI in UMTS (which can be 10, 20, 40, or 80 ms). Several users can be multiplexed within one TTI ([HP00]). An example of sharing the HSDPA code resource is shown in figure 30.

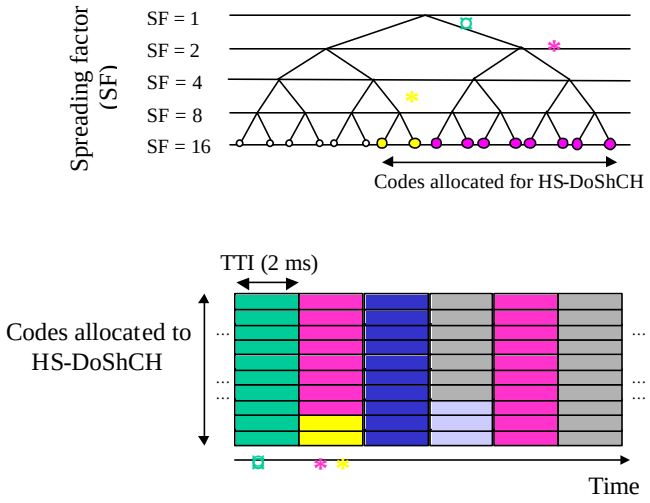


Figure 30. An example of sharing of the HSDPA code resource ([HP00]).

Physical channels

At the physical level, data of HS-DoShCH (i.e. MAC-hs PDUs) are mapped into a frame structure of a physical channel called HS-Physical Downlink Shared Channel (HS-PDoShCH, or HS-PDSCH in 3GPP specifications). Three consecutive slots (T-slot) in the HS-PDoShCH frame structure represent a basic radio "unit" for traffic. The duration of a T-slot, 3*(10/15) ms, corresponds to the duration of a TTI (2 ms).

One HS-PDoShCH corresponds to one channelization code (with a fixed spreading factor SF = 16, [3G25211]). There can be utilized up to 15 channelization codes ([3G25306]). This means that up to 15 physical data channels (HS-PDoShCHs) can be assigned in one T-slot.

Besides the data channel, HSDPA also introduces two new physical control channels that convey signaling information (downlink and uplink signaling). The downlink signaling informs a UE how to decode data on the associated HS-PdoShCHs. Signaling information is carried by a downlink physical channel called HS-Shared Control Channel (HS-ShCoCH, or HS-SCCH in 3G specifications). A downlink-signaling message includes user id, modulation and coding information, transport format, HARQ information ([3G25212]). The signaling message has to be received before data; the transmission of HS-ShCoCH precedes HS-PDoShCH by 1,33 ms (or 2 slots, [3G25211]).

An HSDPA cell disposes of several HS-ShCoCH sets. One set consists of four HS-ShCoCHs. When entering into an HSDPA cell, a UE is assigned one of these sets. In other words, a UE has the capability to simultaneously monitor four HS-ShCoCHs. In a case of transmission to the same UE in the consecutive TTIs, the same HS-ShCoCH is used.

The uplink signaling is carried by an uplink physical channel called HS-Dedicated Physical Control Channel (HS-DePCoCH, or HS-DPCCH in 3GPP specifications). The feedback information consists of HARQ Ack/NAck and Channel Quality Indication (CQI, [3G25212]). Notice that the HARQ process identification (HARQ Id) is not explicitly included in the feedback signaling. There is a predefined delay between the transmission of data and the reception of the corresponding Ack (or NAck). By this way, the HARQ Id relative to an Ack (or NAck) is unequivocally identified by the Node B. Transmission of the feedback information can be repeated over *m* consecutive HS-DePCoCH frames (for more details see reference [3G25214]). The new HSDPA physical channels are illustrated in figure 31.

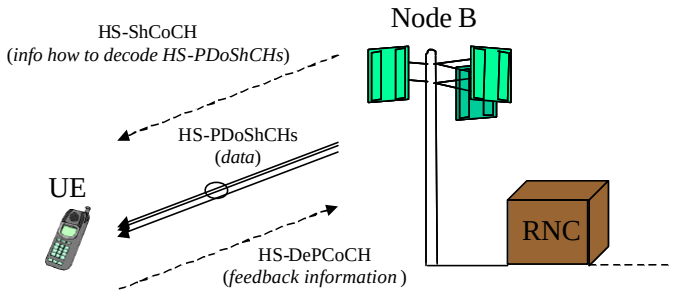


Figure 31. New physical channels that are introduced in HSDPA.

5.2.3 MAC-hs entity

The MAC-hs layer (in the RNC entity) is a composition of several MAC-hs entities. There are as many MAC-hs entities as many active HSDPA users are in the cell. This section describes MAC-hs functionalities.

In our studies and simulation experiments, we look at the HSDPA mode from the point of the view of protocol mechanisms. Our studies do not analyze different types of HARQ schemes (HARQ type I, HARQ type II, etc.) and their performance. Such type of studies can be found in [FP01], [DK02b], [MA01] or [LC02].

HARQ instances

Retransmissions of MAC-hs PDUs (denoted as d-Blocks) between a Node B and a UE are ensured by a HARQ mechanism. The mechanism is based on the simplest ARQ method: Stop and Wait (S&W). As it is well known, advantages of S&W are the minimum overhead and the minimum memory requirements since the protocol handles just one block at time. However,

Acks are not instantaneously received. After each transmission, the transmitter must wait for the corresponding Ack (respectively NAck) before transmitting the next block (respectively re-transmits the currently served block). Thus, the radio channel remains idle and the system capacity becomes wasted in the conventional S&W strategy.

To overcome the S&W drawback, HSDPA employs an approach where several HARQ instances (or processes) can simultaneously be activated. One HARQ instance corresponds to one S&W protocol instance. Hereby, one S&W instance can transmit data on the downlink while another S&W instance is waiting for an Ack on the uplink. Up to 8 independent HARQ instances can simultaneously be activated per UE ([3G25331]), i.e. up to 8 d-Blocks (MAC-hs PDUs) can be handled at the same time.

Priority handling

Since each user can activate several radio bearers with different priorities, HSDPA makes possible to set up to 8 priority queues per UE (or MAC-hs entity). Figure 32 shows an example where two priority queues are activated per UEId = 1; both priority queues share the pot of 8 HARQ processes. Within a T-slot (three slots), the MAC-hs scheduler specifies a user (UE Id) and the user priority queue that will be served. Information about the user queue, a flag called QId (3 bits), is carried in the MAC-hs header. The UE Id is carried in HS-ShCoCH. According the QId flag, the receiver puts data into the appropriate reordering received queue (see reordering queue distribution in figure 33).

HARQ processing

Each time a new d-Block (i.e. MAC-hs PDU) is transmitted, the MAC-hs scheduler determines a priority queue and a suitable payload size *L* of this d-Block. A d-Block consists of one or several RLC blocks. The d-Block is assigned a Transmission Sequence Number (TSN, modulo 64) and one of the free HARQ Ids (3 bits, 8 values). A priority queue manages TSN independently to other queue of a given user. In our example (figure 32), two priority queues are active per the user thus two TSNs have to be managed. The HARQ Id is carried in the physical downlink control channel (i.e. HS-ShCoCH), whereas TSN is carried in the header of MAC-hs PDU. The HARQ process controls transmission and retransmissions of the d-Block.

Through TSN, the MAC-hs layer provides two functions: (i) in-sequence data delivery to the upper layer and (ii) “flow control”. One of tasks of MAC-hs is to deliver received data to the upper layer in-sequence. The sorting of received d-Blocks according to TSNs is realized in a MAC-hs reordering entity (see figure 33). The MAC-hs scheduler may only send d-Blocks with TSN that lie within the MAC-hs transmitter window. The maximum receiver window (and transmitter window) size is 32. Thus, though using the ARQ method S&W, the HSDPA retransmission scheme behaves as if Selective Repeat would be employed.

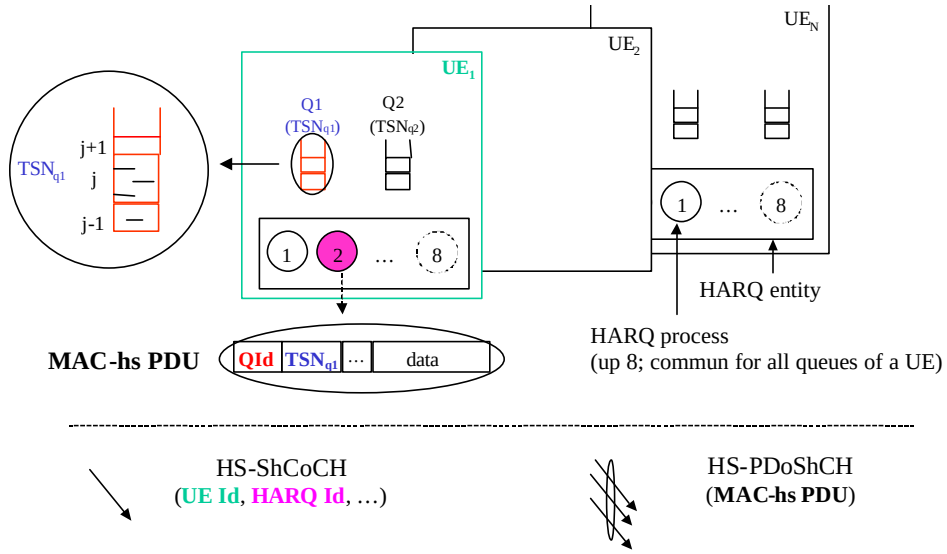


Figure 32. Transmitting MAC-hs entities (in Node B) and distribution of information in the downlink physical control (HS-ShCoCH) and data (HS-PDoShCH) channels. The figure illustrates an example with two active priority queues. The MAC-hs scheduler selects in the given three slots UEId = 1, QId = 1 and HARQId = 2.

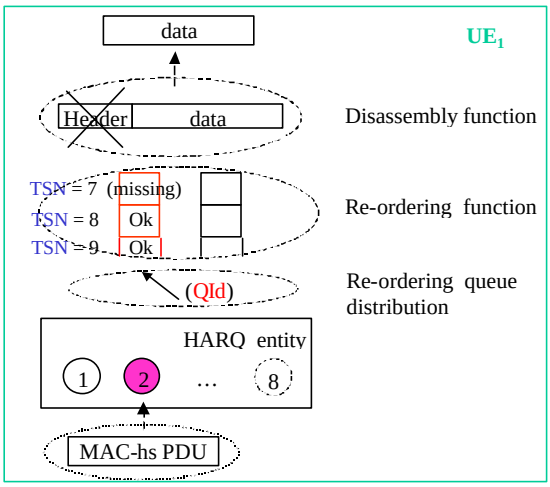


Figure 33. A receiving MAC-hs entity (in UE) and their different entity/function (as in the previous figure UEId = 1, QId = 1 and HARQId = 2).

