Simulation of radio resource management for UMTS

Examensarbete utfört i Kommunikationssystem
av
Björn Knutsson

LiTH-ISY-EX-3414-2004
Linköping 2004
Simulation of radio resource management for UMTS

Examensarbete utfört i Kommunikationssystem
vid Tekniska Högskolan i Linköping
av

Björn Knutsson

LiTH-ISY-EX-3414-2004

Linköping 2004

Examinator:
Fredrik Gunnarsson, ISY

Handledare:
Frida Gunnarsson, ISY
Simulering av radioresurshantering för UMTS
Simulation of radio resource management for UMTS

Björn Knutsson

Nyckelord
Keyword
simulation, UMTS, RNC, resource management
Abstract

A current trend in the information society is that traditionally fixed computing resources are made available to mobile users. Most of the existing techniques for communication have been developed for stationary computing, and they must be adapted to the different connection properties of the mobile environment. One of the emerging mobile computing environments is the Universal Mobile Telecommunication System, UMTS. This system places demands on the quality of service that is provided to data flows, which requires resource management in the connection network. The most scarce resources in this system is the radio resources. The easiest way to conduct research in new and adapted techniques for communication is to perform simulations. Management of resources places restrictions on connections, and to get reliable results during simulations it must be included in the simulated environment. This paper discusses and builds a basis for development of UMTS radio resource management in the network simulator ns-2. A limited version of UMTS radio resource management is added to ns-2 and evaluated.
Tabel of Contents

Chapter 1 Introduction ......................................................... 1
  1.1 Object of thesis ......................................................... 2
  1.2 Reading guide .......................................................... 3
  1.3 Abbreviations ............................................................ 3

Chapter 2 Background information ........................................... 5
  2.1 Network layering and protocols ......................................... 5
  2.2 UMTS ............................................................................ 8
  2.3 Ns-2 .............................................................................. 14

Chapter 3 Component design ..................................................... 19
  3.1 Ns-2 components .......................................................... 19
  3.2 Limitations ................................................................. 19
  3.3 RNC component architecture ............................................. 21
  3.4 Radio resource management algorithms .................................. 23
  3.5 Future compatibility ..................................................... 26
  3.6 Adaptive streamer architecture ......................................... 26

Chapter 4 Evaluation of the RNC component ................................. 29
  4.1 Impact of implementation properties on simulation results ............. 29
  4.2 General points for simulations ............................................ 29
  4.3 Functional evaluation ..................................................... 31
  4.4 Evaluation of link sublayer versus RLC sublayer ....................... 34
  4.5 Evaluation of adaptive radio resource management effects .......... 38
  4.6 Evaluation of full simulations compared to simplifying parts of the network ................................................ 41
  4.7 Evaluation summary ..................................................... 44

Chapter 5 Evaluation of the adaptive streaming application .................. 47

Chapter 6 Conclusions ............................................................ 49

Chapter 7 Future work ............................................................ 51
  7.1 RNC ................................................................. 51
  7.2 Adaptive streamer ...................................................... 52

Bibliography ........................................................................... 53

Appendix A User guide .......................................................... 55
  A.1 Using the RNC component .............................................. 55
  A.2 Using the adaptive streamer ............................................ 60
1 Introduction

A current trend in the computing world is having access to data and other computers while on the move and simultaneously having access to telecommunication resources. To meet this trend, mobile access to mixed networks for both data and telecommunication is currently emerging. Most of the existing techniques for data communication have been developed to suit wired connections. A mobile connection has very different properties compared to a wired one, and to make efficient use of connection resources the control algorithms need to be adapted to the requirements of mobile computing.

One of the emerging systems for combined mobile data and telecommunication is the Universal mobile telecommunication system (UMTS), a mobile telecommunication system of the third generation (3G). UMTS is based on a radio access network to provide the mobile data connections. One of the main goals of UMTS is to improve the data transport ability compared to previous generations of mobile communication systems. A part of this is the introduction of quality of service (QoS) assurance for data flows. In UMTS, QoS assurance means that the connections used by data flows have some guaranteed properties, such as bandwidth usage and transfer delay limits. To assure that the guaranteed properties of a connection are fulfilled, resources must be reserved in the network to meet the requirements of the data flow. Examples of resources that can be reserved are bandwidth, power in radio transmissions and buffer space in network nodes. All such resources are limited and to avoid overreservation they must be managed. Such management have impact on higher layers of the network and thus the algorithms that controls the network behaviour in the higher layers. Examples of these algorithms are data transport protocols like TCP and data streaming protocols.

With the introduction of data communication into telecommunication systems the problem of managing resources to suit the needs of data flows becomes much more complex. Pure telecommunication have well-defined statistical properties that can easily be used to manage resources. With data traffic, there are no such well-defined properties. Different types of data have different properties and requirements on connections. An example of such a difference is a video-conference call compared to the delivery of e-mails. The voice of a video-conference have very strict limits on transfer delay, but low demands on bandwidth. The video data have high demands on both bandwidth and delay, but lower demands on delivery accuracy of data. Some data may be lost without compromising the picture quality. E-mail on the other hand have almost no requirements on delay (it can be measured in seconds or even minutes instead of milliseconds as for voice transmission) but all data must be delivered without errors. Data traffic on flows may also be changing over time which means that the resource management algorithms must adapt to the data flows. An example of this is web-browsing, which is characterized by periods of page downloading.
followed by longer periods of inactivity while the information in the downloaded page is examined.

When developing new control algorithms and adapting old ones to mobile computing it is much easier to evaluate them in a simulated environment than in a real one. In early stages of development, there may not even be a real system to test the algorithms in. Access to deployed systems may also be restricted for business secrecy reasons, since they are used to generate business profits. The complexity of mixed data and telecommunication systems makes it difficult to perform theoretical analyses, and simulations is sometimes the only option open for evaluations. To make accurate and reliable predictions about the behaviour of control algorithms, the simulating environment need to match what can be expected of a true system.

When choosing simulator to simulate UMTS there are three alternatives. Commercial simulators, open source simulators and proprietary simulators developed in special projects. The commercial simulators are developed to produce accurate and reliable simulations, and can be very detailed in their simulations. The downside of this is that they cost money to use and can be difficult to extend with new behaviour. There exists both general network simulators and simulators specifically developed for UMTS. Open source simulators are free and easy to extend. They benefit from the fact that many can use them and provide their own extensions, which often makes it possible to handle a large variety of simulation tasks. Proprietary simulators are usually not available for free use and can be directed towards very specific research items.

There are currently no easily accessible and free simulator for UMTS, but one is required to evaluate control algorithms for UMTS. This thesis is part of academic research, which means that an open source simulator is the best solution to be allowed to produce extensions. The base simulator selected for the work in this thesis is the network simulator ns-2, [10]. It is an open-source simulator that is commonly used by researchers to evaluate developed control algorithms and it is considered to be a good tool for simulating real network behaviour. It contains already developed components for many of the existing protocols and mechanisms in networking, but currently it have only limited support for radio access networks.

1.1 Object of thesis

One of the main concepts of UMTS is QoS. To get trustworthy results from simulations we need QoS limitations and guarantees for connections over the radio link, as these can impact results obtained when simulating and evaluating control algorithms in higher levels of the network.

To meet the demands posed by UMTS QoS assurance, resource management is required on all levels in the connecting network. This thesis will focus its work on the radio access network in UMTS. The ns-2 simulator currently lacks support to simulate core properties of UMTS correctly. To allow the use of the existing simulating environment when evaluating control algorithms on higher levels for connections, extensions must be added to the system. Some of the more interesting UMTS properties from a functional point of view are the communication protocols of the radio link, radio resource management and mobility functions. In this thesis, radio resource management functions of UMTS will be added to the
ns-2 simulation system. Mobility functions must also be considered, since resource management is required to let connections move in the radio access network. The radio link of UMTS is concurrently developed for ns-2 in a separate master thesis [11]. Since the resources of the radio link in the separate thesis are managed by the work in this thesis, cooperation is required to produce an interworking system.

There are more elements of UMTS that are not developed in ns-2, but they mainly consists of components that do not directly regulate traffic patterns but provides some of the higher level services offered by a UMTS network and will not be considered in the thesis.

To study some effects of the introduction of radio resource management, such as how connections are affected by resource sharing, the existing application layer of ns-2 is deemed to be insufficient. A choice have been made to implement an adaptive streaming application that tries to adapt its sending rate to the network conditions. The adapting streamer should adapt to changes in connection data rates imposed on the connection by the resource management algorithms, and the resource management should adapt to the changing traffic requirements of the adapting streamer.

1.2 Reading guide

In Chapter 2 the background information for the thesis is presented. It is divided in three sections. The first section describes network layering and protocols, the second section describes details of UMTS, and the last section describes the ns-2 simulator used in the thesis. In Chapter 3 design and implementation considerations are outlined. Simulations and results are presented in Chapter 4 and Chapter 5. Chapter 6 contains conclusions of the thesis and in Chapter 7 items that may be addressed by future work are presented. The appendix contains a user guide for the ns-2 components developed in the thesis.

1.3 Abbreviations

Bps Bits per second
CBR Constant Bit Rate
FTP File Transfer Protocol
MAC Medium Access Control
QoS Quality of Service
RAB Radio Access Bearer
RLC Radio Link Control
RNC Radio Network Controller
RRM Radio Resource Management
RTT Round Trip Time
TCP Transport Control Protocol
UDP User Datagram Protocol
UMTS Universal Mobile Telecommunication System
UTRAN  UMTS Terrestrial Radio Access Network
WCDMA  Wideband Code Division Multiple Access
2 Background information

2.1 Network layering and protocols

This section shortly describes the fundamental principles of network layering and protocols, while a more detailed description can be found in [15].

To reduce the design complexity of networks, they usually consists of several layers built on top of each other. The purpose of each layer is to provide higher layers with services while shielding them from details of how the offered services are implemented. The number of layers and their naming differs between different network architectures.

Each layer basically has a conversation with and exchanges messages with the corresponding layer on the receiving machine by using the connection service offered by the next lower layer as described in Figure 2-1. The rules for how and what messages should be exchanged at each layer in order to maintain the connection is commonly referred to as a protocol.

![Figure 2-1. The relationship between network layers and protocols.](image)

There are a few layers that are commonly used in most network architectures:

- **Physical layer**: transmits the raw data in physical form, and deals with what power levels and timing requirements that are necessary to transmit the data.

- **Data link layer**: responsible for turning the raw data flow of the physical layer into a link that appears to higher layers as an error-free connection between two machines. This can be accomplished by dividing the input data into smaller pieces and transmitting each piece with acknowledgment of received data.

- **Network layer**: concerned with how to interconnect several machines into a network and how data should find its way to the receiver through such networks. The networks may be heterogeneous with different protocols and addressing structure at lower levels, but the network layer should make this transparent to higher layers. This layer and lower layers are only concerned with direct point-to-point connection between neighbouring machines, and...
higher layers are concerned with an end-to-end connection between the source machine and
the destination machine.

The transport layer is responsible for dividing data into smaller pieces if necessary, pass it
to the network layer and ensure that the pieces arrive correctly at the receiver. The transport
layer determines what kind of connection service that are provided to higher layers and in
the end to the users of the network. The most common service offered is an error-free chan-
nel that delivers data in order. Other kinds of services are isolated message transporting and
broadcasting of data.

On top of these layers there are usually more layers, but they depend on the network archi-
teecture. An example is the application layer of the Internet, which contains several proto-
cols to send different types of data such as HTTP to transfer web pages and SMTP to send
e-mails.

2.1.1 Data link layer

The complexity of the data link layer can vary greatly depending on the properties of the
physical medium that carries the link. If the physical medium is of a broadcast type, a spe-
cial medium access control (MAC) layer is used. Its responsibility is to limit the number of
colliding messages on the shared medium (messages that overlap in time). If messages
overlap, they are usually received incorrectly and can not be used. This layer is placed
immediately above the physical layer (the lowest layer) in network layer stacks.

When the physical link is provided via radio communication, the data link layer is com-
bined from a MAC layer and a radio link control (RLC) layer. The responsibility of the
RLC layer is to provide the network with a view of a radio link as a straight line connection
between the two communicating nodes. Compared to the data link layer of a wired link, the
RLC and MAC layers of a radio link is much more complex. A radio link is much more
prone to errors during transmission of data, and the RLC layer solves this by sending small
packets and detecting the errors. Packets with errors may be retransmitted, depending on
what services are provided by the data link layer of each particular network architecture.

2.1.2 Transport layer

Two commonly used transport layer protocols are TCP and UDP. UDP is a connectionless
protocol that does not guarantee delivery of data. Examples of how it is used is in server-
client type request-reply situations and in applications where prompt delivery of data is
more important than accurate delivery. As a contrast to UDP, TCP is a transport protocol
with connections that provides a reliable way of transporting in-sequence data from a
sender to a receiver. Some of the properties of TCP may be desired by certain applications,
but not all and the TCP overhead for the extra features may be undesired. They can then
use UDP messages and provide their own delivery mechanisms.

