Interfacing Node B and RNC by tunneling ATM over IP for test and verification purposes

Ralf Nyrén

ralf@nyren.net

30th November 2007
Master's Thesis in Computing Science, 30 ECTS credits
Supervisor at CS-UmU: Jerry Eriksson
Examiner: Per Lindström

Umeå University
Department of Computing Science
SE-901 87 UMEÅ
SWEDEN

Abstract

A system for tunneling ATM traffic over an IP network has been developed using "commodity" hardware with optical carrier 3, OC3/STM-1 (155Mbit/s), ATM interface and a Free Software operating system. The developed ATMtunnel software has been verified in latency sensitive UMTS Terrestrial Radio Access Network (UTRAN) environment as the interface (Iub) between Radio Network Controller (RNC) and Node B base station. The thesis proves the concept of connecting physically remote ATM based test equipment using existing IP networks for test and verification purposes.

Contents

Abbreviations

3GPP 3rd Generation Partnership Project.

AAL ATM Adaptation Layer.

AAL0 ATM Adaptation Layer 0 (raw ATM cell).

AAL5 ATM Adaptation Layer 5.

ATM Asynchronous Transfer Mode.

BFN Node B Frame Number (counter).

CFN Connection Frame Number.

DCH Dedicated Channel.

DiffServ Differentiated Services.

DL Down Link.

DS1 Digital Signal Level 1 (T1).

DS3 Digital Signal Level 3 (T3).

DSCP Differentiated Services Code Point.

DSL Digital Subscriber Line.

EDGE Enhanced Data rates for GSM Evolution.

E1 2.048 Mbit/s rate European carrier standard to transmit 30 digital channels (voice or data).

E3 34.368 Mbit/s rate European carrier standard to transmit 16 E1.

ETM Exchange Terminal Module (transmission board).

FACH Forward Access Channel.

FDD Frequency Division Duplex.

FEC Forwarding Equivalence Class.

GPRS General Packet Radio System.

GSM Global System for Mobile communications.

HLR Home Location Register. UE register database.

HSDPA High Speed Downlink Packet Access.

ICMP Internet Control Message Protocol.

IETF Internet Engineering Task Force.

IMA Inverse Multiplexing for ATM.

INET InterNET ATM tunnel receiver and sender module. See chapter 2.

IP Internet Protocol.

ITU International Telecommunication Union.

ITU-T ITU Telecommunication Standardisation Sector.

Iu Interface between an RNC and Core Network components. Provides an interconnection point between the RNS and the Core Network.

Iub Interface between an RNC and a Node B.

Iur Logical interface between two RNCs. Whilst logically representing a point to point link between RNCs, the physical realisation need not be a point to point link.

LAN Local Area Network.

LDP Label Distribution Protocol.

LSP Label Switched Path.

LSR Label Switching Router.

LTOA Latest Time Of Arrival.

MGW Media GateWay. Translation unit between different networks.

MPLS Multi Protocol Label Switching.

MSC Mobile Switching Center.

MTU Maximum Transmission Unit.

`NetEm` Linux Network Emulation service [13].

NTP Network Time Protocol.

OC3 Optical Carrier specification 3. A network line with transmission speeds of up to 155.52 Mbit/s. Also known as STM-1 from the SDH ITU-T standard.

PHB Per-Hop Behaviour.

PSTN Public Switched Telephone Network.

PTI Protocol Type Indicator in an ATM cell.

PVC Permanent Virtual Connection.

QoS Quality of Service.

RAN Radio Access Network.

RBS Radio Base Station. Also referred to as “Node B” in 3GPP.

RFN RNC Frame Number (counter).

RNC Radio Network Controller.

RNS Radio Network Subsystem.

RSVP Resource ReSerVation Protocol.

RSVP-TE RSVP with Traffic-engineering Extension.

RTT Round Trip Delay Time.

SDH Synchronous Digital Hierarchy.

SFN System Frame Number.

SLA Service Level Agreement.

SRNC Serving Radio Network Controller.

STM-1 Synchronous Transport Module. The basic rate of transmission (155.52 Mbit/s) of the SDH ITU-T fiber optic network transmission standard.

T1 1.544 Mbit/s rate North American Digital Hierarchy Signaling standard (DS1) to transmit 24 digital traffic channels (voice or data).

T3 44.736 Mbit/s rate North American Digital Hierarchy Signaling standard (DS3).

TCA Traffic Conditioning Agreement.

TDD Time Division Duplex.

TDM Time Division Multiplexing.

TOA Time Of Arrival.

TOAWE TOA Window Endpoint.

TOAWS TOA Window Startpoint.

TOS Type of Service.

TTI Time Transmission Interval.

UDP User Datagram Protocol.

UE User Equipment.

UMTS Universal Mobile Telecommunications System.

UTRAN UMTS Terrestrial Radio Access Network.

VCI Virtual Channel Identifier.

VPI Virtual Path Identifier.

VPN Virtual Private Network.

WAN Wide Area Network.

WiMAX Worldwide Interoperability for Microwave Access.

WCDMA Wideband Code Division Multiple Access. The primary air interface standard used by UMTS.

Chapter 1

Introduction

Recent years have shown an increasing demand of packet data services in mobile devices. New technologies for higher bandwidth between base station and user equipment are continuously developed to be able to transfer more information in the ether. The old second generation GSM[1] network received packet data transfer with the introduction of GPRS/EDGE[2] and the third generation UMTS[3] networks provides packet data traffic out of the box. With the introduction of HSDPA[4] the available bandwidth has taken yet another leap and todays users can achieve theoretical upload/download speeds up to 4.2/13.1Mbit/s [24]. At these speeds connecting to the Internet using mobile devices, where Wi-Fi is unavailable, is likely to increase in popularity.

Before the introduction of packet data services in telephone system networks the bandwidth demands were rather modest. Today the popular DSL technologies have increased the bandwidth demands and telephone operators have upgraded their core networks with high capacity network solutions.

The main focus of packet data services to mobile devices appears to be how much data that can be squeezed through the ether. However, equally important is reliable low latency transmission with sufficient capacity between base station and core network. Although based on well known technology, transmission between radio network controller and base station[5] may be expensive to upgrade. A base station is often located where low latency high speed transmission lines are expensive to install. Large mobile phone operators have a large amount of base stations and while the hardware upgrade costs will be huge the required infrastructural changes to deliver HSDPA speeds from core network to user equipment will be a significant cost in its own.

Several solutions for increased bandwidth within the Radio Access Network (RAN) have been presented. For example, many telecom operators still use E1 carriers [18] and a technique called IMA[6] can bundle several E1 carriers to

[1] GSM is a mobile phone system widely deployed in Europe and used all over the world.

[2] GPRS and EDGE are two technologies for providing "Internet access" to mobile devices such as mobile phones and computers.

[3] UMTS is the technical term for what is commonly known as the "3G mobile phone system".

[4] High Speed Downlink Packet Access, a technology to provide increased downlink capacity to 3G mobile devices.