To prevent a deadlock due to a missing d-Block(s) in the reordering queue, the receiving entity manages a timer called T1. There is handled one timer per queue. The timer is started when the reordering entity receives a d-Block (e.g., with TSN = *x*, d-Block$_x$) that cannot be delivered to the upper layer due to previous missing d-Block(s) in the MAC-hs receiver window. If the timer is already active for other d-Block, the timer is not reactivated. The timer is stopped at

the moment the d-Block$_x$ can be delivered to the upper layer. If the timer expires, the d-Block$_x$ (and all correctly received d-Blocks up to d-Block$_x$) is delivered to the upper layer and the MAC-hs receiver window is advanced. More details about the management of this timer can be found in [3G25321]. Studies of the timer T1 and the MAC-hs window are provided in [MR03].

Figure 32 shows transmitting MAC-hs entities (in Node B). This figure also illustrates the distribution of information in downlink physical control (HS-ShCoCH) and data (HS-PDoShCH) channels. In our example, the MAC-hs scheduler selects in the given T-slot a user with UEId = 1 and its HARQ process with HARQId = 2; the served data is from a priority queue with QId = 1. The corresponding receiving MAC-hs entity (in UE) is shown in figure 33 (UEId = 1, QId = 1 and HARQId = 2).

The MAC-hs layer (entity) can be regarded as a layer composing of two sub-layers: upper and lower one. The upper sub-layer manages flow control, the renumeration (modulo 64) and reassembling of RLC PDUs with fixed size into a MAC-hs PDU with variable size. Notice that the last functionality is the contrary that the RLC layer does; RLC segments variable data units and forms blocks with fix size. The lower sub-layer handles functionalities concerning the HARQ mechanism.

Management of the S&W flag

An S&W protocol instance of HSDPA manages an S&W flag. In HSDPA, this flag is called New Data Identifier (NDI, modulo 2, [3G25321]) and is carried in HS-ShCoCH. The NDI$_x$ flag is incremented each time a new d-Block is assigned to the HARQ$_x$ process (with HARQ Id = x). If the HARQ$_x$ receiver receives an NDI$_x$, which value differs to the previous one, any old data in the receiver's HARQ$_x$ buffer is automatically replaced by a new context.

In certain unfavorable situations, the HARQ receiver can misinterpret the value of NDI. To illustrate such event let's suppose a transmission scenario in figure 34. The scenario assumes that just one HARQ process (with HARQ Id = x) is activated per the user. The figure shows values of NDI at the transmitter side (Node B) and the last stored value of NDI at the receiver side (UE). Furthermore, TSN of the currently served d-Block at the transmitter side (respectively the "assumed" TSN of the served d-Block at the receiver side) is indicated. Since TSN is carried in the header of d-Blocks, the receiving HARQ entity does not know (see) TSN.

In our scenario, the HARQ$_x$ receiver incorrectly decodes data, therefore the receiver sends NAck. This NAck gets lost on its way to the transmitter. Since no feedback information (Ack/NAck) is received at the transmitter side within a specific time, the transmitter assumes that data has been lost and retransmits it. This time, the receiver correctly decodes data. Decoded data, i.e. the d-Block$_i$ (with TSN = i) is delivered into the MAC-hs reordering entity. The HARQ$_x$ receiver sends an Ack, which gets lost. After the *n-th* unsuccessful retransmission at-

tempt, the d-Block$_i$ is discarded and the HARQ$_x$ transmitter is assigned a new d-Block$_{i+1}$. The NDI$_x$ value of the HARQ$_x$ process in Node B is incremented (NDI$_x$= 1).

The HARQ$_x$ transmitter tries again several times to get the new d-Block$_{i+1}$ over the radio interface but without any success. Due to bad channel conditions, data does not even reach the HARQ$_x$ receiver; e.g., due to the incorrect receptions of the downlink signaling channel. Thus, the HARQ$_x$ receiver still stores the old value of NDI$_x$ (NDI$_x$= 0). After the *n-th* unsuccessful retransmission, the d-Block$_{i+1}$ is replaced by a new one (d-Block$_{i+2}$). The value of NDI$_x$ is incremented (NDI$_x$= 0).

The radio condition improves and the HARQ$_x$ receiver correctly receives and decodes the d-Block$_{i+2}$. Since the NDI value in HS-ShCoCH (NDI$_x$= 0) corresponds to the NDI value stored at the HARQ$_x$ receiver side (NDI$_x$= 0), the HARQ$_x$ receiver assumes retransmission of the old d-Block (d-Block$_i$). Supposing that the new data is the old ones, the receiver resends the Ack. The HARQ$_x$ transmitter considers this Ack as the notification of correct transfer of the d-Block$_{i+2}$ and the transfer of d-Block$_{i+2}$ is terminated. Hereby, the d-Block with TSN$_{i+2}$ gets lost. If required, an upper layer (such as RLC or TCP) provides retransmissions of missing data.

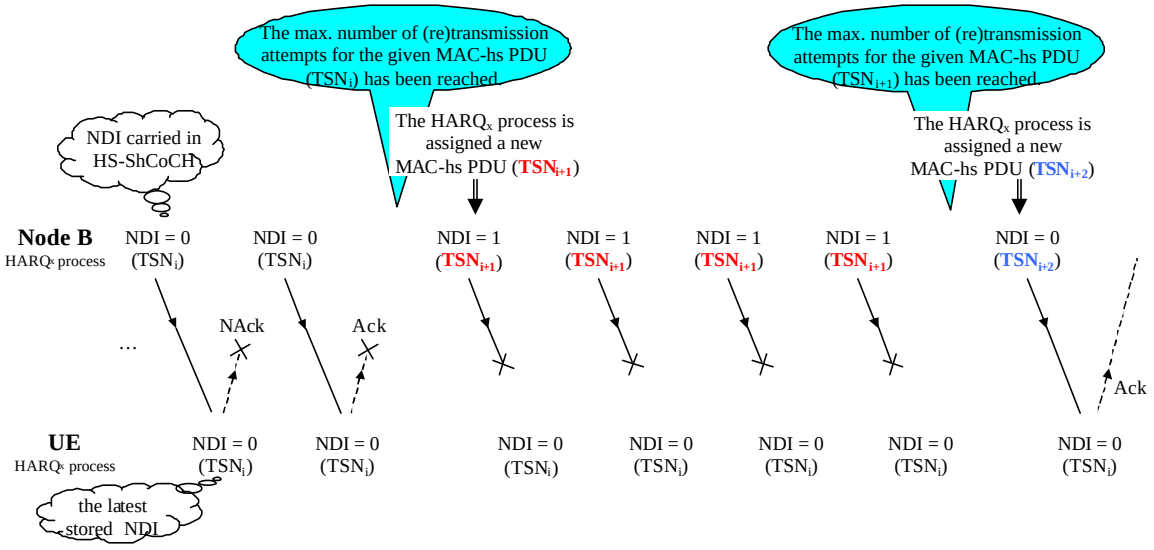


Figure 34. An example of transmission scenario during which the HARQ receiver (in UE) misinterprets the NDI flag.

To reduce the risk of NDI misinterpretation at the receiver side, the following two solutions can be considered. The first one would be to assign a new d-Block to the HARQ process only if the feedback-signaling message is received. In such case, the HARQ transmitter can suppose that the receiver correctly register the last valid NDI value of the given HARQ process.

The second solution could be to deliver a d-Block to the MAC-hs reordering entity every time data is correctly decoded by the HARQ receiving process. Hereby, some d-Blocks could be delivered to the reordering entity several times. Duplicated d-Blocks would manage the MAC-hs reordering entity.

5.3 Study of the HSDPA allocation mode

This section focuses on the HSDPA allocation mode. We look at the flexibility of the allocation mode and its usefulness for a certain type of services. The aim of this study is not to compare performance of different scheduling algorithms. Such types of analyses can be found for example in [LG01], [PP01], [KF02a] or [WJ02].

5.3.1 Basic features of the HSDPA allocation mode

The HSDPA allocation is very flexible due to the reduction of the basic allocation period from 10 ms (Release 4) to just 2 ms (one T-slot). HSDPA users are assigned T-slots according to a scheduling algorithm that is implemented at the MAC-hs layer. Short duration of the allocation period allows the scheduler to reallocate shared radio resources very quickly. The scheduler can assign in a T-slot up to 15 channelization codes, which may be shared either among several UEs or they are all assigned to just one UE.

A fast reallocation of resources produces a fast variation of scheduled UEs in time. Information about UE(s) that is allocated resources in a given T-slot is carried in HS-ShCoCH(s). This means that the UE has to firstly correctly decode the downlink-signaling message in order to correctly decode data on the physical data channels (HS-PDoShCHs). If a UE misses or incorrectly decodes the signaling message, the UE misses as well as data on HS-PDoShCHs. To minimize HSDPA losses, it is very important to design a good HS-ShCoCH coding scheme. A performance comparison of different coding schemes for HS-ShCoCH is done in [DK02a].

UE's capability parameters

When allocating radio resources, the MAC-hs scheduler has to take into account, among others, UE capability parameters. We focus on two of them. The first one determines the maximum number of channelization codes that a UE can receive in one T-slot, i.e. how many physical channels the UE can process at one T-slot. The second UE's capability parameter is called minimum inter-TTI interval (denoted as MinInter-TTI). The MinInter-TTI parameter specifies the minimum interval between the beginning of a TTI and the beginning of the next TTI that a UE can support. The MinInter-TTI parameter ranges from 1 up to 3 T-slots (or from 2 up to 6 ms, [3G25306]). Other UE's capability parameters can be found in [3G25306]. This reference also classifies UEs into 12 categories according to their capability features.

An example of impact of the UE's capabilities parameters on the assignment of T-slots

To illustrate the impact of the previously two UE's capability parameters on the assignment of T-slots, the following simple example is considered. The MAC-hs scheduling is based on a simple Round Robin algorithm and the radio interface is supposed to be error free.