The reliable connection of TCP is achieved by segmenting the incoming data flow into
packets and sending each packet separately to the destination. By using an acknowledg-
ment mechanism consisting of sequence numbering the data packets, the transmission of
all data in the flow is assured. Any lost data packets are detected by the acknowledgement
mechanism and are retransmitted. This allows all data to be received without any losses.
The data packets are then reassembled into the original byte flow using the sequence numbers of bytes within the transported data packets.

Another service provided by the TCP protocol is congestion control. A property of the network layer is that congestion of data in this layer causes lower overall data throughput. By avoiding or limiting congestion in the network layer, TCP can maintain a high level of data throughput. A short basic overview of how TCP tries to govern its send rate and control congestion is given here, while a full explanation can be found in [16]. Many different versions of the TCP protocol have been developed to address various details of the data transport. They generally use the same mechanism as outlined here, but the details in their operation differs.

TCP sends data packets and the receiver acknowledges the received data to the sender. Congestion is detected by measuring the time between data is sent and it is acknowledged, called the round trip time (RTT). If this time exceeds a time-out value, TCP assumes that the data have been delayed by congestion in the network. The time-out value is a prediction based on, and updated by, the measured RTTs. The TCP protocol uses a congestion window when determining its send rate. The window describes how much data are allowed to be transferred through the network without congestion and together with the RTT it determines the send rate of the TCP algorithm. To determine the initial allowed transfer rate, TCP starts with a low amount of data and then uses an exponential increase in data rate. This is called a slow start, which in steps double the data that are transferred to the receiver. When congestion is detected, the initial congestion window size is determined to be half the amount of data transferred in the last slow start step. After the slow start have determined an initial send rate, the congestion window is gradually increased, allowing a larger amount of data in transfer until it again detects congestion at which point the unacknowledged amount of data is reduced again. The process is presented in Figure 2-2.

![Figure 2-2. Evolution of TCPs congestion window with time.](image)

An important part in achieving a high throughput with TCP is to have a stable RTT. If the RTT varies a lot, the TCP algorithm can not predict it accurately. This means that the algorithm more often signals for congestion and reduces its send rate, although it might just be a short, temporary increase in RTT.
2.2 UMTS

UMTS stands for Universal Mobile Telecommunications System which is an example of a third generation mobile telecommunication system. It has been developed as an industry standard in the 3G partnership program, [1], to provide compatibility between different manufacturers equipment. The joint effort by the participating industries is also encouraged by the complexity and large scope of the problem.

From the beginning UMTS have been developed as a modular system. This allows easy upgrading of system behaviour and extends the technical life-span of the system. The core radio connection technique used by UMTS is WCDMA (wideband code division multiple access). This is a cell-based technique where a fixed radio base station provides data connections to all mobile users within range. If a user moves out of range, the radio base station can hand over the connection to another station that is closer. The WCDMA technique is further described in Section 2.2.2

UMTS is on a high level generally described in four different layers, which are presented in Figure 2-3. In standards these four layers may have different names depending on the currently used view of the system, but they usually contains the same components of the network. The core network is the backbone of UMTS and includes connection to the Internet. It may also connect to other networks specifically directed toward the mobile users needs. The core network is mainly used as a transport medium for data and most of the UMTS functionality is independent of its properties. The UMTS core network contains several modules that provides certain functions to the system e.g. billing functions, user mobility functions and connection to the core network. UTRAN (UMTS terrestrial radio access network) contains the radio interface that connects the mobile equipment to the functions provided by the UMTS core network. It is responsible for handling all aspects of the radio connection, including user mobility between different WCDMA cells.

This report will concentrate on the radio interface of UMTS, i.e. UTRAN. Some parts of the supporting network will be mentioned when explaining how UMTS is composed, but the report is mainly biased towards the radio interface.

![Figure 2-3. General UMTS domains.](image-url)
2.2.1 UMTS concepts

Quality of Service

One of the new and main concepts of UMTS compared to earlier systems is the incorporation of quality of service (QoS) assurance for data flows as described in [2]. QoS of a data flow can be many things. A few examples are connection data rate guarantees, delay guarantees and error rate guarantees. QoS have been included since different types of data have different QoS requirements, and one of the main goals of UMTS is to improve data delivery to mobile users compared to earlier communication systems. To ensure that the data network can handle all data transport requests, resources must be reserved for all data flows in the network. This includes both links in the core transport network and the radio link from the mobile equipment to the radio access network. A more detailed description of the UMTS QoS concept than the one given here can be found in [4].

In UMTS each connection by a mobile user is mapped to an entity called a bearer service. Each bearer service is adapted to suit the requirements of the connection in terms of QoS by associating it with a specific set of requirements of the data traffic for the connection. To provide end-to-end QoS for a connection (which is what the user sees), these requirements must be met in all parts of the network that handles the connection. To solve this the bearer service of a connection is composed of several bearer services, one for each part of the network (e.g. radio access network and UMTS core network). Each minor bearer service must comply with the requirements on the main bearer service. The bearer service used in the radio access network is called the radio access bearer (RAB).

Since the radio link connection to UMTS and the wired links of the UMTS core network have different connection properties, they use the QoS parameters for the bearer service in different ways. In the UMTS core network bearer service the QoS parameters are mapped to existing techniques for QoS assurance, see [4]. The actual mapping used is left to UMTS operators to decide and depends on their provisioning of resources in their networks. This is a typical example of how the UMTS standards are constructed. The standards define what is to be implemented, and the operators can decide how to implement it. The RAB uses the QoS parameters of the main bearer service more directly, but the values of a few parameters are stricter. This comes naturally since the RAB is only one part of the main bearer service and the main bearer service must provide for the end-to-end parameters.

Below is a short explanation of some of the RAB parameters that are interesting from a radio resource point of view. A full explanation of the RAB parameters is given in [4].

- **Traffic class**
  This parameter is used to describe the overall requirements of the connection data flow. It is used by UMTS to optimize transport through UTRAN for the data flow and to prioritize between packets from different flows. A connection that requires close attention by UMTS is assigned to the highest traffic class while a more tolerant connection selects a lower class. There are four classes defined, conversational, streaming, interactive and background class. The conversational class have the highest priority of adherence to the other QoS parameters and the background class have the least priority.
• **Bandwidth requirements**
  Both maximum bitrate and guaranteed bitrate can be specified as parameters, depending on traffic class. They are used to reserve resources in UTRAN and limits the rate actually delivered over the radio interface between UTRAN and the mobile equipment.

• **Transfer delay**
  This parameter allows a connection to specify its delay requirements. It allows UTRAN to set internal transport options to meet the requirements.

• **Error requirements**
  Several parameters regarding error properties of the connection can be set and they are used to set various connection options in UTRAN for the radio link. If bit errors are allowed in delivered packets, UTRAN can deliver the packet and the receiver gets the data with reduced quality. This may be allowed in flows with video data for example. Parameters can also be set to indicate exactly how tolerable the connection is to bit errors in the data flow.

Although the traffic class gives a priority on the handling of flow requirements, the guaranteed properties of a bearer service are never compromised.

**Mobility**

User mobility is another main concept of UMTS. It is not a new concept from earlier systems as QoS is, but it has a heavy impact on the UMTS infrastructure. The goal of the mobility concept is that the service should be available everywhere and the service should continue even if the user moves out of range of the initial serving radio base station. The service should not be terminated just because the user moves around, and it should preferably not even impose lower connection quality. This is made possible by overprovisioning of resources in UTRAN and data splitting between different channels, described in Section 2.2.2. Overprovisioning means that adjacent cells reserves some resources for connections in a cell to handle connections that move between the cells. The controller modules of the cells are interconnected and handles the data rerouting transparently without user knowledge. UTRAN handling of mobility is further described in [2]. UMTS also contains functions that allow mobility on a larger scale by rerouting data flows within the UMTS core network.

### 2.2.2 UMTS techniques

The internal structure of the UMTS core network consists of several modules. A few of the more notable ones are gateways that connect UTRAN to the main core network and modules that handles the mobile users location within the network. The UMTS core network components are out of the scope of this thesis, and the rest of this section will focus on details of UTRAN.
UTRAN components

**Figure 2-4. Overview of UTRAN components and WCDMA cell structure.**
UTRAN is the radio access network that allows mobile users to connect to UMTS. It consists of several interconnected RNCs (radio network controllers) and Node Bs (base stations) as presented in Figure 2-4. A Node B is the logical representation of a base station in UMTS. In this figure, the core network label also includes the UMTS core network. The figure only gives a logical realization of connections in UTRAN. The actual physical realizations may have a different connection layout. The connection between two RNCs is usually not a direct wired link, and an RNC may be co-located with a base station.

From a geographical point of view, an area is covered for mobile access to UMTS by a radio base station. Several of such base stations are controlled by RNCs, which gives the system a larger total covered area. RNCs are in turn connected to mobile switching centers in the UMTS core network, which enables UMTS coverage of entire countries.

The Radio Network Controller (RNC) is the main controlling element of resources and data flows in UTRAN. It controls all transmissions over radio links to base stations, performs radio resource management and handles user movement in the network. The RNC is responsible for that UTRAN fulfill the requirements of its active bearer services and thus supervises all radio access bearer creation, operation and termination.

A Node B is a logical module of the system that contains and controls one or more WCDMA cells. Each cell usually corresponds to a single radio base station. In Figure 2-4 each Node B controls three cells.

The radio resource management functions of the RNC makes sure that enough physical resources exist in the radio access network to meet all connection QoS requirements as defined in their RABs. Examples of this are channel allocation to RABs and power allocation, both of which are described later in this section. Resource management in UTRAN is not only limited to the radio link, but since radio link bandwidth is the most scarce resource, this is usually where the main bottleneck is. However, it is still necessary to have control of other resources related to data flow handling as temporary conditions may cause over-
load in the system. An example of this is the computer processing resource that actually performs the functions of the system. This is described further in [7].

Radio resource management of a connection usually consists of a relatively static part executed at connection setup and a dynamic part during data transfer over the connection. The connection setup includes setting up a RAB (and usually a main bearer service handled by the UMTS core network), radio network access control and channel allocation control. This may also be a part of a handover that support mobility in the network as the new serving UTRAN needs to reserve resources for the connection. The dynamic part includes data packet scheduling on shared channels and power control of radios in both base stations and the users mobile radios. The reason why power control is necessary is described later in this section.

Channels
Connections from users mobile equipment to UTRAN is realized through entities called channels. A channel is a logical representation of how data is transmitted by radio links and the RNC interfaces between the data flow of the connection and the channels by using protocols defined for the UMTS radio link [6]. The protocols consist of a Radio Link Control (RLC) protocol that provides the connection with a view of the radio link as a normal wired link and a Medium Access Control (MAC) protocol that controls how the air is accessed by various radio links. The RLC protocol is a very important part since a radio link have very different behaviour compared to a wired link in a standard network. A radio link is subject to radio interference and mobility effects. In particular, the transmission error properties of a radio link is much more complex than for a normal wired link, and the RLC protocol employs techniques that overcome these differences.

In UTRANs radio link layer there are several different channel types which can be used in a connection. There are a few types of control channels that only communicates control information between UTRAN and mobile users equipment. There also exist dedicated channels where only one connection sends data, and shared channels where data from different connections are mixed by time-sharing. All channels have capability to exchange control information while dedicated and shared channels also have capability to carry user data.

There are two levels of channels, one logical and one physical. The physical channels are the data that are transmitted by the radios. Logical channels provide a higher level of abstraction. Packets from the logical channels are mapped to the physical channels before transmitting.

It is possible to split data for a connection between different logical channels, depending on what is most advantageous at the moment. The split data is then joined at the receiver before sent on. Channel splitting can be used both to change data flow rate and to improve connection quality by using channels transported via different base stations. The latter use of channel splitting is pictured in Figure 2-4 and may successfully be used when a user is close to a cell border and have poor connection quality. In the figure, the user data is split between two channels in a cell connected to Node B #1 and one channel in a cell connected to Node B #2. The channels on which data is split can be in different cells controlled by different RNCs, in which case the RNCs communicates and joins the split channels before
sending data to the UMTS core network. Efficient channel allocation to connections is discussed in [8].

**Code division**

A base station in UMTS have a specified frequency range in which it may transmit, and this limits the total bandwidth a single base station can support. To allow different mobile users to concurrently use this frequency range for transmitting data, WCDMA uses a technique called code division. The code division technique used in WCDMA is called direct sequence spreading. Codes with different spreading factors are used to separate the physical channels in UMTS transmitting at the same time in the same frequency band and allows different data rates on the channels. The value of the spreading factor for a code is inversely proportional to the data flow rate of the corresponding channel.

![Figure 2-5. UMTS spreading factor code tree.](image)

The different spreading factor codes can be organized into a bipartitioned tree where each branch is orthogonal to the other branches as presented in Figure 2-5. When one code have been allocated to a channel, no other codes on the same branch of the tree may be used. As an example from the figure, if the spreading factor code (1,1) was allocated to a channel the entire upper half of the tree would be forbidden for further assignment of codes. The details of spreading factors are described in [3]. The spreading factor at each depth of the tree corresponds to how many concurrent channels that can send, and the total bandwidth is equally divided by all the nodes at the level.