[5] Think of the radio network controller as your computer and the base station as a radio modem, albeit both with rather hefty price tags.

[6] Inverse Multiplexing for ATM (Asynchronous Transfer Mode) [7].

provide a higher capacity solution using existing cables. TDM over IP (TDMoIP) is another technique which encapsulates Time Division Multiplexing bitstreams (T1, E1, T3, E3) as pseudowires over packet-switching networks [26]. Radio links are also common for base stations located in places where physical cables are difficult to install.

TietoEnator, Telecom and Media, research and development have contributed to the development of the Ericsson Node B base stations used in UMTS[7] Terrestrial Radio Access Network (UTRAN) systems. Extensive verification procedures are required in order to maintain telecom grade quality of hardware and software. In complex trouble shooting and verification situations involving Node B base stations and user equipment[8] access to a Radio Net Controller (RNC) is a key resource. An RNC is an essential resource in a Radio Access Network (RAN) and must interoperate with other RAN components such as Home Location Register (HLR), Mobile Switching Center (MSC), Media Gateway (MGW), etc to provide a working Universal Mobile Telecommunications System (UMTS) network.

The new technologies of high capacity links for RBS communication have increased the demand of verification and trouble shooting on the transmission level. Tools for traffic analysis and simulation of different link characteristics such as latency, delay jitter and packet loss are therefore of great value.

1.1 Problem description

The purpose of this master thesis project is to investigate the possibilities of connecting an RBS ATM [20] communication interface with an RNC using a pre-existing UDP/IP [25] Local/Wide Area Network (LAN/WAN).

Asynchronous Transfer Mode (ATM) is a network and data link layer protocol commonly deployed in telecommunication systems due to its low latency characteristics and Quality of Service (QoS) capabilities [20]. The Internet Protocol (IP) is generally a very cost effective network solution compared to ATM and although classical IP networks are incapable of providing guaranteed QoS new technologies exist to introduce QoS in IP networks, see chapter 3.

The idea behind the thesis project is to intercept the ATM traffic sent between RNC and RBS and forward it over a UDP/IP network such as a LAN, an intranet or even the Internet. Although this technique might even be used in a real UMTS network its primary value is for test and verification purposes.

This master thesis project will try to answer the following questions:

- Would it be possible to connect an RNC with an RBS at a different physical location using ATM tunneling software over an existing UDP/IP WAN?
- Could ATM tunneling software be used for traffic analysis and simulation of network characteristics for test purposes?

The thesis project will utilise OC3/STM-1 155Mbit/s fiber optic [19] ATM transmission interfaces together with a Free Software [11] platform.

[7] Universal Mobile Telecommunications System.
[8] E.g. Wideband Code Division Multiple Access (WCDMA) mobile phones.

1.2 UTRAN

UMTS Terrestrial Radio Access Network (UTRAN) is a collective term for the components that builds a Universal Mobile Telecommunications System (UMTS) radio access network [14]. Figure 1.1 illustrates the UTRAN components Radio Network Subsystem (RNS), Radio Network Controller (RNC) and Node B base station (RBS). The Core Network consists of nodes that provide

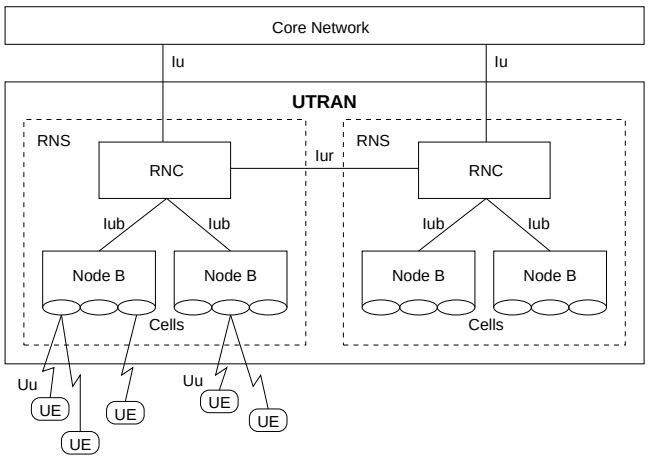

Figure 1.1. UTRAN Architecture [4]. The Core Network consists of nodes that provide services and fundamental network features. UMTS Terrestrial Radio Access Network (UTRAN) is the radio access network of the Universal Mobile Telecommunications System (UMTS). A Radio Network Subsystem (RNS) contains a Radio Network Controller (RNC) connected to one or more Node B base stations. A Node B can manage several network cells where User Equipment (UE), e.g. mobile phones, interact with UTRAN. The Iu, Iur, Iub and Uu are interfaces connecting UMTS components.

services and fundamental network features including:

- Management of user location information.
- Control of network features and services.
- Switching and transmission mechanisms for traffic and signalling.

It is the Core Network that routes traffic between a UTRAN and other telephone networks such as the Public Switched Telephone Network (PSTN) and other mobile phone networks.

UTRAN deploys several interfaces to communication with internal and external components. Internal components include RNS, RNC and RBS while the Core Network and User Equipment (UE) are external components. The UTRAN communication interfaces can be described as follows [4]:

Iu Interface between RNC and core network components providing an interconnection between the RNS and the Core Network.

Iub Interface between RNC and Node B.

Iur Logical interface between two RNCs for use of services such as handover. Figure 1.1 illustrates the Iur as a point-to-point link but the Iur can also be routed through the Core Network.

Uu Radio interface between UTRAN and User Equipment (UE) where Wideband Code Division Multiple Access (WCDMA) [14] is deployed.

Ordinary UMTS uses Frequency Division Duplexing (FDD) together with WCDMA radio interface technology. In FDD the uplink and downlink transmit on different frequencies. There is another technique called Time Division Duplexing (TDD) which allows the uplink and downlink to use the same frequency spectrum. Mobile phone operators use regular UMTS, i.e. FDD, for phone services while TDD is targeted at Internet services competing with other standards such as e.g. WiMAX [14].

UMTS-FDD is not directly compatible with UMTS-TDD although they share many specifications. This report only describes the technology relevant to FDD.

1.2.1 Node Synchronisation

Precision timing is vital in a UMTS Terrestrial Radio Access Network (UTRAN) [4] and both RNC and RBS require a clock source with a precision of at least 0.05ppm[9] for wide area base station deployment [2]. Although UTRAN nodes have high quality internal clocks an external reference is required to maintain the necessary precision. A common technique is to use the network synchronisation [17] features included in the physical layer of the Iub to provide a clock synchronisation reference. However, although the clock reference can be shared between different UTRAN nodes this is not a requirement. For example, an RNC and an RBS within the same RNS are allowed to use independent clock references as long as they provide the required precision. Since different clock references are allowed the frame counters of UTRAN nodes are not required to be phase aligned [3].