Let us firstly evaluate a logical round trip time between a UE and the MAC-hs entity ($LgRTT_{UE/MAChs}$). The HS-ShCoCH precedes the HS-PDoShCH by 2 slots or 1,33 ms (T_{signal}). The radio delay propagation T_{propag} between the Node B and a UE is set to 1/2 slot. The processing time T_{proces} in a UE (i.e. the interval between a reception of the last data bit by a UE and the transmission of the corresponding Ack/NAck from the UE entity) is set to 7,5 slots. Thus, the time between the transmission of the first bit on the HS-ShCoCH and the reception of the last bit of the corresponding Ack/NAck on the HS-DePCoCH is 16,5 slots:

$$LgRTT_{UE/MAChs} = T_{signal} + 2*(T_{T\text{-}slot} + T_{propag}) + T_{proces} = 2 + 2*(3 + 1/2) + 7{,}5 = 16{,}5 \text{ slots (11ms)}$$

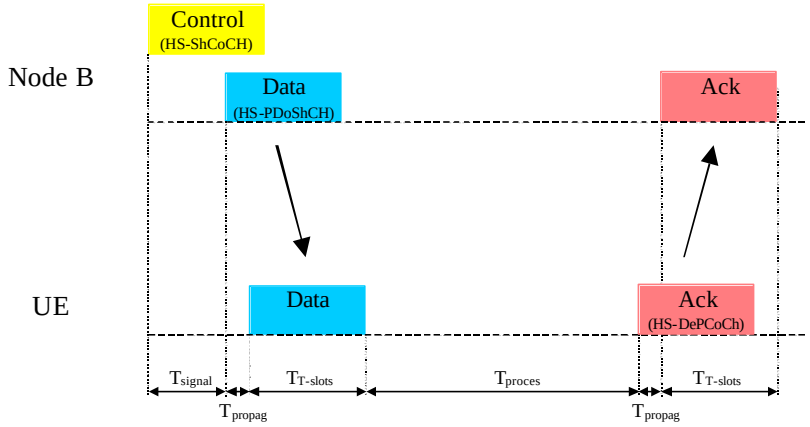


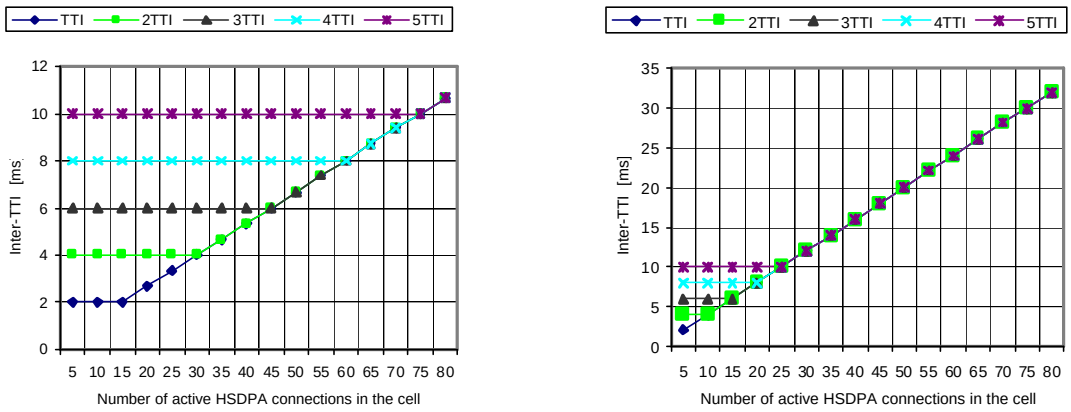
Figure 35. Logical Round Trip Time between a UE and the MAC-hs entity in Node B.

To see impact of higher values of MinInter-TTI on the number of active HARQ processes, we vary values of MinInter-TTI from 1 up to 5 T-slots (i.e. 2 to 10 ms). All UEs in the cell have the same capability parameters (channelization codes and MinInter-TTI). The MAC-hs scheduler can assign in a T-slot up to 15 channelization codes. The number of active HSDPA connections (or active UEs) in the cell varies from 10 to 80.

Figure 36 shows the inter-TTI of a UE as a function of number of active HSDPA connections in the cell for different values of MinInter-TTI. Figure 36a is for a case of one code per UE, whereas figure 36b describes a situation with three codes per UE. Beyond a certain number of UEs in the cell, the inter-TTI starts to increase above the MinInter-TTI. For example, in a case of one code per UE and MinInter-TTI = 5TTI (or 10 ms), the inter-TTI start increasing

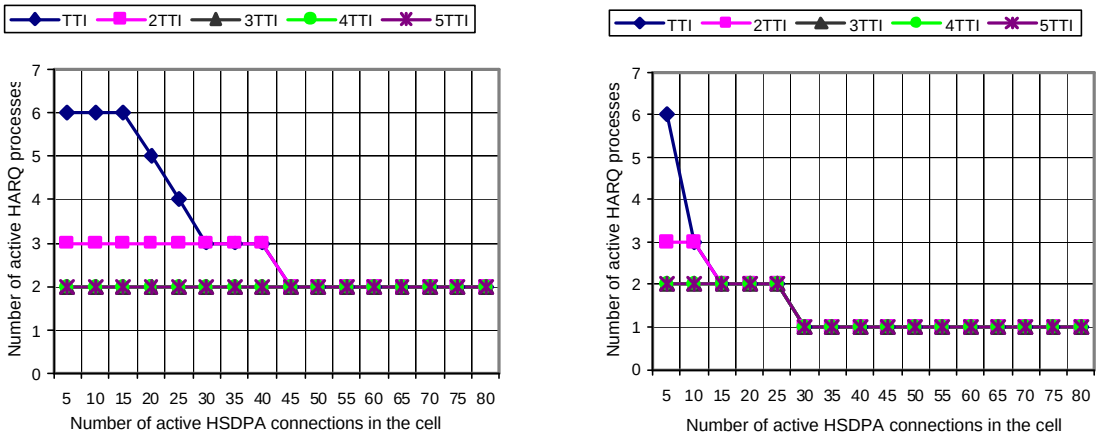
above MinInter-TTI when the number of UEs in the cell becomes higher than 75 (see figure 36a). In the case of three codes per UE, the threshold is three times lower (25 UEs, figure 36b).

Figure 36 provides information how often T-slots are assigned to a UE with regard to the MinInter-TTI and the number of active UEs in the cell; the effective user data rate will depend on BlER and on employed MCSs. Figure 37 illustrates how many HARQ instances (S&W processes) are active per UE for specific system parameters (number of active HSDPA connections in the cell, MinInter-TTI, number of channelization codes per UE). Notice that for certain combinations of system parameters, there is only one active HARQ process per UE.



a) one channelization code per UE b) three channelization codes per UE

Figure 36. Inter-TTI as function of number of active HSDPA connections in the cell for different values of minimum inter-TTI Î (1, 2, 3, 4, 5).



a) one channelization code per UE b) three channelization codes per UE

Figure 37. Number of active HARQ processes per UE as function of number of active HSDPA connection in the cell for different values of minimum inter-TTIÎ (1, 2, 3, 4, 5).

5.3.2 Periodic allocation mode

The previous example illustrates how rapidly the scheduler can reallocate radio resources for different system parameters. Reference [M02] investigates the HSDPA performance for streaming services. However, audio or video streaming services do not strictly require such a dynamic allocation of HSDPA resources. Streaming services generate regularly outgoing data. Such data flows postulate a periodic allocation of T-slots with a constant period instead of the dynamic allocation. We propose to introduce a specific allocation denoted as a periodic allocation that is more suitable for streaming or broadcast services. In this mode, radio resources are allocated periodically with a constant period. This period is set up when (re)configuring the radio bearer. A value of the constant period is determined according to the reserved rate for the given radio bearer. The proposed allocation mode can still coexist with the dynamic allocation.

Another reason of using the periodic allocation could be when UEs dispose of restricted memory and the buffer sizes are too small to compensate long jitter. The memory constrains could be expected in first versions of 3G mobiles.

Within the periodic allocation, a UE waits for data in its "reserved" T-slots. Hence, the UE knows in advance when to expect data prior to receiving and decoding the downlink-signaling message (HS-ShCoCH). If channel conditions are steady, the UE does not necessarily need to correctly decode the associated downlink-signaling message in order to decode data on the HS-PDoShCHs. In such situation, the UE may employ a "blind" detection and reutilize signaling information from the previously correctly received downlink-signaling message. Figure 34 shows that for certain system parameters the UE does not theoretically even need to know the HARQ Id since just one S&W process is active. The knowledge of the next assigned T-slot makes the periodic allocation more robust to errors occurring on the downlink-signaling channel in comparison with the dynamic allocation.

If a UE is not assigned radio resources in its reserved T-slot (e.g., due to lower data rate or higher priority services), the UE has to be informed about it. This notification can be provided, by setting up a flag (1 bit) in HS-ShCoCH that would be dedicated to such an event.

In certain situations, it would be useful to dispose of a possibility to modify the fix allocation period (e.g., if the number of HSDPA connections in the cell considerably change or the user data flow increased/decreased). Thus, a third allocation mode can be assumed; for example denoted as semi-static allocation. In the periodic allocation, the inter-TTI period remains constant during a whole establishment of the radio bearer. In a case of the semi-static allocation, the "constant" period could be updated according to the evolution of system parameters.

5.3.3 Periodic allocation mode - simulation

This section analyzes impact of the periodic allocation on the dynamic allocation. The analyses do not cover issues concerning the HSDPA link adaptation procedure or the HSDPA retransmission scheme.

Two types of services are assumed in our experiments: (i) background and (ii) streaming. The background service is considered mainly for a traffic modeling issue. Streaming source of traffic is not precisely modeled; we assume that the average data rate of this traffic is proportional to the inter-TTI interval. Simulation results are presented for different system parameters such as (i) the number of active HSDPA connections in the cell, (ii) MinInter-TTI of UEs and (iii) ratio between users running streaming and background services.

Parameter setting

A UE is assumed to run either a background or a streaming service. The MinInter-TTI parameter is set differently for Background and Streaming UEs. The MinInter-TTI of a Background UE is set to one T-slot (2 ms) and the MinInter-TTI of a Streaming UE is set to 5 T-slots (10 ms).

The MAC-hs scheduling is based on Round Robin algorithm. The scheduler can assign in a T-slot up to 15 channelization codes. When allocating radio resources, the algorithm takes into account UE parameters such as (i) the MAC-hs transmitter window size, (ii) MinInter-TTI and (iii) the maximum number of channelization codes that the UE can utilize. The scheduler uses in a T-slot all available codes of a currently served UE. The studied model considers that the MAC-hs buffer always contain data that can be sent; for this modeling reason we chose a background service rather than a conversional one.

The logical round trip time between a UE and the MAC-hs entity, $LgRTT_{UE/MAChs}$, is set to 16,5 slots (or 11 ms). The evaluation of $LgRTT_{UE/MAChs}$ is provided in section 5.3.1 (figure 35). The number of active HSDPA connections (or active UEs) in the cell varies from 10 up to 80. The maximum number of channelization codes (or physical channels) is three.

Our experiments consider two scenarios. The first scenario (scenario A) is for a case where the MAC-hs scheduler utilizes just the dynamic allocation mode. The second scenario (scenario B) describes a case where the periodic and dynamic allocations are used together.

Scenario A (dynamic allocation mode)

This scenario considers that both types of UEs (Background and Streaming) are served with the use of the dynamic allocation mode. Both types of UEs have the same priority. There is assumed automatic adaptation of the serving rate to UEs.

The simulation results are presented for three different proportions between Background and Streaming UEs in the cell: 70% + 30%, 50% + 50% and 30% + 70%. The first number (respec-

tively the second one) indicates a percentage of Background UEs in the cell (respectively Streaming UEs). For example, 20 UEs and the proportion 70% + 30% gives 14 Background UEs and 6 Streaming UEs.

Figure 38a and 39a shows the inter-TTI of a user as function of number of active HSDPA connections in the cell for different proportion between Background UEs and Streaming UEs. The figures 38a and 39a are complementary. Figure 38a describes the inter-TTI of a Background UE and figure 39a illustrates the inter-TTI of a Streaming UE. The figures show that different proportions of Background and Streaming UEs in the cell have a little impact on their inter-TTI.

Until 30 UEs in the cell, the curves of both types of UE differ. Beyond this number, the curve of Background UE becomes identical with the curve of Streaming UE.