The user data bit stream is mapped to the raw channel data bit stream using the selected code for the channel. Each bit in the user bit stream is mapped to one instance of the code for the channel. Thus, for a code with spreading factor 4, each user data bit results in 4 channel data bits.

Each code is orthogonal to every other code on other branches in the spreading factor code tree. This means, in simple terms, that one active channel only adds to the background noise of all other channels. The data rate a user can receive depends on the signal-to-noise ratio for the channel. If one channel starts to use more power, the signal-to-noise ratio for all other channels decreases and they may need to raise their own sending power. Cells (base stations) also interfere with adjacent cells by raising the background interference level. An
important part of maintaining connection quality is therefore power control in base stations in different cells. The total base station power is also a fixed resource, which means that it must be shared among all currently active radio links in the base station. This limits the power available to each radio link which means that a link can be forced to reduce its data rate when the signal-to-noise ratio drops.

The maximum data flow rate in a UMTS frequency band is about 2 Mbps, but this data rate is rarely allocated to a single channel. Instead, the total data rate is split between several channels using codes with different spreading factors. Some of the data rate is usually allocated to control channels. The maximum user data rate is still very close to the maximum data rate since user data can be split on several channels. In a special operating mode, the control channels can be minimized and a single user can essentially get access to all the available data transfer ability in the frequency band.

Through the use of channel splitting and spreading factor codes it is in theory possible to provide a very wide range of bitrates to a user connection. From a practical point of view this needs to be limited, e.g. to simplify billing and charging for connections and to limit data processing requirements. The decisions of how to limit available bitrates are operator-dependent and not regulated in the standards.

2.3 Ns-2

Ns-2 is a network simulator that is originally based on the REAL network simulator [9] and have been further developed in the VINT project [10]. It is an open source software and commonly used by network researchers for evaluating and developing network related research. Since it is an open source software and additions are encouraged, many other network researchers have contributed to the development of the simulator.

The simulator is developed and runs in two programming languages, OTcl and C++. OTcl is an object oriented version of the Tcl language [14], which is an interpreted scripting language. The OTcl part of the simulator is mainly concerned with configuring the network prior to simulation start and the C++ part mainly handles the packet processing within the network when running a simulation. The scripting properties of OTcl lets developers quickly explore many network layouts, while the compiled C++ code effectively executes the large amounts of data processing required by packet handling during simulations.

A simulating environment in ns-2 consists of network elements that simulate the behaviour of a network and network connections that generates the data traffic used in the simulation. The network elements in ns-2 mainly consists of nodes and links between the nodes. At a higher detail level, each link and node consists of several internal elements to implement the behaviour. Nodes acts as routers and traffic generation points, and links acts as transfer
element between the nodes. In Figure 2-6 a simple network scenario is outlined. The circles denote the nodes in the network with links in between. In this scenario, node A have an agent that generates data and sends the data on its outgoing link with node B as the destination. When the data reaches node C, a decision have to be made which of the outgoing links to send the data on. This is handled by routing functions in the node.

A connection in ns-2 consists of a traffic generator, a traffic sink and a transport agent. All these components can be configured into a node by simple OTcl procedure calls. A traffic generator produces traffic in form of request to transport agents to send a certain amount of data. The transport agent then generates packets to transport the data to the receiver and inserts them in the network which propagates the packets to the receiver. When the packets arrive at the destination node, they are delivered to the other end of the transport agent, which announces the arrival of the data to the traffic sink. This is the way in which most data is propagated through a ns-2 network, although some protocols and networking scenarios allow different ways of data transport.

Ns-2 is an event driven simulator where each event is executed at a precise moment in simulated time. Events are processed synchronously and no events are executed concurrently. An event may be a packet arrival at a component input point or other scheduled activities that performs controlling actions in the simulator.

The simulator consists of a large collection of network elements that have been developed to suit a large number of different scenarios. The components can be described from two different views, the conceptual view and the implementation view. The conceptual view is mostly used when describing network scenarios, and the implementation view is used when implementing new components and by the simulator during simulations. In the conceptual view, overall behaviour of components are considered and not implementation details. The conceptual components consists of the nodes and links in a scenario as well as connection agents. The simulator’s conceptual view is mostly implemented in OTcl, with an interface to C++ where required. Nodes and links are viewed as single components with input lines and output lines and some internal properties that provide the behaviour of the components. In the implementation view, the conceptual objects are broken down into their internal building blocks, where all actual object connections are implemented through associations between objects. An example of this is a link, which in conceptual view is a single component which connects two nodes to each other and have internal bandwidth and delay prop-

Figure 2-6. A simple network scenario.
erties. In the implementation view of a link, presented in Figure 2-7, all the internal objects are visualized. The objects ending with T_ are mostly used for traces, and the ones that provide the functionality in this case is queue_, link_ and ttl_. They provide packet queueing, packet propagation delay and removal of old packets from further processing by the network.

The rest of this section describes a few of the core ns-2 components and their behaviour.

### 2.3.1 Node

A node is a main point of routing and start and end points for connections in ns-2. A node consists of several interconnected objects to create the functionality of a network node. An example of this is given in Figure 2-8. Here a hierarchical routing module have been installed in the node. This is an addition that allows ns-2 to run with a different node addressing scheme compared to the standard one, which uses another routing module. Hierarchical routing is required by certain network scenarios. The routing module in turn consists of other components that produce its routing behaviour. The ones pictured are all classifiers, but it may also be a few other components. The main difference between a node with the hierarchical routing and a normal node is the way in which the routing classifiers are organized, since they have different address routing modules.

#### RtModule

An RtModule contains a collection of objects that collectively can implement different kinds of node behaviour. The name stands for routing module, but it does not...
necessarily have to contain routing functionality as it is a highly configurable structure. This concept allows modular configuration of a node. The concept of RtModules as collection of node behaviour has recently (in the last year) been introduced in ns-2. In the future it will be the main way of introducing new behaviour into a node. It usually consists of classifiers, a few other connection elements and a routing agent. As long as they do not interfere with each other, there are no limit on how many routing modules that can be installed in a node which allows unlimited configuration abilities of a node.

Classifier
A classifier’s main responsibility in ns-2 is to provide packet routing functionality to a node. It acts like a demultiplexer and directs its input to different output channels based on a switching decision. When used as a packet router, the input is packets, the switching decision is handled by the routing code based on the input packet and the output channels are directions towards different destinations in the network.

Agents
Agents allow a node to act as a data source in a network. They are further described in Section 2.3.5.

2.3.2 Node/MobileNode
This is a special nodetype with a few additional node elements that allows simulation of all protocol layers and other components necessary for wireless communication. A Node/MobileNode is a normal node with an added capability as a wireless sender which means that it can also act as a base-station (link between radio interface and wired interface). It is currently implemented as a subclass extension to a normal node, but will be migrated to a plug-in routing module.

2.3.3 Link
This object models a standard wired cable connection between two nodes. Its connection components implements behaviour from the network layer and down. It simulates link propagation delays, network protocol layer queue and link errors (both partial and complete failure). On a link there is only one sender at all times, which means that data packets always are allowed to be sent over the link when they arrive at the link input.

2.3.4 Channel
This is a structure similar to a link since it connects nodes but it also allows simulation of a shared transport medium for data where sending collisions between different senders may occur. Compared to a normal link this allows simulation of all protocol layers down to the MAC layer. There is also a special subclass named Channel/WirelessChannel which bases the propagation delay of packets in the channel on the distance between the two commun-

---

1. In ns-2 the Tcl convention of naming derived classes is to put the base class name first, then a ‘/’ and then the name of the derived class. Here we can see that MobileNode is derived from Node, forming the full name of the class: Node/MobileNode.
cating nodes. A normal channel uses a fixed propagation delay, and this addition is required in wireless simulations where distances between nodes may vary greatly.

2.3.5 Agent
An agent can be both an active and a reactive component in a network node that implements network behaviour. As a comparison, links and nodes are only reactive components which react to incoming packets and apply their behaviour to the packet. One example are transport agents like TCP and UDP. They react to external orders to send data but may do it both by generating necessary packets for data transport and generating connection control packets. Another example is routing agents which can perform dynamic route calculations. They actively generate packets and send them to other agents of the same type to determine routes through the network. In the case of transport agents, they require external components that generate data to send.

The basic TCP agent in ns-2 is a simple version of a TCP agent. It immediately sends acknowledgement packets when data arrive. As a consequence, the round trip time of data packets and their acknowledgment packets are important when the TCP determines its send rate in ns-2. When the acknowledgment packets arrive, the send mechanism is triggered and new data is output at the rate specified by the congestion window.

2.3.6 Traffic generators
Traffic generators are the objects that generate data in the network. They can be of two types, either a data source that sends data according to a traffic distribution or a source that simulates an application. Two examples of distribution-type traffic generators are constant bit rate (CBR) and exponentially distributed traffic in terms of active and inactive sending. Two examples of simulated applications are FTP-transfer and a Telnet-connection. The traffic generators use transport agents to insert the data into the simulated network.
3 Component design

In this chapter the design of the components that include RRM functionality and adaptive streaming into ns-2 are outlined.

3.1 Ns-2 components

Ns-2 is an extensive network simulator which contains a lot of already developed structures, components and mechanisms (e.g. protocols). The components have been used in many projects and their functional correctness have been proven. Therefore the existing components should be reused wherever possible, both to reduce development time and to reduce the risk for errors. The main components used was described in Section 2.3.1-2.3.6.

A main requirement for the ns-2 components produced as part of the thesis is that they should be simple to include in network scenarios. Components in UTRAN that require extensive setup procedures should therefore be avoided if possible.

Since most of the UTRAN RRM functionality is located in the RNC component of UMTS, it is reasonable to use this component as the base for development. It is a component that is part of a node in the network, and should thus be placed as an internal component of a node. The best way of doing this is to implement a routing module with RNC functionality.

3.2 Limitations

The RRM functionality in UTRAN covers a large area, and to include it all would be impossible within the thesis time-schedule. Therefore a decision must be made which parts of UTRAN RRM to include. The requirement to limit the thesis scope becomes obvious by looking at the sheer size of the document that describes the RRM protocol in UMTS [5]; it is almost a thousand pages long. The final design of the components must be made with care, so that they can be extended in the future to include the functionality currently left out of the design.

Since the RNC component needs an underlying layer to transport data to mobile users, and to handle resources for, an initial limitation of the connection was made together with the thesis that introduces the lower layer in ns-2. A decision was made to limit connections to a single channel and base station to reduce problem complexity. This removes the channel type selection and channel allocation normally required by a connection. From the RNC point of view, the radio link (RLC) can be viewed as an abstraction of the lower details of a connection.

To reduce dependency on the thesis that develops the radio link functionality [11], a separate layer using standard ns-2 links have to be constructed. It was also cooperatively decided to use the ns-2 component Node/Mobilenode as a base for the RLC and MAC pro-
tocols. This nodetype has an internal structure that already includes several components that are required by full UTRAN RRM, but the components have limited functionality. The alternative would have been to develop a new version of a ns-2 link, but this would have required much more development to support future extensions.

In UTRAN, specific spreading factor codes are used to regulate data rate of channels as described in Section 2.2.2. By allowing a radio link to use rates that can be combined from several codes with different spreading factors, the link can simulate channel spreading. With an operator-dependent option to limit available bitrates, it becomes easier to regulate allowed flow rates with direct bitrates instead of using the spreading factors. To simulate the likely limitations UMTS operators will place on allowed bitrates for connections, allowed flow rates are taken from a ‘RAB-table’. This table should contain an approximation of the data rates a user connection may get through UTRAN, and should be configurable in a network scenario.

In UMTS, power considerations must be made during radio transmission to maintain a low background interference level in the air. Interference levels can also have an impact on allowed channel data rates. Ns-2 have a very rudimentary interface for power considerations on transmissions, and it was decided to ignore all power considerations and concentrate on higher level RRM functions.

On the flow QoS level, it was decided to limit flow- and packet-handling to best-effort traffic. A reason was that the time to develop both infrastructure and control algorithms was limited, and to complete a working subsystem in time was considered to be a better option than a full system that was only partly completed.

Handling of user mobility in the system is also ignored. The main purpose of the developed module is to include RRM handling of flows in a node of ns-2 to provide upper layers with a more accurate view of UTRAN than standard ns-2 components can provide. A purpose of mobility functions in UMTS is to make mobility seamless from a user point of view. Thus, from upper layers, mobility effects in UTRAN should be nearly invisible. Together with time considerations, this lead to exclusion of mobility functions from the beginning. However, an opening was left if it was later determined that it was possible to include mobility within the time-schedule.