Data communication between RNC and UE through an RBS uses frame transport. A frame corresponds to a certain amount of time, usually 10ms. In the downlink scenario it is the RNC which decides when a frame should leave the radio interface in the RBS and thus be transmitted to the UE, i.e. the frame receiver needs a reference to determine when to forward the frame. This is addressed by the use of frame counters in RNC and RBS. The RNC Frame Number (RFN) is used by the RNC and the Node B Frame Number (BFN) is used by the RBS. Think of a frame counter as a clock which *ticks* one step every 10ms, e.g. in a time interval of 30ms it is possible to send three frames.

Node Synchronisation is used to handle the phase difference between the RFN and BFN frame counters. The Node Synchronisation mechanism determines the offset between the RFN and BFN counters as well as measuring the latency of the interface between RNC and RBS (Iub), see figure 1.2. Notice the values of for example T1 and T2 in figure 1.2. The RFN has incremented to 4094 just before the downlink Node Synchronisation is initiated and the value of T1 is **4094**1.250 ms. The node synchronisation control frame is sent by the RNC and received by the Node B (RBS) just after the BFN changed to 149 and likewise T2 has a value of **149**2.500 ms, i.e. a frame corresponds to 10ms.

[9] Parts per million.

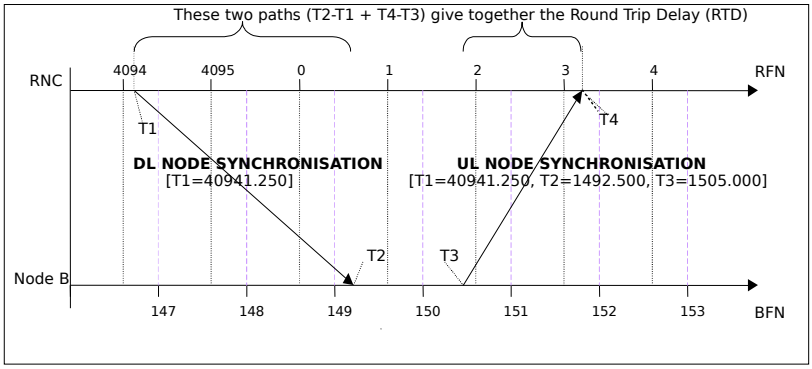

Figure 1.2. RNC–Node B Node Synchronisation [3]

1.2.2 Transport Channel Synchronisation

Transport Channel Synchronisation is the concept of synchronising frame transport between RNC and RBS. This section describes the transport channel synchronisation mechanism valid for all downlink transport channels, such as the Forward Access Channel (FACH), Dedicated Channels (DCHs), etc.

In addition to the RFN and BFN the transport channel synchronisation uses two additional frame counters, the Cell System Frame Number (SFN) and the Connection Frame Number (CFN). The SFN is equal to BFN adjusted with T_cell. The T_cell offset is used within the same Node B to skew different cells[10] in order to avoid colliding Synchronisation Channel bursts. The CFN is the frame counter used for transport channel synchronisation between UE and UTRAN and relates to the SFN by means of the Frame Offset. Figure 1.3

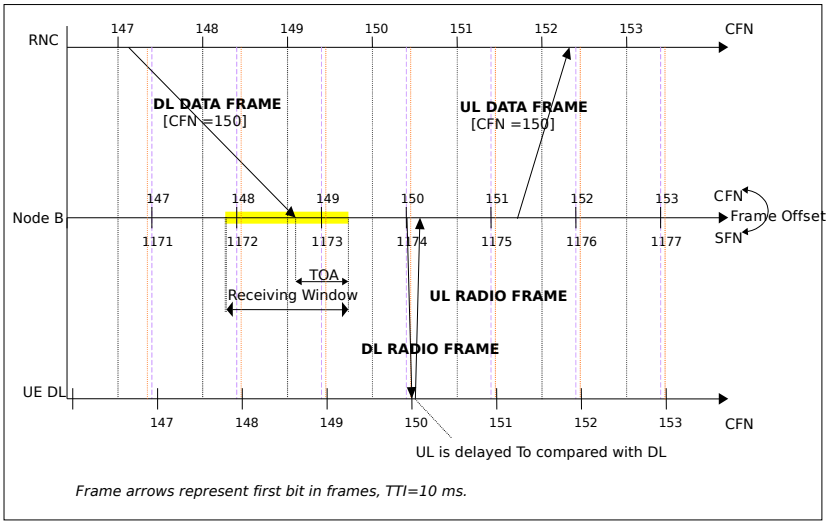

Figure 1.3. Transport Channel Synchronisation [3]

[10] A Node B (RBS) can be responsible for several network cells.

illustrates the scenario of a downlink data frame sent from RNC through RBS to UE with subsequent response [3].

Sending data over a network interface always results in some sort of delay before the information is received at the other end. In order to handle latency variations on the Iub the RBS has a Receiving Window which allows for buffering of received frames scheduled for transmission on the radio interface. The RBS will respond to the RNC with a timing adjustment control frame for each data frame received out of the configured Receiving Window. Figure 1.4 illustrates

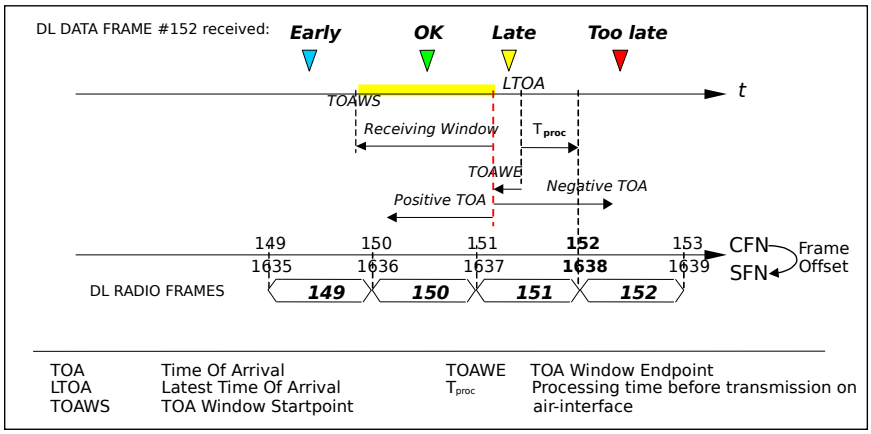

Figure 1.4. Illustration of TOAWS, TOAWE, LTOA and TOA [3]

the receive window handling for a downlink data frame. A frame can be received with the following status:

Too Early The data frame is received too early to be buffered in the RBS. The boundary between Early and Too Early depends on the available buffer space in a particular Node B.

Early The data frame is received before the desired receive window but can be buffered until the scheduled transmission time.

OK The data frame is received within the Receiving Window.

Late The data frame is received after the desired receive window but still early enough to be processed and sent through the radio interface in time.

Too Late The data frame is received too late to be transferred. This might be before the scheduled transmission time but always after the Latest Time Of Arrival (LTOA).