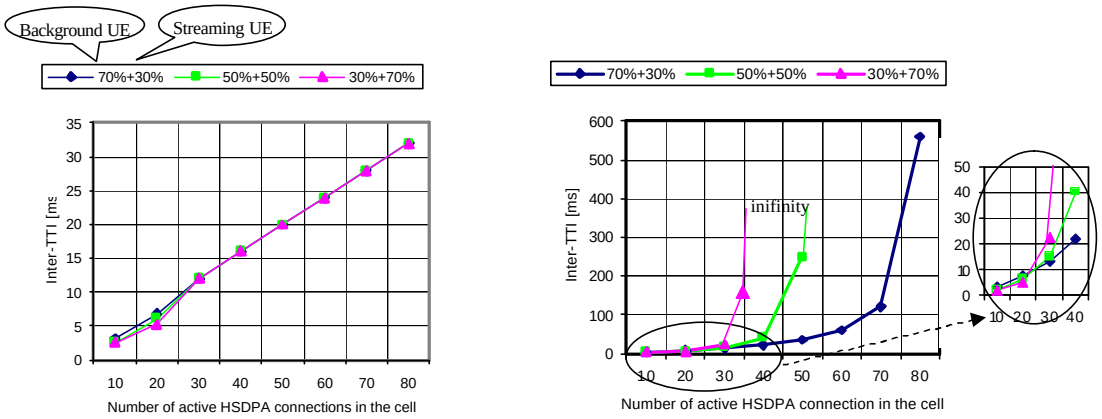
Scenario B (dynamic + periodic allocation modes)

In this scenario, the periodic allocation is activated. The Background UEs are allocated T-slots by using the dynamic allocation, whereas Streaming UEs are allocated T-slots by using the periodic one. Streaming UEs have higher priority than Background UEs. Resources are fairly shared among Streaming UEs and Background UEs take the rest of available resources. As in scenario A, there is automatic adaptation of the serving rate to UEs.

Parameter setting remains same as in scenario A and simulation results are again presented for three proportions between Background and Streaming UEs (70% + 30%, 50% + 50%, 30% + 70%).

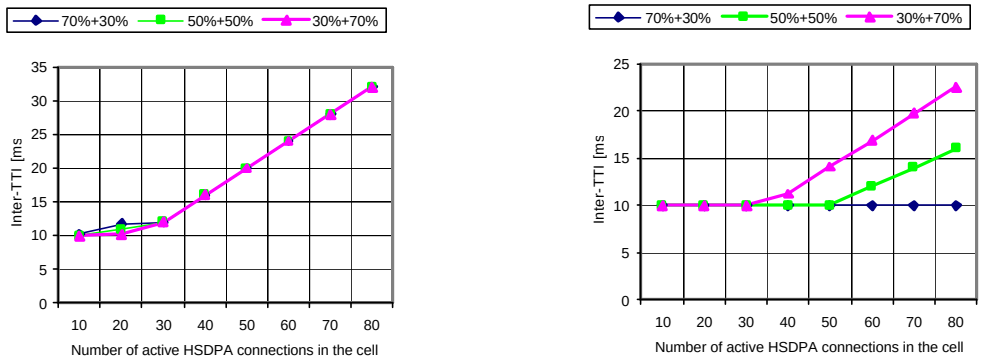
Likewise figure 38b), figure 39b) shows the inter-TTI of a user as function of number of active HSDPA connections in the cell for different proportions between Background and Streaming UEs. These two figures are again complementary; figure 38b (respectively figure 39b) describes the inter-TTI of a Background UE (respectively a Streaming UE). When comparing the inter-TTI of Background UE and Streaming UE in scenarios A and B, we observe that for small number of UEs (until about 20 UEs) curves coincide in both scenarios. Beyond this threshold, the inter-TTI of Background UE and the inter-TTI of Streaming UE grow differently. The inter-TTI of Background UEs (figure 38b) starts considerably increasing. After the number of UEs exceeds a certain level (about 35 UEs for the proportion 30% + 70% and about 50 UEs for the proportion 50% + 50%), the radio resources become shared just among Streaming UEs.

The inter-TTI of Streaming UEs (figures 39b) remains constant until the number of Streaming UEs becomes so high that they cannot all be served in the interval of the constant period, which corresponds to MinInter-TTI (10 ms).



a) scenario A b) scenario B

Figure 38. Inter-TTI of Background UEs (minimum inter-TTI = 2 ms) as function of number of active HSDPA connections in the cell for different proportion between Background and Streaming UEs.



a) scenario A b) scenario B

Figure 39. Inter-TTI of Streaming UEs (minimum inter-TTI = 10 ms) as function of number of active HSDPA connections in the cell for different proportion between Background and Streaming UEs.

5.3.4 Study of the HSDPA allocation mode – conclusions

Shortening of the allocation period from 10 ms to 2 ms makes the allocation mode of HSDPA very dynamic and scheduled UEs quickly vary in time. Nevertheless, streaming or broadcast services do not really need such fast reallocation of resources. For such type of services, we propose to utilize a periodic allocation where radio resources are allocated periodically with a constant period. The periodic allocation may also be needed if UE buffer sizes are small and cannot compensate long jitter.

The study analyzing impact of periodic allocation on the dynamic allocation shows that a compromise has to be made between the proportion of error sensitive services (such as background services) and delay sensitive services (such as streaming services) in the cell.

In the periodic allocation, the UE knows in advance when to expect data on HS-PDoShCHs prior to receiving and decoding the downlink-signaling message (HS-ShCoCH). Thus, the UE does not really need to correctly decoded HS-ShCoCH in order to decode HS-PDoShCHs. This feature makes the periodic allocation more robust to errors on the downlink-signaling channel (HS-ShCoCH) in comparison with the dynamic allocation.

A third type of allocation mode, semi-static allocation, can be considered beside the periodic allocation. In the periodic allocation, the inter-TTI period remains constant during a whole establishment of the radio bearer. In a case of the semi-static allocation mode, the inter-TTI period could be updated according to the evolution of a number of HSDPA connections in the cell, user data rate, radio channel conditions, etc.

5.4 Link adaptation in HSDPA

In comparison with the traditional CDMA context (IS95, WCDMA), the HSDPA link adaptation procedure is not provided through fast power control and variable spreading factor ([KF02a]). Instead of it, Modulation and Coding Schemes (MCSs) are continuously adjusting according to currently known radio channel conditions in the Node B.

Contrary to the EGPRS link adaptation procedure, the HSDPA one does not consider the possibility of modifying the MAC-hs PDU size during retransmissions attempts. This subject is treated in this paragraph. We firstly describe the standard HSDPA link adaptation procedure and issues that it brings. Then, we depict modifications that we suggest introducing into this procedure. Impact of this enrichment on MAC-hs HARQ process management and on the specific numbering used by MAC-hs is analyzed.

5.4.1 Link adaptation procedure – standard procedure

When transmitting a new d-Block, the MAC-hs scheduler determines a suitable payload size L of the d-Block according to the feedback information from the UE. The d-Block is assigned to one of free HARQ Ids. This HARQ process controls the transmission (and retransmission) of the d-Block. If a retransmission occurs, the originally selected size of the d-Block is kept constant and other MCS (more robust) may be employed. Different MCSs lead to different sizes of coded blocks. Hereafter, a d-Block on which is applied channel coding is denoted as coded Block or c-Block.

Problem of the inaccurate selection of the d-Block size

The selection of a suitable payload size L of a d-Block is important since the size of the d-Block cannot be modified during retransmission attempt(s). Due to inaccurate estimations, feedback delays and quick change of radio conditions, the feedback quality information from a UE (indicated by the CQI flag in HS-DePCoCH) may be inaccurate. Thus, the MAC-hs scheduler may select the payload size L of a d-Block inadequately. A bad selection of the parameter L can be to some extent rectified by using more robust MCSs during retransmissions. A more robust MCS requires a higher number of HSDPA channelization codes, i.e. more physical channels has to be assigned the UE. Nevertheless, the number of codes may not unrestrainedly be increased. The upper bound of 15 channelization codes (see section 5.2.2) is further constrained by UE capabilities. As we have seen in section 5.2.2, one of UE's capability parameters limits the maximum number of HS-PDoShCHs that a UE is capable to receive in a T-slot. Therefore, a bad selection of the parameter L cannot always be remedied by using a more robust MCS during retransmissions.

Notice that the issue of the inaccurate selection of the data block size also arises in the GPRS or EGPRS system. In GPRS, the initial selection of data block size too big (i.e. the coding scheme, CS, too light) force us to reuse this light CS during all subsequent retransmission attempts. The initial data block size cannot be modified in order to employ a more robust CS during retransmissions. If channel conditions remain poor, this leads to a real deadlock. This strong constraint is remedied in the EGPRS system. Contrary to GPRS (and HSDPA), the EGPRS link adaptation procedure allows modifying the data block size in order to apply a more robust MCS during different retransmission attempts (see e.g., [3G0460] or [MF02]). We detail this EGPRS mechanism at the end of section 5.4.2.

5.4.2 Link adaptation procedure – proposed modifications

To deal with a first bad estimation of the parameter L in the HSDPA link adaptation procedure, we propose to rectify the payload size L of a d-Block during different transmission attempts. Hereafter, we consider the rectification of the d-Block size before the first retransmission attempt.

If a d-Block has to be retransmitted and there is no suitable MCS among the allowed set of MCSs that can be employed, the payload size L of the d-Block is halved (if possible). Thus, the blocking of the d-Block due to poor radio channel conditions is avoided since a more robust MCS set can be applied on the d-Block.

Implementation of proposed modification

The proposed modification can be implemented by the following way. The management controlling the cutting of d-Blocks only halves the payload size L of a d-Block if the HARQ

transmitter receives the NAck. By receiving the NAck, the HARQ transmitter can assume that the HARQ receiver correctly registers the last valid value of NDI. The NAck condition reduces the risk of later misinterpretation of the NDI value by the HARQ receiver (see the example in section 5.2.3, figure 34).

If the cutting of a d-Block takes place, the originally assigned HARQ Id ($HARQId_{old}$) is released. The original d-Block ($d\text{-}Block_{orig}$) forms two new d-Blocks, denoted as d-Block' and d-Block". The cutting of the $d\text{-}Block_{orig}$ is realized in such a way that the new d-Blocks contain entire Data PDUs of RLC. To avoid several successive cutting of the $d\text{-}Block_{orig}$, neither the d-Block' no the d-Block" size is further modified.

Both d-Block' and d-Block" are assigned to two new available HARQ Ids. One of them can also be the old one ($HARQId_{old}$). Both HARQ processes correspondingly update their S&W flags. The d-Block' keeps the old TSN, whereas the d-Block" is assigned the next new available TSN. There is not necessity to introduce any new additional signaling to inform the HARQ receiver (in UE) about the modification of the d-Block size. All necessary signaling is already provided through the existing NDI management (for more about this management see section 5.2.3).

Remark: A different implementation solution could be to assign next new available TSNs as to the d-Block' as to the d-Block". If we want to avoid the detection of the old unused TSN at the receiver side via the timer T1 (see section 5.2.3), a new flag has to be introduced in the downlink-signaling message. This flag indicates the occurring of split event (1 bit) and the old unused TSN (modulo 64, 6 bits). According to this flag, the HARQ receiver advances the MAC-hs receiver window without waiting for the expiration of the timer T1. In our simulation experiments (scenario B), we implement the solution where the d-Block' keeps the old TSN and the d-Block" is assigned the next new available TSN.