When a user connects to a network, a lot of RRM functions are executed in an RNC. In ns-2, it is complicated to do this since all parts of a network scenario, including connections, are completely defined before the simulation starts. This makes it hard to implement connection setups. The extra work required to execute connection setups during a simulation is considered to be too time-consuming to be implemented. Instead, connections are to be dynamically detected and initialized by the RNC component.

So far a lot of things from UMTS have been excluded, what is still left of the RRM functionality in UTRAN?

Since many connections share the available data rate in a radio base station, it is necessary to control their access to the radio links. Flow rate management is required to let all flows get their fair share of the available data rate. Since data flow rates of connections may vary during network scenarios, it is required that the flow rate handling adapts to changes in data
flow rates and efficiently allocates the available bandwidth in the radio link layer, since this usually is the most scarce resource in the entire system.

The basic demands on the RNC component then are:

- Implement adaptive flow-control in the RNC. An infrastructure of ns-2 components and an algorithm for flow-control should be developed.
- Implement a substitution sublayer that can be used instead of the normal RLC/MAC layers.

The impact the algorithms have on the architecture is described in Section 3.3 and the algorithms themselves are explained in Section 3.4.

3.3 RNC component architecture

The RNC component have been implemented as a routing module (RtModule). It currently does not include any routing code at all but this may well change in future extensions, e.g. when extending the RNC to handle several base stations or using inter-RNC handovers. Since it does not include any routing functionality by itself, it requires that the node contains an additional addressing routing module. Fortunately, such routing modules already exists.

The completed component architecture of the RNC routing module is presented in Figure 3-1, and below it the reasoning that lead to this construction is outlined.

![Figure 3-1. RNC connections and internal components.](image)

The main responsibilities of the flow rate handling algorithm is to:

1. Adjust rates for each flow based on available traffic and total rate limitations.
2. Adjust total rate from RNC to avoid overloading base station by reducing flow rates.

To adjust rates per flow, the flows need to be separated in the RNC. This means that a mechanism for deciding which flow a packet belongs to is necessary at the entry point of the RNC. The typical component for that function in ns-2 is a classifier. However, the currently implemented classifiers have a behaviour which is not desired for this classifier. When the simulator starts, it adds routing information to all classifiers in the network, which allows packets to find their way. All implemented classifiers are address classifiers that should route packets, so for them it is fully acceptable. Since the required classifier is not concerned with routing information, this behaviour have to be changed and means that a new classifier have to be developed, the FlowClassifier. Most of the other functionality
of the classifier can remain, and just the part that adds packet routing destinations to the outgoing ports have to be changed.

Conceptually the link layers for all flows are separated and begins in the RNC. This means that it is preferable to let each flow have an independent connection through the RNC down to the sublayer. In the current implementation, there is only one sublayer that joins all the flows, but this may change with further development. By letting each flow have its own separate exit point from the RNC, they can be split in the future.

The RNC component should be able to handle two cases of network configuration, both when the underlying transport structures between nodes are normal links and when it is a fully implemented RLC/MAC radio link layer. To interface with and support both these different sublayers a main interface that never changes is derived. From this main interface, the specific sublayers are derived and their specific behaviour is implemented. This structure for the sublayer allows for easy extension in the future if some other variant of the sublayer is required. The main interface handle the common operations of the sublayers, e.g. rate updates. The derived sublayers handle the parts that differs, e.g. setup and installing new connections in the sublayer. Since no connection setup is performed, the RNC needs another way of detecting connections. We only have to worry about a single base station (no mobility), and as a result it can be done by looking at packets that pass the base station.

The concept of the base station can be replaced by the sublayer in the current implementation that have one base station per RNC, and the functions that detect and adds new connections can be added in the sublayer. Since the structure of the sublayer is different in the two cases, the code have to reside in the derived sublayers.

Each flow in the RNC is further subdivided. The division is described in Figure 3-2. At the head of the RNC flow is a control object for the flow. It handles all information regarding the flow and is the interface between actions performed on the flow and the RNC flow handling algorithms that controls the actions. A normal ns-2 queue object is then added and it is used to temporarily store packets in the RNC flow when they cannot be immediately sent to the sublayer. To allow the type of queue to change it have been added as a stand-alone component in the flow instead of as a flow-internal data-structure. This allows for greater flexibility in deciding queueing behaviour in the RNC. To still allow the main RNC flow control object to control packet departures, an extra connection component is added to the RncFlow chain. Its only function is to report to the main flow control object that a packet is leaving the flow.

The second part of the flow rate handling mechanism is handled by the packet scheduling algorithm. By inspecting buffer sizes in the sublayer, the packet scheduler can refrain from adding more load to the sublayer at overload situations. This requires some support from

![Figure 3-2. Internal component structure of a RncFlow.](image-url)
the sublayer, and have been enabled through a interface negotiation with the RLC/MAC developer. The sublayer using normal links already have support for this.

### 3.3.1 Properties of implementation

The implemented component is limited compared to a full RNC. Most of the effort have been concentrated in providing a basic infrastructure that supports future development of RNC behaviour. As a result, the current implementation have some limitations that need to be considered when constructing network scenarios.

- Only the best-effort traffic-class is handled which means that there are no delay requirements in UTRAN. Scenarios need to be careful with the queue-length assignment in the RNC component, since too long queues will cause long delays and little reactivity.
- The flow rate algorithm, described in Section 3.4.1, does not maximize the utility product perfectly. During congestion, it may free some bandwidth that is not immediately consumed by a rate allocation.
- Flows get bandwidth strictly on demand, which means that someone can block the system and other users by sending excessive amounts of data.
- There is no limit on the number of flows from a buffer space point of view. The RNC is assumed to have enough memory to provide buffer space for all accepted flows.
- A side effect of the way flows are detected and installed in the RNC is that initial packets of a connection are not subject to RNC flow control. Temporary overload situations may therefore occur in the RNC and its base stations. Such overloads are currently not specially treated, because when the new flows are established, they start out with a low allowed data rate which will reduce or totally nullify any overload situations caused by the non-flow controlled data. If call setup is implemented, this problem will go away entirely.
- RNC flows can be created, but they can never be destroyed. This is a property which limits the number of consecutive connections through the implemented RNC. This limitation will have to be removed to allow large-scale simulations of connections by adding call setup, which allows both connection creation and connection closing.
- Delay when using the normal link sublayer is fixed whereas it depends on packet-length in the RLC/MAC sublayer. This makes a big difference in comparing up- and down-link. Small packets, like acknowledgements, get very short delay in the RLC/MAC sublayer compared to the link sublayer. If a packet is too large, the RLC splits it into several packets and transmits them one by one. Small data packets only use one or a few RLC packets which means they get a short delay. In the link sublayer, the delay is fixed regardless of packet size but the transmission output time varies. This must be considered when dealing with asymmetric data traffic, e.g TCP flows which have large data packets going in one direction and small acknowledgments going in the other direction.

### 3.4 Radio resource management algorithms

The RNC component uses one algorithm to assign available flow rate to flows, and one algorithm that schedules packets from different flows for transmission by the sublayer.
3.4.1 Flow rate algorithm

The algorithm that sets the allowed flow rates is based on a cooperative game theory developed in economic studies [12]. The theory gives a solution to how two rational persons bargaining for a resource can agree on a fair bargaining division of the resource, if they know the value of each resource division for the other person. Each resource division between the persons are associated with a utility value of how useful the resource division is to them. One of the properties of such a fair bargaining solution is that it maximizes the product of utilities for the persons. The theory is presented as a 2-person bargaining problem, but the solution given in the theory scales without problems to any number of persons.

To apply the theory to the problem of fairly allocating a restricted resource (the total radio link bandwidth) to several data flows, a few preparations have to be made.

An assumption is made that users like to get as much bandwidth as they desire. This allows us to choose a utility function in the bargaining problem that satisfies the requirements of the theory. The chosen utility function is the ratio between the allocated flow rate ($R_i$), and the bitrate received at the RNC($r_i$) for each flow. The ratio is chosen instead of the plain rate to match the theory requirements.

$$u_i = \frac{R_i}{r_i}$$

A fair solution for dividing the bandwidth resource between the competing flows are then reached by maximizing the product of the utilities for all the flows:

$$\max \prod u_i$$

As long as there is available bandwidth on the radio links, all flows are allocated rates from the RAB-table just above the incoming rate of the flow. This preserves the scarce bandwidth resource, $B$, of the radio link by not allocating excessive rate to a flow.

$$\sum R_i \leq B$$

The utility product may not be maximized in this case, but it preserves the bandwidth resource and still meets the connection demands of bandwidth resource.

When there comes in more data on the flows than the radio link can handle the bandwidth resource have to be reassigned to the flows. This time, not all flows can get enough bandwidth to meet their demands. The algorithm then selects the distribution of bandwidth resource between the flows that maximizes the utility product for all flows.

An example of how the algorithm works follows here:

When there is no congestion, flows get a utility of more than 1 (the rate allocated to the flow is higher than the data rate of the flow) and the flow rate is only reduced if the utility remains above one when decreasing the rate. At congestion flows that will get the highest utility after rate decrease are decreased first since this maximizes the utility product.

To make the flow rate algorithm adaptive, the allowed flow rates are updated at fixed time intervals. The time interval is user-defined and can be configured in a network scenario.
A limitation of how to maximize the product is that the allocated rate to a flow may only change stepwise in the RAB-table. This restriction have been introduced to give flows a chance of reducing the data rate to the RNC instead of just chopping them off when overload conditions occur.

To change the flow rate of a connection in UMTS takes time, since all parts involved in the connection needs to be reconfigured for the new rate. These include the connecting channels, the base station and the user equipment. Control messages need to be exchanged and the change must be synchronized between all components. To simulate this, there is a configurable delay added between the rate change decision and the introduction of the rate change in the RNC flow control algorithm.

The developed algorithm is susceptible to bandwidth hogging, since a “bad” user can send as much data as it can produce which reduces the bandwidth available to flows that only sends as much as they are allowed by the RNC. This may lead to stepwise down-prioritizing in the RNC of the well-behaving flows. The misbehaving flow will have the lowest utility value, and the utility product of the flows will be maximized if that flow gets as much bandwidth as possible. The “bad” user breaks the fair bargaining assumption of the theory. A solution might be to use a different utility function, maybe a function that adapts over time to flow rate behaviour.

The total bandwidth in a radio base station, $B$, is available as a configuration parameter in network scenarios. In a real-world situation, the radio resource management algorithm would have access to all the resources of the base station. In this limited version where only best-effort traffic is managed, the bandwidth limit can be viewed as higher priority traffic from the other traffic classes, which have reserved the rest of the total base station bandwidth.

### 3.4.2 Packet scheduling algorithm

The purpose of the packet scheduling algorithm is to provide the sublayer with packets to send and to limit the data traffic load in the sublayer. Sometimes the radio link conditions are bad and the sublayer needs extra time to transmit data. The packet scheduling algorithm should check for this and avoid drowning the sublayer in packets, both on a per-flow basis and on a total sublayer basis. It should give all competing connections a fair chance of getting access to the radio links, and this is facilitated through the use of the fairly allocated flow rates from the flow rate algorithm.

The next flow to send are selected by using a token bucket-like algorithm for each flow, and comparing the number of tokens in each bucket. The token bucket rate of each flow is equal to the current flow rate, and the size of the token bucket equals sending for two seconds at the flow rate. The flow that has the largest amount of tokens in its buckets, has a packet to send and is not blocked by sublayer congestion are then selected for sending.

The packet departure algorithm consists of two separate steps:

- At specified time intervals, it checks if the sublayer can receive more packets. If so, let the next flow in turn send a packet.
• When a packet arrives on a flow it is directed towards the flow queue. Then the algorithm checks if it is this flow's turn to send and if the sublayer can receive more packets. If these checks are passed, the flow is allowed to send one packet to the sublayer (this may be the incoming packet if the flow queue was initially empty).

The second step of the algorithm helps to speed up processing in some cases, e.g. when there is only one active flow. It is also useful if the time interval check is made adaptive. This was tried early in the implementation stage, but was dropped because of implementation complexity.

3.5 Future compatibility

The use of Node/MobileNode as base for the radio link gives access to the physical layers and allows introduction of power management.

Mobility functions can be implemented through extension of the RNC routing module with an active routing agent. This would also allow dynamic connection setup.

Multiple channels can be supported by adding an additional packet classifier at the back end of the RNC module and changing the packet scheduling algorithm. The input to the classifier would be all packet flows, the switching would be controlled by the packet scheduling algorithm and the outputs from the classifier would be the different channels.

3.6 Adaptive streamer architecture

To implement an adaptive streaming application, two main approaches to adaption can be taken [13]. They are reactive adaption and passive adaption. In reactive adaption, the application asks the network if more bandwidth is available by sending more data and reacting to how the network responds to the extra data. This approach requires some form of feedback information from the network. In passive adaption, a flow with a fixed bit rate is sent and the receiver makes use of all data that arrives. It is also possible to use a mix of these two approaches.