Upon receiving a Timing Adjustment control frame, see figure 1.5, the RNC will adjust its transmission time according to the supplied Time of Arrival (TOA) value in order to minimise the buffering required in the RBS.

The Receiving Window boundaries are specified through the TOA Window Startpoint (TOAWS) and the TOA Window Endpoint (TOAWE). These parameters are initially configured in Node B at Transport bearer Setup but can also be reconfigured from the RNC.

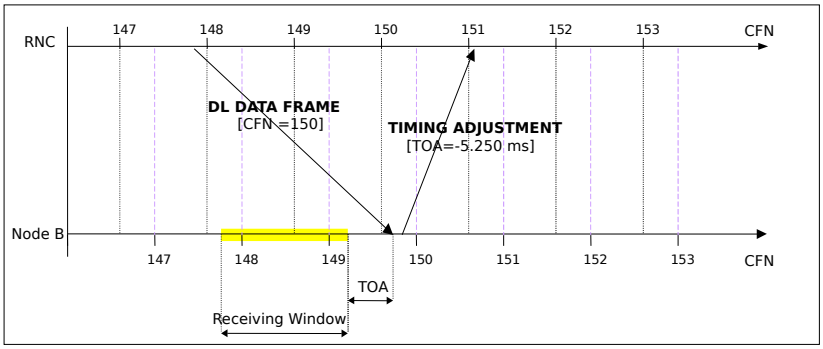

Figure 1.5. Timing Adjustment Procedure [3]

1.2.3 Forward Access Channel (FACH)

The Forward Access Channel (FACH) is a downlink transport channel belonging to the common transport channels[11] which sends downlink data frames to User Equipment (UE). The FACH is transmitted over the entire cell [1].

[11] A common channel is accessible by all UEs within a network cell whereas a dedicated channel is unique for a particular UE.

Chapter 2

Method

The master thesis projects started by building an ATM tunneling software for the available hardware platform. This process is described in section 2.3. When a working tunnel implementation had been achieved a methodology for evaluating the tunnel behaviour at different network characteristics was developed. The test methodology is described in section 2.1. The setup of the equipment used to perform the actual testing is described in section 2.2.

2.1 Test method

The basic tests were to see if a network cell could be setup with one or more transmission channels using the tunnel for ATM transport. This was done using both a real RNC [16] and an RNC simulator. Both setups proved successful but were limited to using a LAN between the tunnel endpoints.

Due to limited access to a real RNC another approach had to be taken in order to answer the question if the tunnel could be used to connect an RNC with an RBS over a UDP/IP WAN. Although access to an RNC was limited it was possible to measure the network characteristics of an IP-based WAN within Ericsson usable for ATM tunneling. Figure 2.1 describes the typical delay dis-

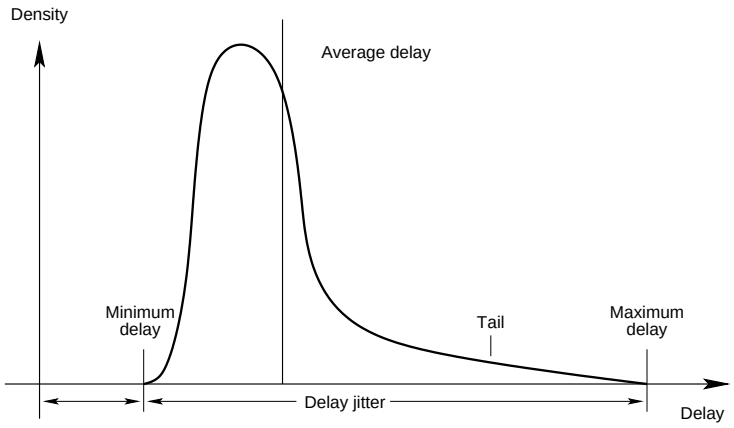

Figure 2.1. Delay distribution and delay jitter [27]

tribution and delay jitter of a packet switching network. By measuring these quantities within the existing WAN the results could be used to create a simulated delay scenario where the tunnel performance could be tested. The software platform and the techniques used to simulate the desired network characteristics are described in section 2.1.1.

A test network with configurable delay distribution and delay jitter makes it possible to see how the ATM tunnel would operate in a real WAN setup. However, it would also be interesting to measure how the RBS responds to the latency imposed by the tunnel. This would make it possible to give a more detailed answer to the question if the tunnel could be used for linking RNC and RBS using a UDP/IP WAN. Section 2.1.2 describes a method to accomplish such a measurement.

2.1.1 Software platform

Debian GNU/Linux [10] was chosen as the software platform. In this case the particular GNU/Linux distribution is not significant as long as it provides `libatm` [5] and an Network Time Protocol (NTP) client.

A custom 2.6 kernel[1] was compiled for the tunnel endpoint machines in order to enable three important options:

- ATM networking and the ATM driver provided by PROSUM [21].
- Quality of Service and Network Emulation, `NetEm` [13].
- 1000Hz kernel timer for higher `NetEm` precision.

The tunnel machines synchronise their clocks with a Symmetricom[2] NTP-server, stratum 1, using a NTP client with an update rate of 1 Hz yielding a perceived precision of 0.1ms or lower. The round trip time and latency in each direction for UDP packets traversing the network could then be monitored using a UDPping[3] software developed specifically for the thesis project.

The Linux kernel includes sophisticated traffic control facilities with advanced Quality of Service (QoS), Differentiated Services (DiffServ) and filtering capabilities [15]. `NetEm` provides network emulation capabilities and can simulate constant delay, delay jitter, packet loss, packet re-ordering etc.

For example to enable an egress delay of 10ms with 3ms delay jitter a simple command such as the following is the only thing required:

```
tc qdisc add dev eth0 root netem delay 10ms 3ms
```

2.1.2 Time of Arrival

The Transport Channel Synchronisation, see section 1.2.2, mechanism used for all downlink transport channels will send a Timing Adjustment control frame to the data frame sender, i.e. the RNC, only if the data frame was received *outside* the Receiving Window. Hence if the Iub has an acceptable latency a

[1] Linux kernel 2.6.16.29 to be exact.

[2] `http://www.symmetricom.com/`

[3] UDPping extends the ICMP ping/pong [25] scheme and includes a timestamp in its *ping* and *pong* packets enabling measurement of the network latency in each direction. However, this approach requires the two nodes' clocks to be synchronised.

real RNC would adjust its frame transmission time so that the data frames are received within the RBS receiving window and no timing adjustment control frame would be sent after a stable transmission timing has been found.

The use of an RNC simulator which is not configured to adjust its downlink data frame transmission timing creates an interesting situation. In this case each downlink data frame will cause a Timing Adjustment control frame to be sent in return since the data frame will always arrive Early. If the maximum latency on the Iub stays within the time window of Early arrival a Timing Adjustment control frame will be sent in return to each transmitted downlink data frame.