The new available TSN assigned to d-Block" does not have to necessarily follow the TSN of the $d\text{-}Block_{orig}$. Between the transmission and retransmission of the $d\text{-}Block_{orig}$ some new d-Blocks can be scheduled. Consequently, the in-sequence delivery function of MAC-hs become violated. Notwithstanding, the in-sequence delivery of MAC-hs is not really necessary, if an AM entity is used at the RLC level. Notice that a UM RLC entity cannot be employed in this situation due to the UM behavior upon receiving a RLC block out of order. Any (UM) Data PDU out of order results in the discard of all Data PDUs that are part of the same SDU.

If the in-sequence functionality of MAC-hs has to be preserved (since the RLC level uses the UM entity), a similar mechanism to the EGPRS one can be considered. In EGPRS, a data block can be resegmented into two parts in order to move from light MCSs to more robust MCSs during retransmission attempts. Each part of the data block is transferred in different radio blocks. The resegmentation and sub-numbering of the data block are indicated by *Split Block indicator* field (SPB, 2 bits). Through this flag, the receiver is signaled the first and sec-

ond part of the split data block. The SPB flag is carried in the RLC/MAC header. In case of the HSDPA mode, the resegmentation and sub-numbering of a MAC-hs PDU require to introduce a new flag (2 bits) in the downlink-signaling messages (HS-ShCoCH).

5.4.3 Simulation environment and parameter setting

The simulation model and protocol architecture are illustrated in figure 40. All UEs in the cell have the same UE capability parameters: (i) maximum number of channelization codes per UE is set to 6 and (ii) MinInter-TTI = 2 ms (or one T-slot). The number of UEs in the cell does not vary.

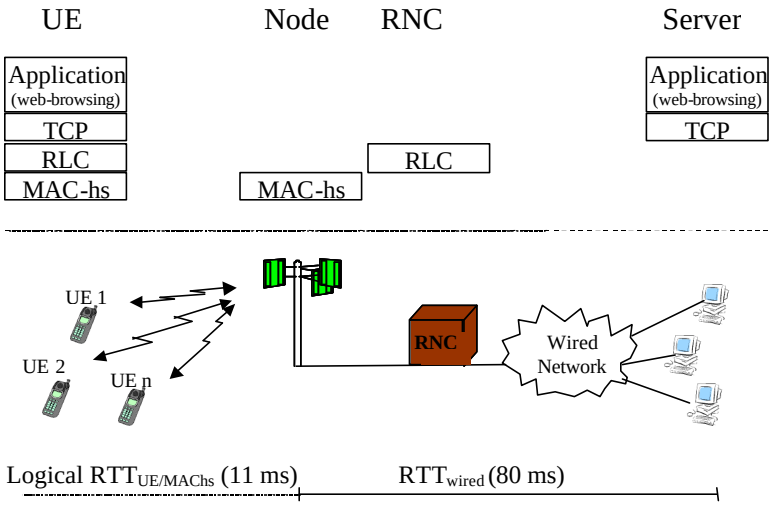


Figure 40. The simulation model and layer architecture.

MAC-hs scheduler

The MAC-hs scheduling is based on Round Robin algorithm. The maximum number of HSDPA channelization codes that the scheduler can assign in a T-slot is set to 12 ([PP01]). If possible, the scheduler assigns in the T-slot all available channelization codes. Several users can be scheduled in a T-slot. An HARQ process of a UE that requires a retransmission of block has higher priority than an HARQ process of the UE that transmits a block for the first time

MCSs and simulation of radio conditions

There are considered 9 types of MCSs and 6 sizes of d-Block (MAC-hs PDU). The d-Block sizes correspond to 2, 3, 4, 6, 8 and 12 RLC Data PDUs; the RLC Data PDU size is set to 240 bits. Possible combinations of MCSs and d-Block sizes are given in table 9. The d-Block size of 720 bits is only employed in the scenario B.

	Size of MAC-hs PDU (data rates), MCSs					
Channelization codes	480 bits (240kb/s)	720 bits (360kb/s)	960 bits (480kb/s)	1440 bits (720kb/s)	1920 bits (960kb/s)	2880 bits (1,44Mb/s)
2	QPSK 1/4 (MCS4)	QPSK 1/3 (MCS5)	QPSK 1/2 (MCS6)	QPSK 3/4 (MCS7)	16QAM 1/2 (MCS8)	16QAM 3/4 (MCS9)
4	QPSK 1/8 (MCS2)	QPSK 0,18 (MCS3)	QPSK 1/4 (MCS4)	QPSK 1/3 (MCS5)	QPSK 1/2 (MCS6)	QPSK 3/4 (MCS7)
6	QPSK 0,08 (MCS1)	QPSK 1/8 (MCS2)	QPSK 1/8 (MCS2)	QPSK 1/4 (MCS4)	QPSK 1/3 (MCS5)	QPSK 1/2 (MCS6)

Table 9. Applicable Modulation and Coding schemes for different MAC-hs PDU sizes.

Variable radio channel conditions are simulated through a variable Signal to Interference Ratio (SIR). The SIR follows a normal (gaussian) distribution $N(m,d^2)$; the mean m is set to 0 and the standard deviation $d \in$ (4 dB, 5 dB). A memory of the random process is indicated by a parameter Tv, where Tv ∈ (15 ms; 20 ms; 40 ms; 0,1 s; 0,4 s; 1,5 s).

Link adaptation

In our simulation experiments, the link adaptation is provided according to SIR's values, instead of CQI values. For a scheduled UE, a MCS is selected in such way that SIR(MCS)<SIR', where SIR' is the last known value of SIR indicated by the UE. The SIR(MCS) thresholds for every MCS are indicated in table 10 (inspired by [PP01]).

	MCS 1	MCS 2	MCS 3	MCS 4	MCS 5	MCS 6	MCS 7	MCS 8	MCS 9
SIR [dB]	-12	-7	-5	-4	-1	1	3	5	9

Table 10. SIR(MCS) thresholds for every MCS.

Due to the feedback delay (radio propagation, data processing in the Node B / UE) and scheduling, there is a certain delay between the last indicated value of SIR (from a UE) and the moment of selecting a MCS. The minimum delay is set to 6 ms (3 T-slots) and the maximum delay is set to 20 ms (10 T-slots), i.e. after 20 ms a value of SIR in the Node B is refreshed.

When transmitting a new d-Block, the selected MCS can correspond to several d-Block sizes (see table 9). In order to enlarge the set of possible MCSs that can be utilized for retransmissions, the lowest d-Block size is selected; we are pessimist and we assume that channel

conditions get worse rather than they ameliorate. The selected MCS and d-Block size determine a number of channelization codes that need to be employed for the transmission. To illustrates the above describe procedure, let's consider the following example where SIR' = 2 dB. From table 10, we obtain MCS6. Table 9 proposes for this MCS three d-Blocks sizes (2880, 1920 and 920 bits); the smallest d-Block size is selected. By this way, we obtain the following values: MCS6, d-Block = 920 bits and 2 channelization codes (or physical channels).

Retransmissions of a d-Block are performed by selecting a MCS in the column corresponding to the d-Block size (for more see scenarios A and B).

The HARQ processing at the UE side is simulated in a simple way. A received c-Block is evaluated according to the following procedure:

if $SIR_{UE} < SIR(MCS)$ *than* the c-Block erroneous
else the c-Block without errors (d-Block correctly decoded)

where SIR_{UE} represents the current value of SIR calculated by the UE. A correctly decoded d-Block is delivered the MAC-hs reordering entity.

Air interface

The logical RTT between a UE and the MAC-hs entity, $LgRTT_{UE/MAChs}$, is set to 16,5 slots (11 ms). The evaluation of $LgRTT_{UE/MAChs}$ is provided in section 5.3.1. The feedback-signaling message carrying Ack/NAck and SIR (instead of CQI) is assumed to be error free.

Traffic model

The simulation model considers 30 UEs in the cell. Each UE runs a web-browsing session. A web-browsing session comprises of several packet calls (or web page downloads). A packet call is followed by a reading time interval to view the download contents. At the end of reading time interval, the UE downloads another web page and so on. Each packet call is characterized by its size that is modeled by a Pareto random variable with cutoff (α = 1,6; min = 1,8 kByte; max = 40 kByte; mean = 4,4 kByte [SN99]). The reading time interval is modeled by an exponential random variable (mean = 5 s).

Every connection includes in its layer architecture TCP, which controls the transmission rate according to the current TCP soft state (for more about TCP soft states see chapter 3). Therefore, traffic parameters such as the packet size and the packet inter-arrival time (see [ETSIa]) are not specified since they are not needed in our case.

A TCP transmission directly starts with data transfer; the three-way handshake and the closing TCP phase are not simulated. The MSS (Maximum Segment Size) equals to 540 bytes. The *ssthresh* (slow start threshold) is set to 32 MSS. The receiver buffer size (*rwnd*) is set to a suf-

ficiently high value that during a whole TCP transmission *twnd* = *f* (*cwnd*). The delay Ack function is deactivated, i.e. the TCP receiver acknowledges without delay every received segment. The RTT_{wired} (i.e. RTT between the server and Node B) is set to 80 ms for all TCP connections. The wired part of a TCP connection is error free.

5.4.4 Simulation

There are considered two scenarios. The first scenario (scenario A) preserves the standard link adaptation procedure, whereas the second scenario (scenario B) employs the proposed modifications.

Scenario A (standard scheme)

If a d-Block is required to be retransmitted and if there is not any suitable MCS (among the possible set of MCSs) that can be utilized in the given T-slot, the radio resources are assigned either to a different HARQ process of the given UE or to a HARQ process of a different UE. When the assignment and/or retransmission attempts fail 6 times, the d-Block is discarded from the HARQ transmitter buffer.

This scenario does not use the MAC-hs PDU size of 720 bits (see table 9). Discarded d-Blocks at the MAC-hs layer are retransmitted by the RLC entity; the RLC entity is assumed to operate in the acknowledged mode. The RLC Round Trip Time is set to 60 ms.

Scenario B (proposed scheme)

If a d-Block is required to be retransmitted and there is not any suitable MCS that can be utilized in the given T-slot, the d-Block is halved (d-Block', d-Block"). Otherwise, the d-Block remains unchanged. In the simulation experiment, the d-Block' keeps the old TSN, whereas the d-Block" is assigned the next new available TSN. Likewise in scenario A, if the assignment and/or retransmission attempts fail 6 times, the d-Block is discarded from the HARQ transmitter buffer.

The d-Block size of 720 bits is only used when halving the d-Block size of 1440 bits. The lowest d-Block size of 480 bits is not modified. Discarded d-Blocks are again retransmitted through the AM RLC entity; RLC RTT = 60 ms.

Results

The average number of discarded d-Blocks (MAC-hs PDUs) per UE and per 1 minute for different values of the parameter Tv is illustrated in figures 41 and 42. The figure 41 shows a case where the standard deviation δ is set to 4 dB and figure 42 is for the case of δ = 5 dB. The simulation runs 10 minutes. The average inter-TTI of a UE is 23 ms in both scenarios.