A decision was made to use a reactive approach. The RNC currently only supports best-effort traffic, and a streaming application needs to adapt its sending rate to work properly together with the RNC. A passive approach would not work very well since it appears to the RNC as a CBR data source. A CBR data source generates a typical misbehaving flow from the RRM algorithm point of view, as the flow pay no attention to the conditions of a connection. As noted in Section 3.4.1, this may cause problems for other RNC flows.

Since ns-2 already contains some application-type data sources, see Section 2.3.6, the adaptive streamer will only be another variant of this type.

The reactive adaption method used is rate shaping adaption. Rate shaping means that an algorithm is used to control the output rate of the streamer based on feedback from the network. In the current implementation, the streamer periodically increases its sending rate and checks for how the network reacts to the increased rate. To simplify simulation of different types of rate shaping, a helper module in the streaming sender called a rate handler is used. If a new rate shaping type is wanted, the only change needed to the streaming sender code is to implement the new functionality in the rate handler module. The currently implemented rate shaping methods are stepwise and table based shaping. With stepwise
shaping, the output rate of the streamer is changed in uniform steps. The table based shaping method selects its send rates from a user-specified table. Existing rate shaping functions may be configured into a network scenario using simple functions during scenario setup. The send mechanism of the streamer reports the network conditions to the rate handler module and lets it handle the rate shaping decisions. A visualization of the internal structure of the streamer and how it operates is presented in Figure 3-3.

The algorithm that handles the feedback is divided in two parts, one in the receiver and one in the sender.

The receiver have a playback buffer that receives data and it also extracts data from the buffer with the current streaming rate. Before playback starts, the buffer is initially filled with data. Packets that miss their playback time causes the receiver to inform the sender of the miss and request a rate reduction. To handle the playback buffer, the streaming application needs to transport information of rate changes in the data stream from sender to receiver. Based on ns-2 documentation, an early decision was made to use an existing ns-2 TCP-implementation as transport technique as it was the only one that could support transfer of application data. It was later discovered through source code mining that more protocols supported transfer of application data packets, but the technique used differs between the protocols and time considerations ruled out adaption to the other protocols.

The sender have an algorithm that monitors the total round trip time (RTT) for packets via acknowledgement packets (acks). If this exceeds a threshold value, the sender should reduce its send rate as this is an indicator of network congestion. Since TCP was used as transport protocol, the original thought was to monitor its acks for this purpose. However, this was made impossible by the ns-2 interface to the TCP-agent, and this is a proper behaviour since the reason for network layering is to shield upper layers from the data transport details. The approach could have been used if a simpler transport protocol had been used,

---

**Figure 3-3. Internal components and connections in the adaptive streamer.**
but this would also have increased the complexity of the send mechanism of the streamer. Instead of using TCP acknowledgements, the application uses an application-level ack.
4 Evaluation of the RNC component

4.1 Impact of implementation properties on simulation results

This section describes how the implementation limitations may affect the simulation results.

A real system must support all traffic classes, which requires a different RRM algorithm since it needs to support delay sensitivity. From the traffic rate point of view, the extension to include other traffic classes in the algorithm will not cause any major changes in the presented simulation results. The main effect will be that the delay variation will be larger for best-effort traffic. This results from that other traffic classes have bounds on system delays, which forces the scheduling algorithms from time to time to select delay-sensitive packets for transmission before any best-effort packets. These effects will be randomly distributed, and can cause large delay variations for best-effort traffic. At some points in time, a lot of delay sensitive packets may need to be transmitted, but when they have been transmitted there is still room for the best-effort traffic to achieve the required throughput.

The lack of true connection setup will cause a transient behaviour at the beginning of connections since there is no initial allocation of resources, i.e. bandwidth. Data that does not conform to the allocated resources for the connection are temporarily queued in the RNC, initially causing a transfer delay (provided that the data flow rate is larger than the allocated flow rate). How long transients affect the connection varies between different scenarios and must be evaluated for each case individually.

4.2 General points for simulations

First a little comment regarding the reliability of simulation results:

The scenario settings may have a great impact on results produced during simulations. Only a few settings have been changed between the different simulations, and most of the settings have been fixed. If the applied settings are different from those used in a real system, the simulation results may be unreliable. This means that conclusions drawn from those results may not be accurate. As a consequence of this the presented results, numeric and graphic, can not be directly compared to other simulations. The conclusions are only based on differences in results caused by the different settings.
Several different network scenarios have been used to evaluate different aspects of the implementation. They represent different ways of connecting the network elements in ns-2. The four scenarios are:

A. RNC have a link sublayer, the connecting network is only a delay
B. RNC have a RLC sublayer, the connecting network is only a delay
C. RNC have a link sublayer, there is an interference network between the sender and the RNC.
D. RNC have a RLC sublayer, there is an interference network between the sender and the RNC.

In all scenarios, the sender is placed in the main core network and the receiver is a mobile user that is connected to UMTS.

In scenarios A and B, there may be more than one sender and receiver. In such cases, they use parallel links when connecting to the RNC.

The network layout of scenarios A and D are presented in Figure 4-1. The sending nodes are on the left side of the figure, and the receiving nodes are on the right side. In scenario A, single standard ns-2 links are used to connect nodes to the RNC, and the RNC is in the link sublayer mode. In scenario D, the sending node is connected to the RNC through an interference network. This network consists of several links, nodes and data flows that create variable delays. It tries to simulate congestion effects of the Internet or any other best-effort packet-switched network. From the RNC there is a radio link to the receiver and the RNC is in the RLC sublayer mode.

In all these cases, the RNC flow control can effectively be disabled by only allowing a single flow rate. This use of the RNC is explained where needed in the simulations.

The following network settings have been used for all scenarios (except where otherwise noticed):

- Wired network delay of 100 ms.
- Links in the wired net have significantly higher bandwidth (10 Mbps) compared to the simulated wireless net, the radio link is limited to a maximum rate of 256 kbps except where otherwise noted. Different RAB-tables have also been used, and are
explained for each simulation set. The limit of the max rate used by the RNC control algorithm can be viewed as higher priority traffic from other UMTS traffic classes that is concurrently transmitted.

- The uplink rate in the RNC link sublayer mode is fixed at 32 kbps.
- The allowed flow rates are updated by the RNC flow rate algorithm every two seconds. There is a one second delay from a rate change decision until the radio link have assumed the new rate.
- The simple TCP agents of ns-2 have been used. They use immediate acknowledgment, which affects the results of some simulations.

In the functional evaluation, flow rates in the figures are calculated from the raw simulation data using an algorithm that immediately detects rate changes. This is possible since the simulation scenarios uses a simple network layout and the simple ns-2 TCP agents. In the other evaluations, the network scenarios are more complex and the algorithm does not determine the flow rates from the raw data correctly. In these scenarios, a sliding window algorithm with a 4 second window is used to determine flow rates.

Two types of flows are used as data loads in most of the simulations, FTP and CBR. Both use the TCP transport protocol. CBR gives the TCP sender data at a constant fixed rate, while FTP lets TCP send as much as it can. The most significant difference between them is that when CBR gives data packets to the TCP sender, it triggers the send mechanism and causes a send impulse. FTP only triggers a send impulse when it is first started and then leaves TCP to itself. At congestion, FTP flows will generate less traffic, while CBR flows that continuously triggers the TCP send mechanism will generate traffic almost at the CBR rate.

### 4.3 Functional evaluation

The functional evaluation consists of simulations that ascertain that the RNC handling algorithms works and to find out inherent properties of the implementation that have to be considered in other simulations. To make it easy to see such properties, the simplest possible network scenario is used, scenario A. A RAB-table with about 10 different rates have
been used to check the rate adaptation algorithm, and the total radio link rate is limited to 128 kbps.

![Graph of RNC adaptation to a 96 kbps CBR flow.]

**Figure 4-2. RNC adaptation of flow rate to a 96 kbps CBR flow.**

In Figure 4-2 the RNC flow rate adaptation to a constant bit rate data source is presented. The RNC flow rate markers denote changes in the RNC flow rate, and the rate is valid until the next marker. When the allowed RNC flow rate is lower than the incoming flow rate, the RNC buffers data in internal queues. To empty a flow queue, the RNC may temporarily allow a higher flow rate for a short while if there is enough free radio resource. This is what happens at T=7 seconds. At T=17 seconds, the queue have been emptied, and at T=42 seconds the RNC reduces the allowed flow rate to the actual data flow rate. The CBR rate have been selected to match one of the rates in the RAB-table.

In Figure 4-3, a TCP traffic load is regulated by the RNC. The TCP sender simulates a file transfer, and increases its send rate until it reaches the maximum allowed rate of the flow.

![Graph of RNC adaptation to a TCP file transfer flow.]

**Figure 4-3. RNC adaptation of flow rate to a TCP file transfer flow.**
It may appear that the TCP sender follows the allowed flow rate perfectly, but in a close-up a small delay can be seen. The TCP sender responds to the new flow rate immediately when the first packet transmitted with the new flow rate have been acknowledged. This is a result of the TCP senders control algorithm, it immediately reacts to reception with an acknowledgment packet. The network delay in the scenario is constant, which allows TCP to predict its allowed send rate with 100% accuracy.

Packets that are sent through the RNC are initially delayed a lot as presented in Figure 4-4,

![Figure 4-4. Packet delay at connection startup for a TCP flow.](image)

which shows the packet transfer delays of the TCP scenario. In the figure, the solid line is the packet delay, which uses the scale on the left side of the figure. The dashed line is the received flow rate and it uses the scale on the right side of the figure. The initial delay is a connection startup transient caused by a combination of absence of call setup, TCP slow-start and that allowed flow rates are slowly increased by the flow rate algorithm. The transient is not limited to only TCP, it affects all transport protocols that adapts to network conditions by increasing their initial send rate. The buffered data at connection startup affects the delay for later data as well. With the higher flow rates, the RNC buffers contain the same amount of data that was buffered in the beginning, but the higher flow rates result in shorter waiting time for packets in the buffers.
In Figure 4-5, a FTP flow using TCP as transport protocol is started. At T=60 seconds and T=120 seconds extra load in form of 64 kbps CBR flows are started. When the extra load is added to the RNC, the algorithm distributes the available base station bandwidth on a requirement basis. Since the added flows are of the CBR type, the stepwise down-prioritizing of the FTP flow, as described in Section 3.4.1, is visible after the second CBR flow is started. At T<60 s, the flow rate algorithm allocates all the available bitrate to the FTP flow. At T=60 s, a new flow requires bandwidth and the flow rate algorithm reduces the FTP flow rate one step. At T=120 s yet another flow requires bandwidth, and the flow algorithm reduces the well-behaving FTP flow rate one step further. After about 10 seconds at this rate, the CBR flows takes over and the stepwise down-prioritizing begins.

4.4 Evaluation of link sublayer versus RLC sublayer

To simulate behaviour of the air interface in a radio link, errors are introduced in the radio link packet flow. Both flat error rates (error in every nth packet) and burst error rates are considered. Burst errors are interesting since the air interface depends on lot of things, e.g. multipath interference, which can cause a burst error behaviour of the radio link. In this evaluation, burst errors are realized through a Markov chain. The Markov chain consists of two states with different error probabilities and probabilities of switching between the two states. A more detailed description of how this works can be found in [11]. In UMTS the error rate of a connection can be set as a QoS parameter, and the physical layer is adapted to meet the parameters based on actual measurements of errors.

Packet errors in these scenarios are generated using random number generators, and the generators use different seeds for each simulation. This causes subtle differences in the behaviour for each simulation, including differences between two runs of the same scenario.

The evaluation consists of two parts. First the properties of the RLC sublayer are investigated, and then a comparison is made with the link sublayer.
To evaluate the behaviour of the RLC sublayer, delay distributions for 1000 byte TCP packets transmitted over the RLC sublayer are produced for several different error models. The packet error rates stated in the simulations are the error rates in the RLC layer and not at the TCP level. Each TCP packet gets fragmented into several smaller packets by the RLC layer. For these simulations, network scenario B is used.

The simulation results for the different error rates are presented in Figure 4-6.

In Figure 4-6a, a delay distribution with no error packets is included for comparison. It shows that packets get an offset transfer delay of up to 0.8 seconds, which is caused by queueing effects in the simulated UTRAN(RNC and RLC). There are a few packets that gets a higher delay, up to 3.2 seconds, but this is caused by the connection startup transient. It occurs in all the simulations in this section, but is only noticeable in this simulation. The actual queueing delay for packets in the different scenarios can vary a lot and depends on the interaction between queue lengths, allowed flow rate, actual data flow rate and radio link packet errors.
With a 10% packet error rate, the delay distribution looks as in Figure 4-6b. The delays appear to be following some kind of exponential distribution. It is important to remember that there is a delay offset caused by queue effects in UTRAN. The average transfer delay for a 1000-byte package seems to be just below 2.5 seconds. The large variation in delay for different packets is caused by packet resending in the RLC sublayer. When a packet is received with error, it can not be used and it is retransmitted by the RLC. Since retransmissions also are subject to packet errors, a packet may be retransmitted several times and this causes the long delays.