The Timing Adjustment control frame contains a Time of Arrival (ToA) value together with the indication of whether the data frame was Early, OK, Late or Too Late. As illustrated in figure 1.4 the ToA value uses the TOA Window Endpoint (TOAWE) as reference with a negative value if the data frame was late and positive value if the frame arrived within the window or was early. The earlier a data frame arrives the higher ToA value.

By observing the ToA returned in the Timing Adjustment control frame returned for every data frame sent using the RNC simulator the Iub characteristics can be measured using the RBS.

2.2 Test setup

The equipment used to verify the effects on the Iub using an ATM tunnel is setup as illustrated in figure 2.2. ATM traffic on the Iub between RNC and RBS is transparently forwarded through an ATM tunnel setup consisting of two tunnel endpoint nodes communicating over a Local or Wide Area Network (LAN or WAN).

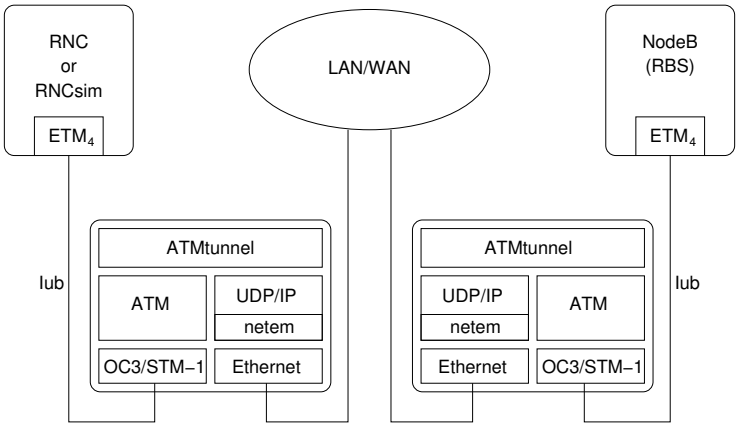

Figure 2.2. A Radio Network Controller (RNC), real or simulator, connects to a Node B base station (RBS) using the RNC–RBS interface (Iub). Two identical ATMtunnel nodes interface with RNC/RBS using the operating system's Asynchronous Transfer Mode (ATM) stack with an optical carrier 3, OC3/STM-1, 155.52 Mbit/s fiber optic interface. The ATMtunnel nodes communicate using the UDP/IP protocol over a Local or Wide Area Network (LAN or WAN). Artificial delay jitter can be introduced between the nodes using the network emulation tool `NetEm`

2.3 Implementation

The first decision on tunneling ATM traffic was to decide at which layer in the ATM protocol model, figure 2.3, the tunnel would operate. The physical

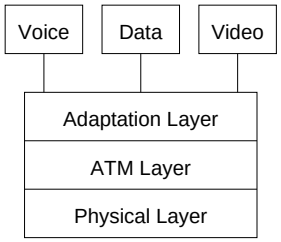

Figure 2.3. Simplistic form of ATM Reference Model [20]

layer is naturally left to the hardware and in order to avoid the complexities in the adaptation layer (AAL) it was decided to let the tunnel operate on the ATM layer. Forwarding raw ATM cells with no adaptation layer processing is a simple and flexible solution. A tunnel operating on the ATM layer has no software dependencies on the physical medium and can handle all adaptation layer protocols transparently. It might have been more efficient to utilise hardware processing[4] in the adaptation layer but the tunnel software would then become much more complex.

The second decision was whether the tunnel should appear as an ATM switch or just forward ATM traffic transparently. Since the tunnel was intended to substitute a real ATM link for testing purposes the later solution was chosen. Forwarding ATM cells over the tunnel without the knowledge of the endpoint equipment has the advantage of the ability to replace a real ATM link with a tunnel based link without reconfiguration.

2.3.1 Hardware

Two "old" Intel-based workstations were equipped with Prosum STM-1[5] ATM cards [21]. After correspondence with the card manufacturer [9] they supplied a modified Linux driver with support for capturing raw ATM cells. The original driver used AAL5 frame buffering based on the most significant bit of the PTI header field which caused arrival of some AAL0 cells to fail to generate an interrupt.

2.3.2 Development process

In order to quickly verify the idea of tunnel ATM traffic using commodity hardware and free software a simple TCP based tunnel was implemented. This implementation established a TCP connection between the tunnel nodes, see figure 2.2, where ATM cells were transfered as 52 bytes[6] fixed size data blocks.

[4] E.g. cell multiplexing/demultiplexing

[5] The STM-1 (Synchronous Transport Module) is the basic rate of transmission of the SDH ITU-T fiber optic network transmission standard. It has a bit rate of 155.52 Mbit/s [19].

[6] An ATM cell is 53 bytes but the Linux ATM socket API [5] does not provide the 8 bit HEC (Header Error Check) to user space.

The TCP based tunnel worked well enough in a LAN environment for a real RNC to successfully establish voice and packet data channels through an RBS to a set of UEs represented by mobile phones from different vendors. As expected the TCP approach suffered from latency problems and did not work satisfactory when artificial network delay was introduced.

The TCP based tunnel was reworked to use a simple UDP based solution. Each 52 byte cell was sent unmodified in a single UDP packet. The latency problems seen in the TCP implementation was much improved and likewise the tolerance for artificial network delay. However since the UDP implementation was based on the assumption that packet reordering is unlikely in a LAN environment it had no tolerance for delay jitter. A fundamental difference between ATM and IP is that ATM guarantees cell order while IP does not. It is okay for the ATM layer to drop a cell but it must not deliver cells out of order.

The good results from the "proof of concept" implementations motivated the development of a more sophisticated ATM tunnel software. The following requirements were placed on the final tunnel design:

- The tunnel must not reorder ATM cells even if the transport protocol reorders packets.
- The tunnel must be able to properly handle delay jitter.
- The jitter handling must have a timeout function which discards cells that have not arrived within a specified receive window.
- The tunnel must be able to transfer several ATM cells within a single IP packet.
- The tunnel should impose as little delay as possible when transferring an ATM cell.

These requirements are easily motivated. The ATM layer guarantees cell order and operating on the ATM layer so must the tunnel. An Internet connection almost always has some sort of delay jitter, it might be small for high quality links but it is still there and so must be addressed by the tunnel. Jitter handling, i.e. the process of replaying received data in the order it was sent, will impose some kind of delay. If for example cells A, B, C, D was sent and C, D, A was received the receiver can immediately send cell A but must wait for cell B before it can send C and D. This waiting period must be configurable or else it could result in unpredictable latency being imposed by the tunnel. IP does generally not have optimal performance when the packet size is small. Therefore in order to address both low latency and high throughput the tunnel must be able to detect bursts of ATM traffic and send several cells within the same IP packet.

2.3.3 Tunnel protocol

In order to maintain low latency and have control over each packet being exchanged by the tunnel endpoints UDP/IP was chosen for the final implementation as well. A simple stateless packet protocol was used for cell transfer. The packet format, figure 2.4, uses a four byte header containing cell count and cell sequence number.