The figures show that for rapid variations of channel conditions (values of Tv are comparable with the average inter-TTI of a user), scenario B provides better results than scenario A. The number of discarded d-Blocks is lower in scenario B than in scenario A. For increasing values of Tv, the results of both scenarios come closer together. For slow variations of channel conditions (values of Tv are higher than the average inter-TTI of a user) both scenarios give about the same performance.

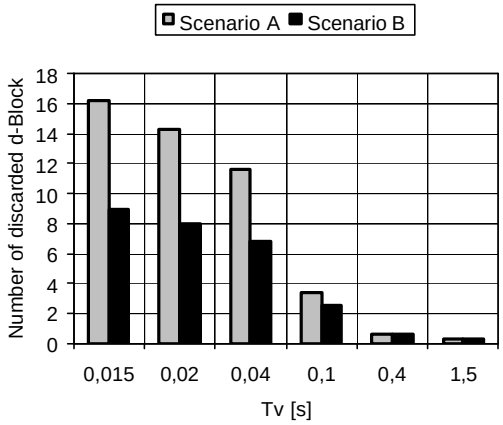


Figure 41. Average number of discarded d-Blocks per UE and per one minute for different values of the parameter Tv, d = 4 dB. The average inter-TTI of a user is 23 ms.
Scenario A: the d-Block size is not modified during retransmission.
Scenario B: the d-Block size is reviewed before the first retransmission.

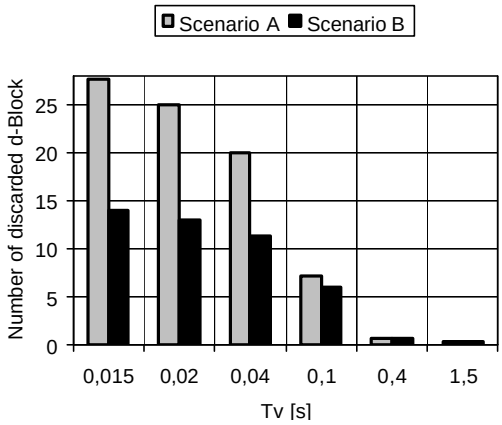


Figure 42. Average number of discarded d-Blocks per UE and per one minute for different values of the parameter Tv, d = 5 dB. The average inter-TTI of a user is 23 ms.
Scenario A: the d-Block size is not modified during retransmission.
Scenario B: the d-Block size is reviewed before the first retransmission.

5.4.5 Link adaptation in HSDPA – conclusions

The last part of chapter 5 proposes to enrich the HSDPA link adaptation and impact of this enrichment on the HARQ process management and on the MAC-hs numbering are investigated.

Each time a new d-Block is sent, its size is selected according to currently known radio channel conditions provided by UEs. The selection of a suitable d-Block size is important since the size cannot be modified during retransmissions. Due to inaccurate estimations, feedback delays and quick change of radio conditions, the MAC-hs scheduler may select the d-Block size inadequately. To deal with this issue, we suggest rectifying the size of a d-Block before its first retransmission attempt. If the first estimation of the size of a d-Block turns out to be inadequate (too big), its size is halved. Simulation results show that for rapid variations of channel conditions, which are comparable with the average inter-TTI of a user, the proposed link adaptation gives better results than the standard one. For slow variations of channel conditions, the performance of both schemes becomes similar.

The proposed link adaptation procedure can be further improved. Instead of just halving the size of a d-Block, the d-Block size could be decreased to other size that would better respect the radio channel conditions.

The upgrade of the link adaptation scheme either requires introducing a new flag in the downlink-signaling channel to inform the HARQ receiver about the d-Block size modification. Or, the HARQ receiver does not need to be notified but the in-sequence delivery functionality of MAC-hs is violated. This second issue can easily be remedied with the use of an AM RLC entity. A UM RLC entity cannot be employed since any RLC (UM) Data PDU out of order results in the discard of all Data PDUs that are part of the same SDU. The MAC-hs out-of-sequence delivery feature can be considered as an option. As in the case of an AM RLC entity, the required mode of an MAC-hs entity (i.e. in-sequence/out-of-sequence delivery) would be set up when creating the entity. If necessary, the initial MAC-hs mode could be modified during the HSDPA session.

Chapter 6
Conclusions and perspectives

6.1 Conclusions

Compared to (E)GPRS, the RLC protocol of UMTS is greatly improved and includes new features such as SDU discard function, ciphering or reconfiguration of the RLC transmission window during a connection. High flexibility of the protocol leads to a long list of adjustable RLC parameters. Our work focuses on the RLC buffer size of an AM RLC entity. We analyze how TCP downlink data transfers affect the RLC (RNC) buffer occupancy. Results in section 4.3 show that an image of the TCP soft behavior can be observed at the RLC buffer: (i) exponential growth of the RLC buffer occupancy during the TCP slow start soft state and (ii) linear growth of the buffer occupancy during the TCP congestion avoidance soft state. We observe that the buffer occupancy decreases as the values of RTT_{wired} increases (see figure 22). Our explication to this is that for high values of RTT_{wired} the RLC buffer occupancy growth is slower during the TCP slow start phase and when the TCP sender switches to the congestion avoidance phase, the RLC buffer occupancy is lower.

The introduction of the RLC protocol bellow the TCP/IP stack is useful for the global performance. The RLC mechanisms deal with radio errors much effectively than TCP ones do (block retransmissions, limited overhead, several Ack policies). A group of TCP extensions that provides local retransmissions through different sort of TCP agents in the base station (such as Snoop, Wireless TCP, etc) are redundant in the UMTS environment; their utilization should be avoided in UMTS.

Losses occurring on the UMTS radio interface are mainly due to the RLC counter (or timer). The TCP sender detects and reacts to a discarded TCP segment at the RLC level by the same way as it detects and reacts to a congestion event. To avoid segment losses at the RLC level, section 4.4 (EEN mechanism) suggests discarding SDUs virtually instead of really. A virtual RLC discard event is used as an indication to notify the TCP sender to update the TCP retransmission timer. The timer update avoids triggering of TCP spurious timeouts during periods of high BlER on the radio interface. To prevent a RLC deadlock due to the virtual discard events, the number of discards per SDU is monitored. If a certain limit is reached, the RLC reset procedure is activated.

The EEN mechanism can be applied as for uplink data transfers as for TCP downlink data transfers. In the uplink case, the TCP transmitting entity and the corresponding RLC entity are

located in the protocol stack of the same equipment (i.e. in UE). Therefore, the communication between the protocol entities can easily be realized. In the downlink case, the communication is not so straightforward. The direct interlayer communication between RLC and TCP can be replaced by an "indirect" communication. The EEN message can be conveyed to the distant TCP sender via ICMP.

The MAC-hs protocol deals with erroneous data much faster than RLC does. Shorter MAC-hs RTT allows using the HSDPA mode for delay sensitive services (such as streaming services) besides error sensitive services. The HSDPA allocation scheme is too dynamic for streaming services. Streaming services generates regularly outgoing data and these flows postulate rather a periodic allocation. Section 5.3 shows that a compromise has to be make between proportion of HSDPA users running error sensitive services and users running delay sensitive services. Knowing in advance T-slots when to expect data, the mobile does not necessarily need to correctly decode downlink-signaling messages to decode the associated data; as in the case of the dynamic allocation scheme.

In comparison with the EGPRS link adaptation procedure, the HSDPA one does not consider a possibility of modifying the data block size (MAC-hs PDU) during retransmission attempts. The enrichment of the link HSDPA adaptation procedure about this feature leads to: (i) introduction a new flag in the downlink-signaling message or (ii) violation of the MAC-hs in-sequence delivery functionality. The in-sequence delivery functionality of MAC-hs is not really needed, if an AM entity is used at the RLC level. The in-sequence/out-of-sequence MAC-hs delivery mode can be configured when creating an MAC-hs entity; as it is done in the case of an AM RLC entity. If necessary, the initial MAC-hs delivery mode can be (re)modified during the HSDPA session.

To our opinion, there is not an easy way to unify all three ARQ protocols. At least, we might try to unify the radio ARQ protocols. The RLC and MAC-hs protocols offer several similar functions such as ARQ, flow control, numbering, in-sequence delivery, etc. The way of combining the RLC and MAC-hs entities is not always obvious. The HARQ together with the AM RLC entity creates a very redundant scheme, whereas the HARQ together with the UM RLC entity may result in imperfect re-ordering. At first glance, a possibility is to create a common RLC/MAC-hs protocol. Nevertheless, such solution arises a question: where to implement this common protocol? For the reason of the fast control, the common protocol should be implemented as close as possible to the radio interface, i.e. in the Node B. On the other hand, for the handover reason, such protocol should be implemented in the RNC entity. We can see that the placement of the common RLC/MAC-hs protocol results in a contradiction. In our next research, we want to analyze if there is not a possible, at least, to utilize a common numbering for these two ARQ protocols.

6.2 Perspectives

The UMTS radio stack offers many variations and possibilities that do not allow treating all aspects of ARQ protocols in a dissertation like ours. Our analyses relate to research issues that could be extended in the future.

The UMTS radio layers dispose of information that can be useful for the Internet stack. In section 4.4 (the EEN mechanism), we have proposed to notify 4 layer if a virtual discard event occurs at the 2 level. In our future research, we want to look on some other possible primitives that could be introduced between the UMTS radio layer and the Internet stack.

Besides the RLC and MAC protocols of UMTS, we also contemplate to investigate PDCP (Packet Data Convergence Protocol). Besides the compression of IP headers, PDCP makes possible to recopy context between two serving radio networks sub-systems during handover procedures.

Other future work relates to the simulation models used in our investigations. The simulation models in section 4.3 assume independently distributed errors on the radio interface. Indeed, errors on the radio interface occur rather in bursts then independently. In order to obtain more realistic results, wireless errors have to be modeled more precisely. In a case of the HSDPA mode, the 3GPP specifications do not designate the type of HARQ scheme to be used. Our model (section 5.4) applies a simple HARQ scheme that does not combine context from different (re)transmission attempts. In the next step, we want to upgrade this model and to implement a HARQ scheme that utilize previously received context when decoding data.