In Figure 4-6d, the delay distribution for a burst error rate scenario with an average error rate of 10% is presented, and it should be compared with Figure 4-6b. The difference compared to 10% flat error rate is not very large, but burst errors result in a slightly more extended distribution towards longer delays. This corresponds to that the RLC layer have had a harder time to transmit a few packets because of the bursty errors. Because of the small difference it is probably possible to use the simple flat rate errors without getting large changes in simulation behaviour.

A 20% error rate is too high in most cases, but it gives a good idea to what might happen with poor connection quality. This delay distribution is presented in the lower left corner of Figure 4-6. It is stretched out, with a center around 4 seconds and a maximum delay of about 8 seconds. In such a case, the RNC should be triggered to try to improve connection quality, e.g. by channel splitting via two cells.

The average and 95 percentile delays of the delay distribution simulations are presented in Table 4-1. The 95 percentile delay is the maximum delay that 95% of the packets experience, i.e. 5% of the packets experience longer delay.

<table>
<thead>
<tr>
<th>Error rate</th>
<th>Average delay</th>
<th>95 percentile delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td>0.73 s</td>
<td>0.75 s</td>
</tr>
<tr>
<td>10%</td>
<td>2.37 s</td>
<td>3.25 s</td>
</tr>
<tr>
<td>10% burst</td>
<td>2.41 s</td>
<td>3.57 s</td>
</tr>
<tr>
<td>20%</td>
<td>3.85 s</td>
<td>5.67 s</td>
</tr>
</tbody>
</table>

Note that the absolute values in this table are dependent on RNC queue-settings. The average delay for 10% burst error rate is slightly longer but almost the same as the 10% flat error rate, but the 95 percentile delay is significantly longer. This means that more packets have been unlucky and have been retransmitted several times with a bursty error rate than with the flat error rate.

To compare the link sublayer with the RLC sublayer, a simulation is made to analyze send rates and total link throughput differences. The comparison is made with the 10% flat error rate RLC link and a normal link sublayer with the same average delay.

To eliminate uplink differences between RLC sublayer and link sublayer, the delay of the link sublayer is adjusted so the total RTT time for packets and their acks in TCP are the same. If only the one-way delay was considered, the link sublayer would be guaranteed to have a lower send rate and throughput. The TCP ack packets are small and propagates faster.
through the RLC sublayer resulting in a shorter return delay, while the delay is constant in
the link sublayer. By using the RTT measurement instead, the TCP algorithm should not
cause any differences in rate. Only differences between the sublayers should affect the
throughput. The RNC flow rate handling must also be eliminated by setting a fixed flow
rate of 128 kbps.

The results of the comparative simulations of the link sublayer and the RLC sublayer are
given in Figure 4-7a and Figure 4-7b, respectively. The solid lines are the radio link
throughput rates, and the dashed lines are the accumulated data throughput. It is clear that
the RLC sublayer in Figure 4-7b only achieves about half the rate and throughput of the
link sublayer. This can be explained by that it is much easier for the TCP sender to get a

![Figure 4-7a. Radio link throughput in the link sublayer.](image1)

![Figure 4-7b. Radio link throughput in the RLC sublayer.](image2)
good estimate of the RTT in the link sublayer, compared to the varying RTT of the RLC sublayer. The more stable link sublayer thus allows a higher throughput rate.

Since there is such a large difference between the results when using the different sublayers, the link sublayer does not appear to be an alternative for simulations. However, it can be useful in certain situations when the complexity of the RLC sublayer is not necessary. An example where it can be used is the evaluation of basic RNC behaviour, which only needs some data to be transferred and the quantity does not matter. The RLC sublayer introduces large variations in flow rates and delays, and this may hide other effects in the network. The link sublayer can not be used to evaluate situations where data quantity matters, since it allows too high flow rates. Generally speaking, the decision to use the link sublayer or the RLC sublayer depends to a large extent on the network scenario and what the simulation results are to be used for.

It may be possible to produce a better approximation of the RLC sublayer by using a packet delay distribution in the link sublayer instead of a fixed delay.

4.5 Evaluation of adaptive radio resource management effects

These simulations are performed to show the functionality added to the system by the adaptive flow control algorithm. Network scenario A is used for the simulations, with simulated web-browsing traffic as traffic load. Web-browsing traffic is simulated by pareto-type distribution senders and the following pareto parameters: The average think time between page downloads is 30 seconds, and the average size of pages is 16 kB. The source data rate of page downloads is 64 kbps, and the shape parameter of the pareto distribution is 1.3.

To show the adaptive RRM effects in a network, two network scenarios with different characteristics are compared. In the first scenario, the RRM functionality of the RNC is not adaptive. The RNC uses a fixed rate of 128 kbps for connections. The total link capacity is 256 kbps, and this means that two simultaneous connections over the radio link are allowed. In the second scenario, the RRM functionality in the RNC is adaptive, with a few RAB-rates (~5). The traffic load consists of 8 web-browsing flows in the same total channel bandwidth as the first scenario, 256 kbps.
The simulations of the two scenarios are presented in Figure 4-8a and Figure 4-8b. In both figures, the throughput rate is the solid line and the accumulated data is the dashed line.

The first scenario with two flows are presented in Figure 4-8a, and it is clear that two flows are not enough to keep the radio channel occupied. There are several periods of low or no activity, and there is only one time period when both of the flows are receiving pages (T=190 s). The throughput rate graph gives an insight to how the pareto-distributions work. Although the average page size is 16 kB, the page size from a pareto-distribution can...
vary greatly. From T=200 s to T=250 s, one of the connections receives a page of about 280 kB. The axes of the graph have been scaled to allow a common reference frame with the second scenario.

The simulation results of the second scenario is shown in Figure 4-8b. In this scenario the radio link bandwidth is used much more efficiently than in the first scenario. There are no periods when the radio link is inactive, and the average throughput rate is much higher. The total throughput is about four times larger in the second scenario compared to the first scenario. This matches the ratio between the input data of the two scenarios.

The point of these simulations is that by including adaptive RRM it is possible to utilize the available base station bandwidth efficiently and allow more concurrent users. With a fixed flow rate assignment, a lot of bandwidth might be unused. Because of the short data download periods and long think times where capacity is unused in web-browsing, it is possible to mix several such flows with adaptive flow rate handling and avoid overloading the radio link.

Having adaptive RRM is not necessarily a good thing however. In Figure 4-9, results are presented from a simulation where the flow rate update algorithm of the RNC is faster than the send algorithm of TCP. The scenario is the same as in Figure 4-8b, but the RTT of the connections have been increased to 6 seconds while keeping the allowed flow rate update cycle at 2 seconds. When doing a comparison with Figure 4-8b, it can be seen that the throughput have dropped by 30%.

Figure 4-9. Throughput for 8 web-browsing flows with a long RTT compared to the RNC flow rate update cycle.
The reason for this can be found in Figure 4-10, where the received rate and the allowed RNC flow rate is presented. The large RTT for TCP compared to the RNC flow rate update cycle causes an oscillatory behaviour between the TCP rate and the RNC flow rate, and this leads to a lower overall throughput. Here is a short explanation to why the oscillatory behaviour continues:

When the RNC allows a high radio link flow rate, the throughput of the radio link increases. Because of the long delay, it takes a while until the TCP sender gets acks for the high send rate. During this time, the TCP sender uses a lower send rate. When the RNC flow rate algorithm does its next flow rate update, it sees the low incoming data flow rate. As a result, it believes that the lower rate is the actual data flow rate and reduces the radio link rate to match the data flow rate. Then TCP gets the acks with the previous high flow rate, and sends at that rate. At the next allowed flow rate update, the RNC sees the higher incoming data rate and adjust the allowed flow rate. However, the TCP sender now uses the previous lower send rate.

The numbers used in this scenario to highlight the problem are exaggerated compared to normal TCP flows, but they may temporarily arise in connections. To work around this problem, flow rates must be updated in a more intelligent way than just at fixed intervals. To find a working solution can be difficult, since RTTs can vary a lot and different transport protocols have different strategies for determining their send rates.

4.6 Evaluation of full simulations compared to simplifying parts of the network

As noted in Section 4.4, it is not always possible to simplify a part of the network and get reliable results from simulations. In all previous scenarios, the wired network have been...
simulated as a link with a fixed delay and in some of the scenarios the radio link have been replaced with a fixed delay (link sublayer). This section tries to find out what effects may be caused by those simplifications. It will evaluate the scenarios based on their total throughput.

In this section all four of the network scenarios presented in Section 4.2 will be used. The RNC flow rate adaption is not required, and to avoid interference allowed flow rates are fixed at 128 kbps.

To use a fixed average delay instead of the RLC is usually bad for link behaviour. The evaluation in Section 4.4 indicates that it may be possible to use delay distributions with good results. Distribution properties depends on the connection quality and thus relies heavily on the air interface, the RRM algorithms in the RNC and the RLC protocol.

The wired network have been simplified in previous simulations. In a normal situation the wired network consists of several nodes where a lot of traffic are competing for resources and causing temporary congestion. This introduces variable delays in the wired network in the same way as the full radio link, and it can cause bad effects in some transport protocols, most notably standard TCP. To evaluate this impact on connections, a small interfering network is introduced. The only job it has is to interfere with the traffic on the connection over the radio link and cause variable delays.

A few preparation steps is necessary before simulations can start. The average delays in the radio link and in the interfering network have to be identified, so that they can be used to emulate the behaviour of those components in the simplified network scenarios A,B and C. The aim of this is to have a constant RTT to ensure that the TCP send mechanism does not interfere with the results.

A pure RLC link was simulated to get an average RTT baseline, and the average RTT was determined to be about 3.53 seconds (in the range 3.46-3.60 seconds for several different runs).

This average RTT was added as delay in a link sublayer with an interference network. A simulation of this setup gave a total average delay of 4.20 seconds. Assuming that the delays are additive, the average RTT in the interference network was determined to be 0.67 seconds.

Now both the interference network and the radio link could be simulated by using a delay of a normal link. A comparison is made between the four scenarios. The base to which all other scenarios will be compared is scenario D. In all the other scenarios some part of the
network is simulated by a simple delay. The result of the simulations are presented in Figure 4-11.

![Graph](image.png)

**Figure 4-11. Total data throughput for different network simplifications.**

Here follows some comments on the simulation of each scenario:

- **Scenario A**
  In this scenario the link sublayer is used and the radio link is modeled as a simple delay. The wired network is also modeled as a simple delay. In this scenario the total RTT was 4.19 s, which almost exactly matches the target value of 4.20 s.

- **Scenario B**
  In scenario B, the network is modeled as a delay but the full RLC sublayer is used to model the radio link. When setting the network delay to the value found during the preparations, 0.67 s, the total RTT became 3.60 s. In an attempt to increase its total RTT to the target RTT of 4.20 s, the fixed network delay was increased. What happened then was that the TCP sender reduced its sending rate, causing shorter delays in UTRAN queues, and the overall RTT almost stayed the same with only a small increase. It was only possible to increase the RTT to around 3.8 s. After that the TCP sender started to deteriorate and almost stopped sending packets.

- **Scenario C**
  In scenario C the network is modeled through the interference network, and the radio link is modeled as a simple delay by the link sublayer. Since this was the scenario used when determining the network delay, it has the same total RTT of 4.20 s.

- **Scenario D**
  In this scenario, no part of the network is modeled as simple delays. The wired network is simulated by the small interference network and the radio link is simulated using the full RLC sublayer. In this scenario, there are no possibilities to try to increase total RTT in the way that was attempted in scenario B since no delay components are used. The total RTT for the scenario was 2.15 s. The short RTT can at least be partially explained. The highly variable delay causes a small send window
in the TCP algorithm. The send rate is from time to time cut off temporarily, and
this causes short queues in UTRAN. When the TCP sender starts again, the pack-
etts can pass relatively freely through UTRAN (where most of the delay is located
in the other scenarios). The result is that a few packets experience long delays, but
most packets experience a short delay and the average RTT is short.

The simulation tells us that scenario A sends too much data (compared to the full simulation
in scenario D), and this is caused by perfect TCP matching (no variable delays). Scenario
C gives a too pessimistic approximation of the network. The one closest to the full imple-
mentation is scenario B.

Although the total RTT times for scenarios A and C were the same, they were the most sep-
arated scenarios when it came to total throughput. The only difference between them is that
scenario C have a variable delay, and this is not liked by the TCP algorithm.

The assumption that the network and radio link delays are additive were proven wrong. If
they had been additive, all scenarios would have gotten the same RTT. The total delay
depend on the interaction between the different parts of the connection, e.g. the data trans-
port algorithm and the radio link errors.

The most interesting result in these simulations is that the full simulation in scenario D
achieves a higher throughput than when the interfering network is simulated by a delay in
scenario C. This is mostly explained by the differences in RTT and its impact on the TCP
send algorithm. The short RTT of the full simulation causes a quicker rotation of data, and
thereby higher data throughput. This is somewhat counteracted by that the congestion
window of scenario D is smaller, since it have larger RTT variations.