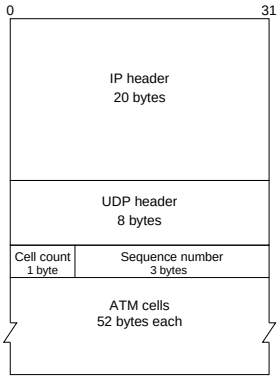

Figure 2.4. ATM tunnel protocol packet format

Cell count 8-bit integer reflecting the number of 52-byte ATM cells sent in the packet. The maximum of 255 cells gives a maximum IP packet size of $20 + 8 + 4 + 255 \cdot 52 = 13292$ bytes.

Sequence number 24-bit sequence number of the first cell in the packet. The sequence number reflects to ATM cells, not tunnel packets. The sequence number wraps around after approximately 16.7 million cells which corresponds to $\frac{2^{24} \cdot 48}{1024^2} = 768$ megabytes of ATM cell payload[7].

The packet header was designed to add as little overhead as possible while still providing robust sequence number handling even for high bandwidth scenarios. The overhead for single cell transfer $\frac{28+4}{28+4+52} = 38.1\%$ is rather high due to the 28 bytes UDP/IP header.

The maximum packet size should be selected according to the MTU of the tunnel link. Although the packet format supports large packets the resulting IP fragmentation might cause latency issues which cannot be controlled by the tunnel software. A typical LAN environment with an MTU of 1500 bytes can transfer 28 cells, $20 + 8 + 4 + 28 \cdot 52 = 1488$ bytes packet, without fragmentation giving an overhead of $\frac{28+4}{1488} = 2.2\%$. An Internet scenario using the minimum MTU for IP networks [25] of 576 bytes can transfer 10 cells giving a packet size of 552 bytes and an overhead of 5.8%.

2.3.4 Algorithm

The ATM tunnel implementation has been divided in two parts, the INET sender and the INET receiver. The INET sender reads cells from the ATM device and sends them using the tunnel protocol to the other tunnel endpoint. The INET receiver reads packets sent by the other tunnel endpoint and sends the received cells through the ATM device, see figure 2.5. The sender and receiver within the same tunnel instance are completely independent of each other and maintain their own cell sequence number handling. However there is one configuration parameters used by both sender and receiver:

[7] An ATM cell carries 48 bytes of payload. However the information specific to the adaptation layer (AAL) is located within these 48 bytes.

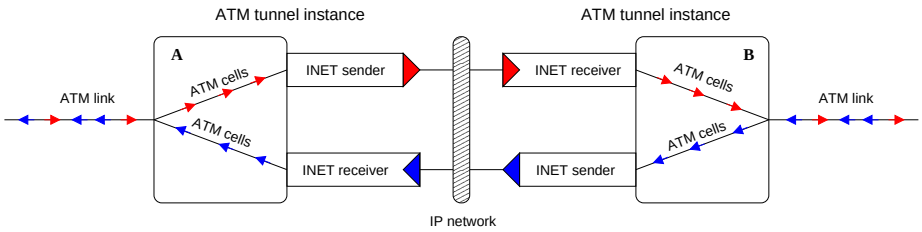

Figure 2.5. ATM tunnel overview

`packet size` the maximum number of cells allowed to be sent *or* received in a single UDP packet. The packet size parameter should be set so that IP fragmentation is avoided. This parameter must have the same value in both tunnel instances.

INET sender

Configuration parameter:

`send buffer delay` the maximum time waiting for cells to fill the packet buffer before transmission.

The INET sender uses a single cell buffer for incoming cells. It does only buffer cells for one packet at a time. Packet transmission is initiated when the buffer is full, i.e. the `packet size` limit has been reached, or the timeout imposed by the `send buffer delay` has expired. The algorithm of the INET sender can be described as follows:

1. Set $seq_{next} = 0$.
2. Set $cell_{index} = 0$.
3. Set $timeout = time_{current} + send\_buffer\_delay$.
4. While $time_{current} < timeout$ and $cell_{index} < packet\_size$
   (a) Read 52-bytes ATM cell[8] (the call must respect the timeout).
   (b) Store cell in packet buffer at position $4 + cell_{index} * 52$.
   (c) Set $cell_{index} = cell_{index} + 1$.
5. Write packet header using cell count $cell_{index}$ and sequence number seq_{next} to the first 4 bytes of the packet buffer.
6. Send the packet buffer contents to the remote tunnel endpoint.
7. Set $seq_{next} = seq_{next} + cell_{index}$.
8. Goto 2.

[8] ATM cell with HEC removed as presented by the Linux ATM socket API [5].

INET receiver

Configuration parameters:

`replay window size` the maximum number of cells handled by the replay window (jitter buffer).

`replay window delay` the maximum time waiting for missing cells to arrive.

`replay window reset timeout` the maximum time received cells are allowed to have a sequence number outside the replay window, thus being discarded, before the window is reset.

The INET receiver must be able to replay received cells in the order they were sent when forwarding them to the ATM link. To accomplish this the INET receiver uses a replay window (jitter buffer) and a receive time list. The replay window buffer stores cells at a position relative to their sequence number, see figure 2.6. The receive timestamp list contains pointers to the received cells

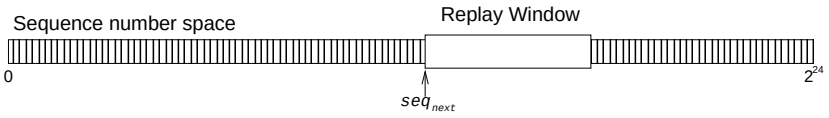

Figure 2.6. The replay window is responsible for a subset of the sequence number space

sorted by time of arrival. The oldest cell in the replay buffer has the first entry in the receive list, see figure 2.7. The receive timestamp list is necessary to

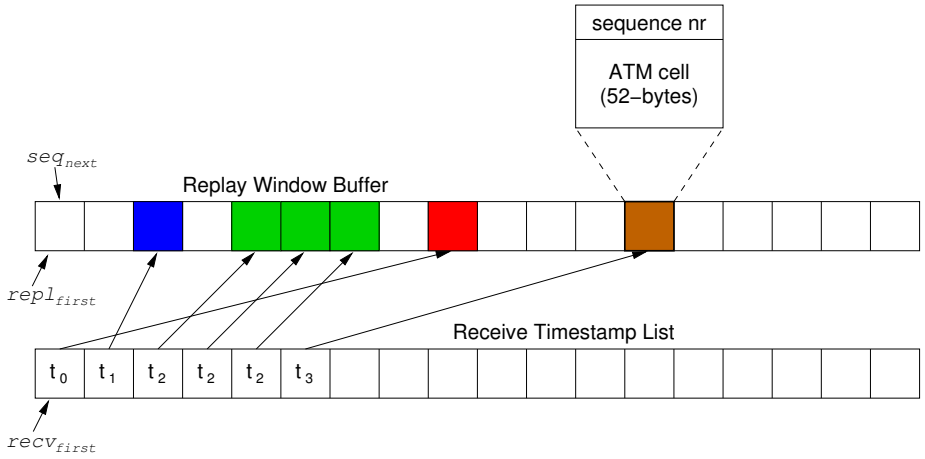

Figure 2.7. Relationship between replay window buffer and receive timestamp list

maintain a complexity of $\mathcal{O}(1)$ in the receive algorithm.