Abbreviations

3GPP	Third Generation Partnership Project
Ack	Acknowledgement
AM	Acknowledge Mode
AMC	Adaptive Modulation and Coding
ARQ	Automatic Repeat reQuest
BER	Bit Error Rate
BlER	Block Error Rate
BMC	Broadcast/Multicast Control Protocol
CDMA	Code Division Multiple Access
CN	Core Network
DNS	Domain Name System
DupAck	Duplicated Acknowledgement
ECC	Error Correction Code
EDGE	Enhanced Data Rates for GSM Evolution
EEN	Early Error Notification
EGPRS	Enhanced General Packet Radio Service
FDD	Frequency Division Duplex
FEC	Forward Error Correction
FTP	File Transfer Protocol
GGSN	Gateway GPRS Support Node
GMSC	Gateway Mobile Switching Centre
GSM	Global System for Mobile communication
GPRS	General Packet Radio Service
HARQ	Hybrid Automatic Repeat reQuest
HLR	Home Location Register
HS	High Speed
HSCSD	High Speed Circuit Switched Data
HSDPA	High Speed Downlink Packet Access
HS-DoShCH	High Speed Downlink Shared Channel
HS-PDoShCH	High Speed Physical Downlink Shared Channel
HS-ShCoCH	High Speed Shared Control Channel
HS-DePCoCH	High Speed Dedicated Physical Control Channel

HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
ITU	International Telecommunication Union
I_u	Interface UTRAN – Core Network
I_{ub}	Interface Node B – RNC
I_{ur}	Interface RNC – RNC
LLC	Logical Link Control
MAC	Medium Access Control
MAC-b	MAC-broadcast
MAC-d	MAC-dedicated
MAC-c/sh	MAC-common/shared
MAC-hs	MAC-high speed
MCS	Modulation and Coding Scheme
MinInter-TTI	Minimum Inter-TTI
MMS	Multimedia Message Service
MSC	Mobile Switching Centre
MSS	Maximum Segment Size
NAck	Negative Acknowledgment
OSI	Open System Interconnection
Phy	Physical Layer
PDCP	Packet Data Convergence Protocol
PDU	Protocol Data Unit
PLMN	Public Land Mobile Network
PU	Payload Unit
QoS	Quality of Services
RLC	Radio Link Control
RFC	Request For Comment
RNC	Radio Network Controller
RNS	Radio Network Sub-system
RTO	Retransmission Time-Out
RTP	Real-Time Transport Protocol
RTT	Round Trip Time
RRC	Radio Resource Control
SACK	Selective Acknowledgment
SCTP	Stream Control Transmission Protocol
SDU	Service Data Unit

SDL	Specification and Description Language
SGSN	Serving General Packet Radio Service Support Node
SMS	Short Message Service
SRF	Simulated Radio Frame
SS7	Common Channel Signaling System No. 7
Status PDU	Status information control Protocol Data Unit
SuFi	Super Field
TCP	Transmission Control Protocol
TDD	Time Division Duplex
TDMA	Time Division Multiple Access
T-slot	Three slots
TM	Transparent Mode
TTI	Transmission Time Interval
UDP	User Datagram Protocol
UE	User Equipment
UM	Unacknowledged Mode
UMTS	Universal Mobile Telecommunication System
UTRAN	UMTS Terrestrial Radio Access Network
U_u	Interface UE – Node B
VLR	Visitor Location Register
WCDMA	Wideband Code Division Multiple Access

Publications

R. Bestak, P. Godlewski, P. Martins, *A TCP connection over uplink UMTS Radio Access Bearer in RLC acknowledged mode*, VTC 2002 - Spring, Birmingham, USA, May 2002.

R. Bestak, P. Godlewski, P. Martins, *Interactions between the TCP and RLC protocols in UMTS*, Multiaccess, Mobility and Teletraffic for Wireless Communications 2002, Rennes, France, June 2002.

R. Bestak, P. Godlewski, P. Martins, *RLC buffer occupancy when using a TCP connection over UMTS*, PIMRC 2002, Lisbon, Portugal, September 2002.

R. Bestak, P. Godlewski, P. Martins, *Three-slot allocation mode in HSDPA for UMTS*, MPRG/Virginia Tech Symposium on Wireless Personal Communications, Blacksburg, USA, June 2003.

R. Bestak, P. Godlewski, P. Martins, *HSDPA adaptation scheme of MAC-hs PDU size for re-transmissions*, Communication Systems and Networks 2003, Benalmadena, Spain, September 2003.

References

3GPP specifications

[3G0460] 3GPP TS 04.60, *General Packet Radio Service (GPRS); Mobile Station (MS) – Base Station System (BSS) interface; Radio Link Control/Medium Access Control (RLC/MAC) protocol*, 3GPP (Release 99), September 2002.

[3G22105] 3G TS 22.105, *Service aspects; Services and Service Capabilities*, 3GPP (Release 5), June 2003.

[3G23107] 3G TS 23.107, *Quality of Service (QoS) concept and architecture*, 3GPP (Release 5), Mars 2003.

[3G25211] 3G TS 25.211, *Physical channels and mapping of transport channels onto physical channels (FDD)*, 3GPP (Release 5), September 2003.

[3G25212] 3G TS 25.212, Multiplexing and channel coding (FDD), 3GPP (Release 5), September 2003.

[3G25213] 3G TS 25.213, Spreading and Modulation (FDD), 3GPP (Release 5), Mars 2003.

[3G25214] 3G TS 25.214, Physical layer procedures (FDD), 3GPP (Release 5), September 2003.

[3G25301] 3G TS 25.301, *Radio interface protocol architecture*, 3GPP (Release 5), September 2002.

[3G25302] 3G TS 25.302, *Service provided by the physical layer*, 3GPP (Release 5), September 2002.

[3G25306] 3G TS 25.306, *UE Radio Access capabilities*, 3GPP (Release 5), September 2002.

[3G25308] 3G TS 25.308, *High Speed Downlink Packet Access (HSPDA); Overall Description*, 3GPP (Release 5), Mars 2002.

[3G25321] 3G TS 25.321, *MAC protocol specification*, 3GPP (Release 5), December 2002.

[3G25322] 3G TS 25.322, *RLC protocol specification*, 3GPP (Release 5), December 2002.

[3G25323] 3G TS 25.323, *PDCP protocol specification*, 3GPP (Release 5), September 2002.

[3G25324] 3G TS 25.324, *BMC protocol specification*, 3GPP (Release 5), September 2002.

[3G25331] 3G TS 25.331, *Radio Resource Control (RRC)*, 3GPP (Release 5), September 2002.

[3G25848] 3G TS 25.848, *Physical layer aspects of UTRAN High Speed Downlink Packet Access*; 3GPP (Release 4), Mars 2001.

[3G25855] 3G TS 25.855, *High Speed Downlink Packet Access; Overall UTRAN description*, 3GPP (Release 5), September 2001.

[3G34108] 3G TS 34.108, *Common test environments for User Equipment (UE) conformance testing*, 3GPP (Release 4), Mars 2003.

RFC specifications

[RFC768] J. Postel, *User Datagram Protocol*, RFC-768, August 1980.

[RFC792] J. Postel, *Internet Control Message Protocol*, RFC-792, September 1981.

[RFC793] J. Postel, *Transmission Control Protocol*, RFC-793, September 1981.

[RFC1122] R. Braden, *Requirements for Internet Host – Communication layers*, RFC-1122, October 1989.

[RFC1191] J. Modul, S. Deering, *Path MTU Discovery*, RFC-1191, November 1990.

[RFC1323] V. Jacobson, R. Braden, D. Borman, *TCP extensions for high performance*, RFC-1323, May 1992.

[RFC1644] R. Braden, *T/TCP – TCP extensions for transactions*, RFC-1644, July 1994.

[RFC1889] H. Schulzrinne, S. Casner, R. Frederick, V. Jacbson, *RTP: A Transport Protocol for Real-Time Applications*, RFC-1889, January 1996.

[RFC1945] T. Berners-Lee, R. Fielding, H. Frystyk, *Hypertext Transfer Protocol-HTTP/1.0*, RFC-1945, May 1996.

[RFC2018] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, *Selective Acknowledgment option*, RFC-2018, October 1996.

[RFC2309] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, L. Zhang, *Recommendations on queue management and congestion avoidance in the Internet*, RFC-2309, April 1998.

[RFC2481] K. Ramakrishnan, S. Floyd, *A proposal to add Explicit Congestion Notification (ECN) to IP*, RFC-2481, January 1999.

[RFC2581] M. Allman, V.Paxson, W. Stevens, *TCP congestion control*, RFC-2581, April 1999.

[RFC2616] R. Fielding, J. Gettys, J.G. Mogul, H. Frystyk, L. Masinter, P. Leach, T.Berners-Lee, *Hypertext Transfer Protocol – HTTP/1.1*, RFC-2616, June 1999.

[RFC2757] G.Montenegro, S. Dawkins, M. Kojo, V. Magret, N. Vaidya, *Long thin networks*, RFC-2757, January 2000.

[RFC2960] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, V. Paxson, *Stream Control Transmission Protocol*, RFC-2960, October 2000.

[RFC2988] V. Paxson, M. Allman, *Computing TCP's retransmission timer*, RFC-2988, November 2000.

[RFC3042] M. Allman, H. Balakrishnan, S. Floyd, *Enhancing TCP's Loss Recovery Using Limited Transmit*, RFC-3042, January 2001.

[RFC3168] K. Ramakrishnan, S. Floyd, D. Black, *The addition of Explicit Congestion Notification (ECN) to IP*, RFC-3168, September 2001.

Articles and Books

[AW02] P. J. Ameigeiras, J. Wigard, P. Mogensen, *Impact of TCP Flow Control on the Radio Resources Management of WCDMA Networks*, VTC Spring 2002, Birmingham, USA, May 2002.

[BB94] E. Berruto, T. Brannlund, J. Gustafsson, W. Schott, *An SDL Methodology Used for Specifying the Radio Protocols in a CDMA System*, VTC Spring 1994, Stockholm, Sweden, June 1994.

[BB95] A. Bakre, and B. R. Badrinath, *I-TCP: Indirect TCP for Mobile Hosts*, Int. Conf. Distributed Computing Syst. (ICDCS), May 1995.

[BC01] R. Bestak, and Y. Chevalier, *Etude et Simulation de la couche RLC de l'UMTS en utilisant SDL*, Rapport de project de SVP, Mars 2001.

[BK96] B. S. Bakshi, P. Krishna, N. H. Vaidya, and D. K. Pradhan, *Improving Performance of TCP over Wireless Networks*, Technical Report 96-014, Texas A&M University, 1996.

[BK98] H. Balakrishnan, and R. H. Katz, *Explicit Loss Notification and wireless web performance*, IEEE Globecom Internet Mini-Conference, Sydney, Australia, November 1998.

[BO94] L. S. Brakmo, and S. O'Malley, *TCP-Vegas: New Techniques for Congestion Detection and Avoidance*, SIGCOMM'94, October 1994.

[BP95] L. S. Brakmo, and L. L. Peterson, *TCP-Vegas: End-to-end Congestion Avoidance on a Global Internet*, IEEE Journal on Selected Areas in Communications, October 1995.

[BP97] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, R. H. Katz, *A Comparison of Mechanisms for Improving TCP Performance over Wireless Links*, Networking, IEEE/ACM Transactions on Networking, December 1997.

[BS95] H. Balakrishnan, S. Seshan, and R. H. Katz, *Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks*, ACM Wireless Networks, December 1995.

[BS96] K. Brown, and S. Singh, *A network architecture for mobile œmputing*, INFOCOM '96, March 1996.

[BV98] S. Biaz, and N. H. Vaidya, *Distinguishing congestion losses from wireless transmission losses: a negative result*, Computer Communications and Networks, 1998.

[C00] D. Comer, *TCP/IP: architecture, protocoles, applications*, 3e édition, Dunod, Paris 2000.

[CA99] A. Calveras, M. Arnau, J. Paradells, *A Controlled Overhead for TCP/IP Header Compression Algorithm over Wireless Links*, Wireless Communications (Wireless'99), Calgary, Alberta, Canada, July 1999.

[CD92] G. Cohen, J. L. Dornstetter, P. Godlewski, *Codes Correcteurs d'Erreurs*, Masson, 1992.