Conclusions that can be drawn from these simulations is that total throughput and through-
put rates are highly dependent on full simulation of all network elements. Replacing com-
plex network elements with average delays can both underestimate and overestimate
simulation results. It is possible to use simplified network elements in simulation, but the
usability of the results are highly dependent on the scenarios. Quantitative measurements
obtained with simplified network components are likely to be different compared to those
obtained using the full network components.

4.7 Evaluation summary

A functional evaluation of the RNC component have been performed to establish the base
properties of the RRM algorithm. It showed how the resource management algorithm
adapted to the incoming flow rates and how flow rates were restricted when the total load
of the incoming flows exceeded the available resources. It also showed that the component
suffers from relatively long delays when used together with transport protocols that adapt
to network conditions by increasing their initial send rate.

A comparison have been made between the two different sublayers that can be used by the
RNC component. It showed the different delay properties of the sublayers and the impact
on data transport with the TCP protocol.

The radio resource management strategy was also evaluated and it was shown how its adap-
tive property can be used to improve data throughput. It was also shown that the adaptive
property can cause undesirable behaviour in terms of oscillatory rates if the RRM algorithm matches data transport algorithms in speed.

Finally an evaluation was made of how simplifications of network components could affect results of simulations. The conclusion was that simplifications can thoroughly wreck simulation results, and care must be taken when identifying what impact the simplifications have in each case.
Evaluation of the adaptive streaming application

The evaluation of the adaptive streaming application have been limited, as problems with the design have been discovered. The problems are described in the simulation presented below.

The network scenario evaluated is an adaptive RNC controlling seven web-browsing flows and one adaptive streamer flow. The link sublayer is used for all flows. This is the same as in Section 4.5, case 2, but with one of the TCP flows exchanged with the adaptive streamer flow. The streamer is set to stepwise rate handling with a rate step of 4 kbps, an initial rate of 64 kbps and probes every 12th second. The RTT delay threshold was 5 seconds and the playback buffer could store 20000 bytes of data.

In Figure 5-1, the send rate for the TCP agent fed by an adaptive streamer data source is presented. At approximately T=60 seconds, the TCP agent receives feedback from the network that causes it to reduce its send rate. Since the adaptive streamer can not detect this, it continues to send at its current streaming rate. This leads to increasing delays for streaming data, since TCP buffers the data it cannot send immediately. After a while, the delay in the TCB buffers in turn causes the adaptive streamer to reduce its data rate because of its RTT limitation rule. The streamer continues to decrease its data rate until TCP empties its
internal buffers and the delay goes below the RTT threshold of the streamer. Then it starts to increase its data rate until TCP panics the next time, which in this case happens at T=150 seconds. There is a small disturbance at 130 seconds, but it does not affect the streamer. As can be seen in the figure, the behaviour of TCP deteriorates as the streaming rate moves closer to the flow rate limit. Between the “jumps” in TCPs send rate, the stepwise increase of the streamer data rate can be seen. The only conclusion that can be drawn from this simulation is that normal TCP is not well suited for streaming data. However, this is a well known property of TCP, and this simulation does not add anything new. To correct this problem, the streaming sender have to be redesigned. This is discussed further in Chapter 7.

The streamer operation have also been evaluated for correctness regarding its control rules. The adaptive streamer adapts to network conditions as required, but is dependent on its settings for good performance. The adaption can be partially seen in Figure 5-1, by comparing the trends before and after the first spike in the TCP send rate. If no adaption had occurred there would have been a single straight line divided by the spike. Now there are two parallel lines which is a result of that the streaming rate have been changed during the spike.

With a careless choice of streamer settings the rate step property of the rate handler can lead to a CBR-like reaction of the RNC compared to TCP flows, especially if the streamer is slow to change its send rate. It can be difficult to produce good settings since a stable streaming rate is desired, yet it should adapt to the network conditions. The easiest way around this is to use the rate table instead, with the streaming rates matching the rates of the connection RAB-table. The settings of the streaming application can also be tuned to provide different behaviour depending on the type of streamed data, e.g. streaming video compared to streaming audio.
6 Conclusions

In the thesis limited RRM functionality have been implemented in a ns-2 node. An algorithm that fairly distributes bandwidth and channel access between flows have been implemented and verified. The effects of using the adaptive ability of the bandwidth handler have been studied and the conclusion was that it made efficient use of available bandwidth for typical best-effort traffic.

The feasibility of replacing network parts with simple delays was studied, and the result was that its usability depends on the network scenario and what the simulation results are used for. In particular, it was shown that quantitative measurements obtained with simplified network elements have little or no use in determining the behaviour of the full network elements.

An adaptive streaming application have been implemented, but problems in the design made it useless for evaluation of RRM functionality. The control rules of the adaptive streamer worked as they were supposed to, but the controlling algorithm and the transport protocol it used wasn’t fully compatible and caused undesired functionality.
7 Future work

7.1 RNC

The implemented module takes a very high-leveled approach to the RNC and the RAB, and is not concerned with the power issues in a UMTS network. This means that the current implementation does not provide an accurate picture of how bit rates are changing in a real UMTS network with user movement within a cell. A future development could be aimed at expanding the physical model and interface in ns-2 to allow the module to manage the base station power instead of the flow rates. Currently the layer that receives packets in ns-2 only decides whether or not to deliver the packets based on the received signal strength. The interface to this layer have to be expanded to give the RNC module access to power and signal strength information.

Another issue which is not addressed in the current implementation is how to handle traffic in the different QoS classes in UMTS. To do this, connections have to be associated with a QoS class, and this class have to be communicated in some way to the RNC. One solution is to add a routing agent to the RNC routing module and create a new routing module with a routing agent for the mobile terminals. This allows true connection setup between the RNC and the mobile terminals and actual RAB requests with varying parameters can be exchanged. During development it have been discovered that the added complexity of manual component setup would be much lower than what was suspected at the beginning of the RNC module implementation. The suspected complexity issue was the main reason to why the setup was not included from the beginning. Unfortunately there have been no time to implement it.

If both power management and call setup are implemented, a closer mapping to a real-world system is achieved while maintaining a relatively simple setup procedure in ns-2 with only one or two added components for inclusion of the UMTS part. This results in a system that is more closely mapped to a real UMTS system and thus provides a better base structure for studies of network behaviour in higher layers.

For further development of the RNC module, these expansions allows the radio resource management in the RNC to include all QoS classes and allows development of improved rate handling algorithms in the RNC module. The new algorithms can take delay in UTRAN into account for the different traffic classes and provide the correct service level for each class.

To allow simulation of user-mobility effects in the network, an ongoing connection must have the ability to move to another base station. Currently there is no such support, but it should not be too complicated to extend the implementation with support for multiple base stations in a single RNC. This scenario can also be extended with the ability to move the connection between different controlling RNCs.
The main bottleneck for further development of current implementation is the connection setup. Most of the other extensions of the RNC module are dependent on the implementation of this part. If it is implemented, most of the true-world functionality of the RNC can be added with small or no overall changes to the rest of the module.

On a coding level, it would be better to separate the setup-specific methods and move them to OTcl. Currently, most things are done in C++ and only those things that can not be done in C++ are in OTcl.

7.2 Adaptive streamer

In the base ns-2 protocol structure, there is support for giving application data packets to transport protocols. The problem is that some transport protocols does not implement the transport of the data contained in the packets. There is no standard way implemented to support application data transport and different application-based techniques must be used for different protocols. The current adaptive streamer implementation is restricted to a single transport protocol and needs an adapting layer that provides an independent way of sending application data regardless of underlying data transport protocol. To simplify the setup of network scenarios, the layer should preferably include functionality to detect which transport protocol is used and adapt appropriately.

The current sending algorithm is very simple and can be improved a lot. It assumes that the data source is capable of producing a bit stream of arbitrary bit rate, which may not be true in a real application. This means that the sending algorithm is probably a main drawback of the adaptive streamer in terms of realism and comparability with real-world systems.
Bibliography

[1] 3GPP, 3G Partnership Program.


[4] 3GPP TS 23.107 Quality of Service (QoS) concept and architecture, V5.7.0 (2002-12).


'Adaptive Load Control Algorithms for 3rd Generation Mobile Networks', in
Proceedings of the 5th ACM International Workshop on Modeling, Analysis and
Simulation of Wireless and Mobile Systems, pp. 104--111, ACM CS-press,
(2002-09).

'Performance Evaluation of Channel Switching Scheme for Packet Data
Transmission in Radio Network Controller’, in Lecture notes in Computer Science,


A User guide

The component have been designed for ns-2.1b9a. If you use a different ns-2 version, it is possible that a few changes are needed to make things work. The changes to the core ns-2 files mentioned in this appendix may not be required by other versions.

A.1 Using the RNC component

The RNC module can work both with a RLC/MAC sublayer and a sublayer consisting of normal ns-2 links. When using links, the rate of the links are updated by the RNC. Such link rate updates will not be effective until the next packet sent over the link. Rate updates only affect the rate of the down-link, the uplink rate are the same as the link is initialized with. Installation and usage of the RLC/MAC layer can be found in [11].

A.1.1 Installation and usage example

The files used by the RNC component is (rnc.(h|cc|tcl), rncflow.(h|cc), flowclassifier.(h|cc), sublayer.(h|cc)) and they should be stored like this(this is preparatory work for changes in the Makefile):

*.cc and *.h in $RNC_CC_LIB (directory of your choice)

*.tcl in $RNC_TCL_LIB (directory of your choice) under your main ns directory.

Make the required changes to the Makefile:

For all files 'file.cc’ add $RNC_CC_LIB/file.o to the OBJ_CC macro.
Add $RNC_TCL_LIB/file.tcl to the NS_TCL_LIB macro for all files 'file.tcl’.
Add -I./$RNC_CC_LIB to the INCLUDES macro.

Some of the files in the ns-2 distribution have to be changed:

ns-default.tcl: Add these lines after RtModule init lines

RtModule/Rnc set rate_change_interval_ 2s
RtModule/Rnc set rate_change_delay_ 1s
RtModule/Rnc set total_rate_ 2Mb
RtModule/Rnc set flow_queue_type_ DropTail
RtModule/Rnc set flow_queue_length_ 10
RtModule/Rnc set flowtype_ Dest

ns-lib.tcl: Add this line somewhere after the line "source ns-default.tcl" (replace RNC_TCL_LIB with the directory where you have put your copy of the rnc.tcl file)
source "RNC_TCL_LIB/rnc.tcl"

classifier-hash.h: Change the lookup() function

55
virtual int lookup(Packet* p) {
  
  Change last line
  return get_hash(mshift(h->saddr()), mshift(h->daddr()),
  h->flowid());
  to
  return get_hash(h->saddr(), h->daddr(), h->flowid());
}

classifier-hash.cc:
DestHashClassifier::classify(Packet* p) {
  
  Change last line
  return -1;
  to
  return (unknown(p));
}

This change makes the DestHashClassifier identical to all the other hash classifiers. The change is required because the RNC module uses the "unknown" call and its TCL callback to detect new flows.

Usage example
The first thing that should be done before a RNC module is created, is to tell the RNC how to separate packets into flows based on packet address. This is done by setting the class variable flowType_. It defaults to "Dest", which means that the RNC associates packets with different destination addresses to different flows. If the default setting is OK, this step may be left out. The flowtype must be set before the RNC module is created:

RtModule/Rnc set flowtype_ SrcDest

To create a node with the RNC module installed, the module have to be registered in the Node class which is done by a call to the Node class-procedure 'enable-module'. The RNC module must be enable after any calls to "ns node-config" that changes the addressing routing modules. This is a hard requirement, as the addressing routing modules replaces any classifiers they find at the node entry-point with their own. The addressing routing modules must therefore be added to the node module list before the RNC component is added. To get a handle on the RNC module installed in a node, call the node procedure 'get-module':

Node enable-module Rnc
set n1 [$ns node]
Node disable-module Rnc ;# Unregister RNC, can create normal # nodes again

set rncmodule [$n1 get-module Rnc]

The RNC may be used in two sublayer modes, which mode to use depends on the underlying layers for packet transport. When used together with the RLC layer, no further setup is
required. The RNC automatically detects the RLC layer and adjust its own sublayer to the RLC layer. When used with links, the RNC must be told which links to control and it must be given a RAB list to select flow rates from. To let the RNC control a link, just give the link as argument to the RNC procedure 'control-link'. The RNC will then control flows that pass through the link. Which flowrates to use is controlled by calling 'set-RAB' with a list of enabled flow rates:

```bash
$rncmodule control-link [$ns link $rncnode $n1]
$rncmodule control-link [$ns link $rncnode $n2]
$rncmodule set-RAB [list 4000 12000 24000 28800 32000 48000 56000 64000]
```

Remember that the order of the arguments to '$ns link' is important, as ns-2 uses directional links to connect nodes. In this scenario, flows going to node n3 are not controlled by the RNC, even though they pass through the RNC module.

The RAB-list must contain at least two elements, and the RNC assumes that the rates are given in increasing order. The initial rate of RNC flows is the second element in the list. It is possible to set a single rate for all flows through the RNC by giving it a RAB list with two elements with the same rate.