Both replay window buffer and receive timestamp list are implemented as single linked lists with entries stored sequentially in an array. The expected sequence number of the next cell to arrive is called seq_{next}. The replay window's first entry, $repl_{first}$, is the buffer slot where the cell with sequence number seq_{next} is to be stored upon arrival. The oldest cell currently stored in the

replay buffer has its arrival timestamp stored in the first entry, $recv_{first}$, of the receive timestamp list. Each entry in the receive list has a pointer to the corresponding cell in the replay buffer. Cells are stored in the replay window buffer until all cells with lower sequence number, if any, has arrived and been sent through the ATM device or until a timeout according to the `replay window delay` occurs.

The replay scenario shown in figure 2.7 is an example where four UDP packets with cells have arrived and the next expected cell, seq_{next}, is still missing. The cells arrived as follows:

1. The "red" cell arrived at t_0 with sequence number $seq_{next}+8$. The received UDP packet only contained this one cell.

2. The "blue" cell arrived at t_1 with sequence number $seq_{next} + 2$. This cell also arrived in an UDP packet with only one cell.

3. The three "green" cells arrived at t_2 in a single UDP packet. They have sequence number $seq_{next} + 4$, $seq_{next} + 5$ and $seq_{next} + 6$.

4. The "brown" cell arrived at t_3 with sequence number $seq_{next} + 12$.

At this stage the INET receiver will not send anything through the ATM device. It will wait for more cells to arrive until either seq_{next} arrives or $t_0 + replay\_window\_delay < t_{current}$. Assuming the timeout of the later case occurs the missing cells with sequence numbers $seq_{next}+\{0, 1, 3, 7\}$ will be skipped while the buffered "blue", "green" and "red" cells are sent in sequential order[9]. After the cells have been sent seq_{next} is set to $seq_{next} = seq_{next} + 9$ and the previously occupied entries in the replay buffer and receive list are cleared. The list pointer $repl_{first}$ is set to point at the entry after the "red" cell and $recv_{first}$ to point at the entry containing t_3. Timestamp t_3 can then be considered as the new "t_0".

The INET receiver algorithm is described below in a simplified form. The following features found in the real implementation are not included in the algorithm description:

- Proper sequence number handling with support for wrap around.
- Out of replay window handling.
- Out of replay window reset timeout.
- Non-blocking I/O.

INET receiver algorithm:

1. Read UDP packet from remote tunnel endpoint or timeout. Timeout occurs if $t_0 + replay\_window\_delay < t_{current}$.

2. If a packet was successfully received:

   (a) Save arrival timestamp $t = t_{current}$.

[9]The "gaps" are processed one at a time so e.g. cell $seq_{next} + 3$ could arrive while the first two cell entries are skipped.

   (b) Extract sequence number, *seq*, and $cell_{count}$ from the packet header, see section 2.3.3.

   (c) Find the replay window entry corresponding to *seq* by offsetting $repl_{first}$ with $seq - seq_{next}$[10]. This is an $\mathcal{O}(1)$ operation since the replay window buffer is implemented as a single linked list *stored* as an array.

   (d) Copy the $cell_{count}$ ATM cells from the packet to the replay window buffer.

   (e) Add $cell_{count}$ entries into the end of the receive timestamp list using timestamp t with pointers to the corresponding entries in the replay window buffer.

3. If the replay window buffer is non-empty:

   (a) If the replay buffer entry at $repl_{first}$ is empty, goto 4.

   (b) Send the cell at $repl_{first}$ through the ATM device.

   (c) Clear the replay buffer entry at $repl_{first}$ and the corresponding receive timestamp list entry.

   (d) Update $repl_{first}$ to point to the next replay buffer entry.

   (e) Increment seq_{next} by one.

   (f) Goto 3a.

4. If $t_0 + replay\_window\_delay < t_{current}$:

   (a) If $repl_{first}$ refers to a non-empty replay buffer entry, goto 5.

   (b) Update $repl_{first}$ to point to the next replay buffer entry.

   (c) Increment seq_{next} by one.

   (d) Goto 4a.

5. Goto 1.

[10] This is a simplification since the real implementation correctly handles sequence number wrap around.

Chapter 3

QoS in IP networks

The network technology to use when Quality of Service (QoS) is a requirement has long been thought to be ATM. The transport mechanism in ATM built on the switching of small cells was motivated by the low latency requirements in telecommunication networks. Today modern fiber optic networks provide such speeds that the queueing delay of even full length 1500 byte packets incur no significant delay. Furthermore there are scalability issues of the extra ATM layer starting to become apparent at speeds increasing above 1 Gbit/s and the cost of maintaining both high speed IP and ATM networks can be significant [20] [28].

The Internet datagram model used by IP networks provide no resource guarantees and does not distinguish between different types of services. This is a severe limitation that is starting to become more apparent as the demand of new services with specific network requirements increase in popularity [27].

The need for QoS in IP networks and the increasing cost of maintaining high speed ATM networks have motivated the development of several new technologies. Differentiated Services (DiffServ), see section 3.1, provides a mechanism for scalable QoS capabilities in IP networks. Multiprotocol Label Switching (MPLS), see section 3.2, introduces a new approach to packet forwarding which can incorporate the demands of both Internet users and ATM deployers within the world of telecommunications.

The Connectivity Packet Platform (CPP) developed by Ericsson is a carrier-class technology that has been positioned for access and transport products in mobile and fixed networks. CPP is deployed within Ericsson's UTRAN equipment and has support for IP technologies such as DiffServ and MPLS to provide cost effective alternatives to ATM for service providers [28].

3.1 Differentiated Services

The Differentiated Services (DiffServ) networking architecture was published by the IETF in 1998 [8] and specifies a relatively simple, scalable and course-grained mechanism providing QoS capabilities for Internet traffic. DiffServ enables resource allocation by use of packet classification, traffic policing and class-based forwarding. Rather than trying to guarantee network resources for individual end-to-end flows DiffServ divides traffic into a fixed number of forwarding classes for which resources are allocated. Resource allocation in DiffServ is thus per-

formed for aggregated traffic and although different service levels and resource assurance can be provided absolute amounts of bandwidth or fixed delay bounds cannot be guaranteed for individual flows [27].