[CG99] I. Cidon, A. Gupta, R. Rom, C. Schuba, *Hybrid TCP-UDP Transport for Web Traffic, Performance*, Computing and Communications Conference, 1999.

[DJ01] W. Ding and A. Jamalipour, *A New Explicit Loss Notification with Acknowledgment for Wireless TCP*, Personal, Indoor and Mobile Radio Communications, September 2001.

[DK02a] A. Das, F. Khan, A. Sampath, H. Su, *Design and Perfomance of Downlink Shared Control Channel for HSDPA*, Proc. 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communic ations, Lisbon, Portugal, 2002.

[DK02b] A. Das, F. Khan, A. Sampath and H. Su, *Adaptive, Asynchronous Incremental Redundancy (A2IR) with Fixed Transmission Time Intervals (TTI) for HSDPA*, Proc. 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, Lisbon, Portugal, 2002.

[E03] N. Enderlé, *Allocation de resources radios pour les services paquets dans l'UMTS*, Thèse de l'Ecole Nationale Supérieure des Télécommunications, Paris, Mars 2003.

[ETSIa] ETSI Technical Report 101 112, *Selection procedure for the choice of radio transmission technologies of the UMTS*, UMTS 30.03, April 1998.

[F94] S. Floyd, *TCP and Explicit Congestion Notification*, ACM Computer Communication Review, October 1994.

[FF96] K. Fall, and S. Floyd, *Simulation-based Comparisons of Tahoe, Reno, and Sack TCP*, ACM Computer Communication Review, July 1994.

[FP01] P. Frenger, S. Parkvall, E. Dahlman, *Performance Comparison of HARQ with Chase Combining and Incremental Redundancy for HSDPA*, VTC Fall 2001, Atlantic City, USA, October 2001.

[GL02] A. Gurtov, and R. Ludwig, *Making TCP Robust Against Delay Spikes*, Internet Draft, draft-gurtov-tsvwg-tcp-delay-spikes-00.txt, February 2002.

[H02] H. Holma, *WCDMA for UMTS*, VTC Spring 2002 – Tutorial, Birmingham, USA, May 2002.

[HP00] T. Hedberg, and S. Parkvall, *Evolving WCDMA*, Ericsson review No.2, 2000.

[HT00] H. Holma, and A. Toskala, *WCDMA for UMTS, Radio Access for Third Generation Mobile Communications*, John Wiley & Sons, Ltd, England 2000.

[IM02] H. Inamura, G. Montenegro, R. Ludwig, A. Gurtov, F. Khafizov, *TCP over 2.5G and 3G Wireless Networks*, Internet Draft, draft-ietf-pilc-2.5g3g-06.txt, February 2002.

[ITUa] ITU-T *Recommendation Z.100: Specification and Description Language SDL*, 1988.

[J88] V. Jacobson, *Congestion Avoidance and Control*, SIGCOMM'88, August 1988.

[JC01] K. R. Jung, J. Choi, P. Song, Y. H. Nam, *Design and Implementation of a Radio Resources Control Protocol in WCDMA using SDL*, VTC Fall 2001, Atlantic City, USA, October 2001.

[JS00] A. Jungmaier, M. Schopp, M. Tuxen, *Performance Evaluation of the Stream Control Transmission Protocol*, Proceedings of the IEEE Conference on High Performance Switching and Routing, June 2000.

[K01] M. A. Kazmi, *Radio Resource Control in Radio Access Networks with Distributed Coverage*, Thése de l'Ecole Nationale Supérieure des Télécommunications, Paris, Janvier 2001.

[KB98] T. J. Kostats, M. S. Borella, I. Sidhu, G. M. Schuster, J. Grabiec, J. Mahler, *Real-time Voice over Packet-switched Networks*, IEEE Network, Jan.-Feb. 1998.

[KF02a] T. E. Kolding, F. Frederiksen, P. E. Mongensen, *Performance Aspects of WCDMA Systems with High Speed Downlink Packet Access (HSDPA)*, VTC Fall 2002, Vancouver, Canada, September 2002.

[KF02b] T. E. Kolding and F. Frederiksen, *Performance and Modeling of WCDMA HSDPA Transmission/H-ARQ Schemes*, VTC Fall 2002, Vancouver, Canada, September 2002.

[L01] P. Lescuyer, *UMTS; les origines, l'architecture, la norme*, Dunod, Paris 2001.

[LC84] S. Lin, D. J. Costello Jr, M. J. Miller, *Automatic-repeat-request Error Control Schemes*, IEEE Communications Magazine, vol. 22, No. 12, December 1984.

[LC02] R. Love, B. Classon, A. Ghosh, M. Cudak, *Incremental Redundancy for Evolution of 3G CDMA Systems,* VTC Spring 2002, Birmingham, USA, May 2002.

[LG99] X. Lagrange, P. Godlewski, S. Tabbane, *Réseaux GSM-DCS, 4e édition revue et augmentée*, Hermes, Paris 1999.

[LG01] R. Love, A. Ghosh, R. Nikides, L.Jalloul, M. Cudak, B. Classon, *High Speed Downlink Packet Access Performance*, VTC Spring 2001, Rhodes, Greece, May 2001.

[LK00] R. Ludwig, R. H. Katz, *The Eifel Algorithm: Making TCP Robust Against Spurious Retransmission*, ACM Computer Communication Review, January 2000.

[LM97] T. V. Lakshman, and U. Madhow, *The Performance of TCP/IP for Networks with High Bandwidth-delay Products and Random Loss,* IEEE Transaction on Networking, Vol. 5, No. 3, June 1997.

[LM02] R. Ludwig, and M. Meyer, *The Eifel Detection Algorithm for TCP*, Internet Draft, draft-gurtov-ietf-tsvwg-tcp-eifel-alg-03.txt, February 2002.

[LV01] F. Lefevre, and G. Vivier, *Optimizing UMTS Link Layer Parameters for a TCP Connection,* VTC 2001 Spring, Rhodes, Greece, May 2001.

[M74] D. M. Mandelbaum, *An Adaptive-feedback Coding Scheme Using Incremental Redundancy,* IEEE Transactions on Information Theory, vol. 20, No. 3, 1974.

[M00a] C. Mihailescu, *Gestion des ressources radio pour les systémes radio mobiles de troisiéme génération*, Thése de l'Ecole Nationale Supérieure des Télécommunications, Paris, Mars 2000.

[M00b] Z. Mammeri, *SDL: modélisation de protocols et systèmes réactifs*, Hermes, Paris 2000

[M02] T. J. Moulsley, *Performance of UMTS High Speed Downlink Packet Access for Data Streaming Applications,* 3G Mobile Communication Technologies, May 2002.

[MA01] N. Miki, H. Atarashi, S. Abeta, M. Sawahashi, *Comparison of Hybrid ARQ Schemes and Optimization of Key Parameters for High-Speed Packet Transmission in W-CDMA Forward Link*, IEICE Trans. Fundamentals, July 2001.

[MF02] D. Molkdar, W. Featherstone, S. Lambotharan, *An Overview of EGPRS: the Packet Data Component of EDGE,* Electronic & Communication engineering journal, 14(1), 2002.

[MH03] M. Malkowski, and S. Heier, *Interaction between UMTS MAC Scheduling and TCP Flow Control Mechanisms*, ICCT2003, Beijing, Chine, April 2003.

[Mprim] Summary of the MPRIM project, http://www.telecom.gouv.fr/rnrt/Mprim.htm

[MR03] G. Manuel, and M. Rinne, *Performance of the Medium Access Control Protocol for the High Speed Downlink Packet Access*, Communication System and Networks, Benalmadena, Spain, September 2003.

[N01] L. Nuaymi, *Contributions sur les algorithmes de contrôle de puissance équilibrés dans les réseaux cellulaires*, Thése de l'Ecole Nationale Supérieure des Télécommunications, Paris, Janvier 2001.

[NA02] M. Nakamura, Y. Awad, S. Vadgama, *Adaptive Control of Link Adaptation for High Speed Downlink Packet Access (HSDPA) in W-CDMA*, Wireless Personal Multimedia Communications 2002, Honolulu, Hawaii, October 2002.

[P01] J. Pollonen, *Quality of Service Based Admission control for WCDMA Mobile Systems*, Master's Thesis, University of Helsinki, Department of Engineering Physics and Mathematics, November 2001.

[PF01] J. Padhye, and S. Floyd, *On Inferring TCP Behavior*, ACM SIGCOMM 2001, USA, August 2001.

[PP01] S. Parkvall, J. Peisa, A. Furuskär, M. Samuelsson, M. Persson, *Evolving WCDMA for Improved High Speed Mobile Internet*, Proceedings of the Future Telecommunications Conference 2001, Bejing, China, November 2001.

[RG01] M. Rossi, A. Giovanardi, M. Zorzi, G. Mazzini, *TCP/IP Header Compression: Proposal and Performance Investigation on a WCDMA Air Interface*, Personal, Indoor and Mobile Radio Communications, San Diego, USA, September 2001.

[RM98] K. Ratnam, and I. Matta, *WTCP: an Efficient Mechanism for Improving TCP Performance over Wireless Links*, Computers and Communications, 1998, ISCC '98, 1998.

[S94] W. R. Stevens, *TCP/IP Illustrated, Volume 1*, Addison-Wesley publishing company, 1994.

[SN99] Stacey, J. Nelson, I. Griffin, *T/TCP: TCP for Transactions*, Linux gazette, 47, November 1999.

[ST01] J. Sanchez, and M. Thioune, *UMTS: services, architecture et WCDMA*, Hermes, Paris 2001.

[T96] A. S. Tanenbaum, *Computer Networks – 3rd edition*, Prentice Hall, 1996.

[T99] L. Toutain, *Réseaux locaux et Internet*, Hermes Science, 1999.

[TC02] A. Talukdar, and M. Cudak, *Radio Resources Control Protocol Configuration for Optimum Web Browsing*, VTC Fall 2002, Vancouver, Canada, September 2002.

[TM02] V. Tsaoussidis, and I. Matta, *Open Issues on TCP for Mobile Computing*, The Journal of Wireless Communications and Mobile Computing, Wiley Academic Publishers, February 2002.

[WJ02] S. J. Wha, D.G. Jeong, B. Kim, *Design of Packet Transmission Scheduler for High Speed Downlink Packet Access Systems*, VTC Spring 2002, Birmingham, USA, May 2002.

[XC02a] H. Xu, Y-Ch. Chen, X. Xu, E. Gonen, P. Liu, *Simulation Analysis of RLC timers in UMTS system*, VTC Fall 2002, 2002 Winter Simulation Conference, December 2002.

[XC02b] H. Xu, Y-Ch. Chen, X. Xu, E. Gonen, P. Liu, *Performance Analysis on the Radio Link Control Protocol of UMTS system*, VTC Fall 2002, Vancouver, Canada, September 2002.

[YB94] R. Yavatkar, and N. Bhagawat, *Improving End-to-end Performance of TCP over Mobile Internetworks*, Workshop Mobile Computing Systems and Applications, December 1994.

[ZL02] L. Zhang, F. Li, J. Zhu, *Performance Analysis of Multiple Reject ARQ Theme at RLC layer in 3G*, VTC Fall 2002, Vancouver, Canada, September 2002.