When using the RLC sublayer, the RAB-list are normally automatically acquired from the RLC by the RNC. It is possible to change the RAB-list used by the RLC when creating a network scenario. Add the following lines to the node setup function:

```bash
proc setupnode {node} {
...
    l1_(0) clear-rab
    l1_(0) add-rab-speed <rate 1>
    l1_(0) add-rab-speed <rate 2>
}
```

Be careful when setting the smallest rate in a RAB-list; make it too large, and the RNC will quickly run out of available rate when the number of flows through the RNC increases.

Figure A-1. Example of a link sublayer configuration.

Let RNC control links to n1 and n2 in Figure A-1 and set a RAB-list:

```bash
$rncmodule control-link [$ns link $rncnode $n1]
$rncmodule control-link [$ns link $rncnode $n2]
$rncmodule set-RAB [list 4000 12000 24000 28800 32000 48000 56000 64000]
```

Remember that the order of the arguments to '$ns link' is important, as ns-2 uses directional links to connect nodes. In this scenario, flows going to node n3 are not controlled by the RNC, even though they pass through the RNC module.

The RAB-list must contain at least two elements, and the RNC assumes that the rates are given in increasing order. The initial rate of RNC flows is the second element in the list. It is possible to set a single rate for all flows through the RNC by giving it a RAB list with two elements with the same rate.

When using the RLC sublayer, the RAB-list are normally automatically acquired from the RLC by the RNC. It is possible to change the RAB-list used by the RLC when creating a network scenario. Add the following lines to the node setup function:

```bash
proc setupnode {node} {
...
    l1_(0) clear-rab
    l1_(0) add-rab-speed <rate 1>
    l1_(0) add-rab-speed <rate 2>
}
```

Be careful when setting the smallest rate in a RAB-list; make it too large, and the RNC will quickly run out of available rate when the number of flows through the RNC increases.

57
Make it to small, and sleeping flows will have a hard time to start sending again. Rate updates only apply to next sent packet, any packet in transfer will not be affected. If the rate is very small, it may take a long time before the rate update is seen on the link.

**RNC tracing**

It is possible to get traces of the RNC activities. Rate updates of flows and packet movement on flows may be traced by calling the following procedures in the RNC module:

```
$rncmodule\{\text{trace-rates}|\text{trace-packets}|\text{trace-all}\} \$trace\_file
```

Before closing the trace-file, make sure all data is written to it:

```
$rncmodule\ \text{flush-trace}
```

The trace-file argument should be a handle to a standard ns-2 trace file. The trace procedures override each other, so only one trace file is used for traces at any time. The different fields of the trace file generated by the RNC are described in Table A-1.

### Table A-1. RNC trace file field descriptions.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1     | Type of packet event. ‘+’, ‘-’, ‘s’ or ‘c’.  
|       | ‘+’: Packet arrival on RNC flow  
|       | ‘-’: Packet departure from RNC flow  
|       | ‘s’: Scheduled rate change  
|       | ‘c’: Completed rate change |
| 2     | Time of event |
| 3     | Flow source identifier |
| 4     | Flow destination identifier |
| 5     | Flow identifier(IPv6) |
| 6     | Flow rate. 0 if event is ‘+’ or ‘-’, new rate of flow if event is ‘s’ or ‘c’. |

### A.1.2 Settings

The variables that are most likely to be changed in different network scenarios are presented in Table A-2.

### Table A-2. RNC variable settings.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type, default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate_change_interval</td>
<td>Time, 2 seconds.</td>
<td>How often RNC performs flow rate update calculations.</td>
</tr>
<tr>
<td>rate_change_delay</td>
<td>Time, 1 second</td>
<td>How much time should pass before a rate update is implemented. Simulates time for control information exchange between RNC and mobile terminal about rate change.</td>
</tr>
</tbody>
</table>
The procedures that are available to change the RNC behaviour are presented in Table A-3.

**Table A-2. RNC variable settings.**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type, default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>total_rate_</td>
<td>Bandwidth, 2Mbps</td>
<td>Total amount of available bandwidth in RNC. Since we only let RNC control one base station, this should equal the rate a single base station can transmit.</td>
</tr>
<tr>
<td>flowtype_</td>
<td>String, ‘Dest’</td>
<td>Determines how to distinguish flows in the RNC based on packet address. Supported values are SrcDestFid, SrcDest, Dest and Fid. Only Dest are currently(2003-06) supported by the RLC layer. This variable must be set before the RNC is created. If it is changed after the RNC have been created, the RNC behaviour is undefined.</td>
</tr>
</tbody>
</table>

The procedures that are available to change the RNC behaviour are presented in Table A-3.

**Table A-3. RNC TCL methods.**

<table>
<thead>
<tr>
<th>Method signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>drop-target [target]</td>
<td>Sets or retrieves the drop-target used by RNC. This is the destination of any dropped packets in case of RNC overflow. If parameter target is specified, the RNC drop target is set to the specified target. Otherwise, the current RNC drop-target is returned. The target should be of type Connector.</td>
</tr>
<tr>
<td>departure-interval time</td>
<td>Specifies how often the RNC should check for remaining stored packets to send. The unit of the parameter is seconds, i.e if you want the RNC to check for unsent packets 10 times per second, you should specify time as 0.1.</td>
</tr>
</tbody>
</table>

RNC trace methods

<table>
<thead>
<tr>
<th>Method signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>trace-rates tracefile</td>
<td>Starts tracing of RNC flow rate changes. The traces are written to the specified tracefile.</td>
</tr>
<tr>
<td>trace-packets tracefile</td>
<td>Starts tracing of packet arrivals and departures in the RNC.</td>
</tr>
<tr>
<td>trace-all tracefile</td>
<td>Enables full tracing of RNC, both rate changes and packet events are traced.</td>
</tr>
<tr>
<td>flush-trace</td>
<td>Atomically writes all generated traces so far to the trace file. by flushing all internal buffers.</td>
</tr>
</tbody>
</table>

RNC link sublayer mode only

<table>
<thead>
<tr>
<th>Method signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>control-link link</td>
<td>Lets RNC take control over the traffic sent over the link. The link parameter should be a directional link between two nodes on which you want to control the rate. Note: RNC can only control the rate over the link correctly, if the traffic through the link also passes the RNC.</td>
</tr>
<tr>
<td>set-RAB ratelist</td>
<td>Sets the rates of the RAB list that the RNC uses to set flow rates. The ratelist parameter should be a plain TCL list with bandwidth elements. The list elements should be specified in increasing bandwidth order.</td>
</tr>
</tbody>
</table>

There are some other variables and procedures that can be used to tune the behaviour of the RNC, but they require intimate knowledge of the controlling algorithms (i.e. the algorithm
code have to be studied). The algorithm that controls rate updates are in the main RNC class in the procedure `updateFlowRates()`. The packet scheduling algorithm are also in the main RNC class, in the procedure `packetDeparture()`.

### A.2 Using the adaptive streamer

The streaming application consists of two different applications, one sender and one receiver. Currently it only works on top of the FullTCP protocol, but it should be possible with some changes to run it on top of any protocol that supports insertion of data in packages (untested).

The algorithm used for automatic adaption of streaming rate is very simple, and reacts slowly to network condition changes. In principle it probes the network with higher sending rate until some packet is lost or have too long delay (maxRtt overrun). Then it decreases one step in rate. The probes occurs at specified time intervals.

Rate changes can be handled in different ways. By setting the `useRateHandler_` variable to false, it is possible to do it explicitly from a TCL script by using the `setRate` call on the streamer object. If you are using the rate handler, it can operate in two modes: Rate step and rate table. In the first mode, you specify a rate increment and a start rate (you may also set Min and Max rates). When a rate change is required, the rate handler changes the send rate by the specified rate step. In the second mode the rates are selected from a specified rate table. Which mode it uses depends on which configuration calls are used.

### A.2.1 Installation and usage example

The streamer component consists of the files (streamer.(h|cc|tcl), streamersink.(h|cc|tcl), streamerdata.(h|cc), ratehandler.(h|cc)) and should be stored as follows (preparatory work for Makefile changes):

- `.cc` and `*.h` in `$STREAMER_CC_LIB`(directory of your choice)
- `*.tcl` in `$STREAMER_TCL_LIB`(directory of your choice) under your main `ns` directory.

Make the required changes to the Makefile:

- For all files `file.cc` add `$STREAMER_CC_LIB/file.o` to the `OBJ_CC` macro.
- Add `$STREAMER_TCL_LIB/file.tcl` to the `NS_TCL_LIB` macro for all files `file.tcl`.
- Add `-I./$STREAMER_CC_LIB` to the `INCLUDES` macro.

Some of the files in the ns-2 distribution have to be changed:

* `ns-process.h`: Add the following lines before last ADU in `AppDataType` enumerator at the beginning of the file

```c
// Adaptive streamer ADUs
STREAMER_DATA,
STREAMER_ACK,
STREAMER_START,
STREAMER_STOP,
```
**ns-default.tcl:** Add the following lines to the Application settings subset

```
Application/Streamer set packetSize_ 500
Application/Streamer set bufferSize_ 2000
Application/Streamer set rate_ 1000
Application/Streamer set useRateHandler_ true
Application/Streamer set probeInterval_ 10
Application/Streamer set maxRtt_ 5
```

**Usage example**

In this example, $tcp0 and $tcp1 are FullTcp agents connected to nodes in a network. $ns is the main ns-2 application.

First, create the network topology and the FullTCP agents. Connect the agents to nodes in the network. How this should be done is described in the ns-2 documentation.

Create the sending and receiving streaming applications and connect them to the agents:

```
set streamer [new Application/Streamer $tcp0]
set streamersink [new Application/StreamerSink $tcp1]
```

Connect the two TCP agents and let the receiving TCP agent listen for new connections:

```
$ns connect $tcp0 $tcp1
$tcp1 listen
```

Connect the streaming sender and receiver:

```
$streamer connect $streamersink
```

Finally, schedule start and stop of the streaming application

```
$ns at 10.0 "$streamer start"
$ns at 60.0 "$streamer stop"
```

**A.2.2 Settings**

The variables that can be changed to tune the adaptive streamer behaviour are presented in Table A-4.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type, default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>packetSize_</td>
<td>500 bytes</td>
<td>Size of data packets.</td>
</tr>
<tr>
<td>bufferSize_</td>
<td>2000 bytes</td>
<td>Size of data buffer at receiver.</td>
</tr>
<tr>
<td>rate_</td>
<td>Rate, 1000 bps</td>
<td>Current streaming data rate.</td>
</tr>
<tr>
<td>useRateHandler_</td>
<td>Boolean, true</td>
<td>Enables or disables the rate handler module.</td>
</tr>
<tr>
<td>probeInterval_</td>
<td>Time, 10 seconds</td>
<td>Time between two attempts to increase streaming data rate.</td>
</tr>
<tr>
<td>maxRtt_</td>
<td>Time, 5 seconds</td>
<td>Maximum time to wait for acknowledgment of data packets. Used to detect congestion in the network.</td>
</tr>
</tbody>
</table>
Some configuration calls exist for the streamer and they are presented in Table A-5.

Table A-5. Streamer TCL methods.

<table>
<thead>
<tr>
<th>Method signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sendrate</td>
<td>Returns current send rate of streaming data.</td>
</tr>
<tr>
<td>sendcount</td>
<td>Returns number of sent data packets.</td>
</tr>
<tr>
<td>missedcount</td>
<td>Returns number of packets that have missed their playback deadline. Even if some but not all data in the packet could be played back on time, it is counted.</td>
</tr>
<tr>
<td>setPacketSize size</td>
<td>Sets amount of data (in bytes) transferred in each packet.</td>
</tr>
<tr>
<td>setBufferSize size</td>
<td>Sets playback buffer size (in bytes) at the receiver.</td>
</tr>
<tr>
<td>setRate rate</td>
<td>Sets current sending rate in bits per second (bps).</td>
</tr>
<tr>
<td>setRateStep size</td>
<td>Sets rate increment in bps for sender and enables rate step function of rate handler.</td>
</tr>
<tr>
<td>setMinRate rate</td>
<td>Sets the minimum transfer rate in bps when using the automatic rate handler.</td>
</tr>
<tr>
<td>setMaxRate rate</td>
<td>Sets the maximum transfer rate in bps when using the automatic rate handler.</td>
</tr>
<tr>
<td>setProbeInterval interval</td>
<td>Sets the time interval between two attempts to increase the sending rate.</td>
</tr>
<tr>
<td>setMaxRtt time</td>
<td>Sets maximum time to wait for an ACK of a sent packet. It is used to detect congestion.</td>
</tr>
<tr>
<td>setRateTable ratelist</td>
<td>Sets a rate table from which the automatic rate handler can select send rates and enables rate table function of rate handler. The rate table should be given as a TCL list of send rates in bps.</td>
</tr>
</tbody>
</table>
Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© Björn Knutsson