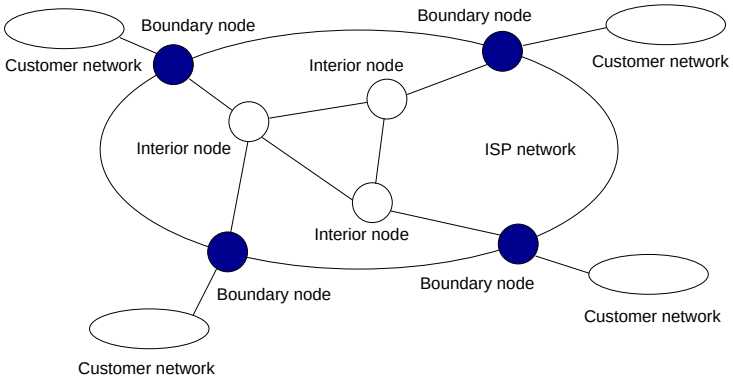

Figure 3.1. Boundary and interior nodes [27]

Traffic conditioning and classification are performed at the boundary nodes of a DiffServ network and interior nodes need only forward packets based on their classification information, see figure 3.1. DiffServ redefines a field in the IP header, the Differentiated Services Field (DS field) is the six most significant bits of the (former) IPV4 TOS octet or the (former) IPV6 Traffic Class octet [12]. The DS field is used to carry the forwarding class index called Differentiated Services codepoint (DSCP). Interior nodes in a network deploying the DiffServ architecture forward packets based on so called Per-Hop Behaviours (PHBs) which express the externally observable forwarding behaviour applied at a node compliant with DiffServ. The PHB to be used for a particular packet is determined by the DSCP in the DS field of the packet through a configurable mapping between DSCP and PHB. Since forwarding is solely based on the information provided in the packet header no resource reservation mechanism is needed in DiffServ [8] [27].

3.1.1 Traffic classification

A DiffServ boundary node is responsible for mapping incoming traffic to one of the forwarding classes, i.e. a PHB, defined for the network. A boundary node must also ensure that the classification process follows the Service Level Agreement (SLA) between the customer and the network owner [27].

Packet classification, also called packet filtering, is the same technique as used in firewalls widely deployed all over the Internet. A set of rules applicable to the packet header information are used to apply actions for each packet. Actions can be to e.g. drop packets, apply traffic conditioning, define forwarding classes or override routing decisions [27].

The packet classification policy applied by a boundary node selects the subset of traffic which may receive a differentiated service. Traffic can be classified by changing updating the DS field with the desired DSCP and/or being scheduled for traffic conditioning [8].

3.1.2 Traffic conditioning

Traffic conditioning is the process of policing traffic according to the Traffic Conditioning Agreement (TCA). The TCA consists of the traffic conditioning rules specified in the SLA together with the rules implicit from the relevant service requirements and/or from a DiffServ network's service provisioning policy. Traffic conditioning is performed at the DiffServ boundary nodes and a conditioner can contain four elements: meter, marker, shaper and dropper [8] [27].

Interior nodes may be able to perform limited traffic conditioning functions such as DSCP re-marking [8].

3.2 Multiprotocol Label Switching

The Multiprotocol Label Switching (MPLS) architecture [22] was organised by the IETF to introduce label switching within IP networks and to allow for seamless IP/ATM integration. Although IP was the initial focus when the IETF started the standardisation process of MPLS it is motivated by a several factors [20] [27]:

- MPLS greatly simplifies IP forwarding making it much more efficient and easy to implement in hardware.
- MPLS allows IP to be transferred over existing ATM networks in a less cumbersome way than e.g. Classical IP over ATM.
- ATM can be transferred over MPLS networks while maintaining guaranteed Quality of Service (QoS).
- MPLS provides explicit routing whereby a node can explicitly specify the destination path rather than having to make a hop-by-hop decision at each node.
- MPLS makes Traffic Engineering easier, i.e. the process of using the different links in a network optimally.

Routing in a classical IP network is performed on a "per-hop" basis where each router decides the next-hop based on the destination address in the IP packet header. The IP routing algorithm uses the *longest-prefix match* approach which is inefficient and expensive to implement in hardware [20].

Figure 3.2 illustrates a simplified overview of MPLS Label Switching. MPLS uses the term Forwarding Equivalence Class (FEC) to describe packets which receive the same forwarding treatment. An ingress Label Switching Router (LSR), a boundary node to the MPLS network, maintains a Classification Table by which incoming packets are matched to a FEC and assigned a label. The forwarding decision is then based solely on the label, thus the name label switching. The concept of label switching in MPLS is closely related to the VPI/VCI and VCI/VPI translation in ATM. Each core[1] LSR has a lookup table for incoming label and interface translating to an outgoing label and router interface. Since the label mapping is fixed at each LSR the Label Switched Path (LSP)

[1] Interior network node.

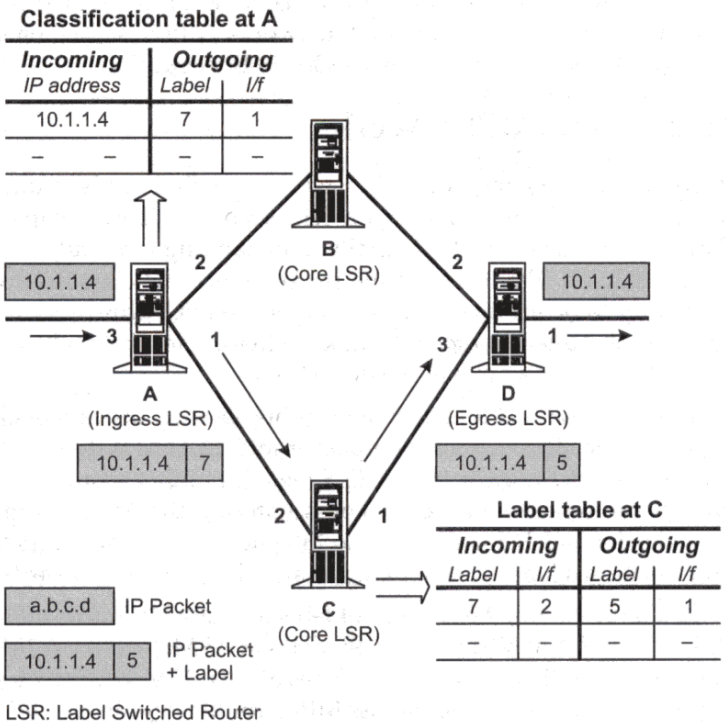

Figure 3.2. MPLS Label Switching [20]

a packet will traverse can be determined from the label assigned at the ingress LSR [20] [27].

An MPLS network need to be configured to populate the label lookup tables at each participating LSR. This process is performed by a Label Distribution Protocol (LDP) of which there exist two standardised protocols, CR-LDP and RSVP-TE [27]. However in February 2003 the MPLS working group decided to focus future efforts only on RSVP-TE and undertake no further work on CR-LDP [6].

Chapter 4

Results

A working software solution for tunneling ATM traffic over an UDP/IP network has been developed and implemented, see section 4.1. The tunneling software has been tested with an RNC simulator and a Node B base station using independent timing measurements collected from the Node B. The results are presented in section 4.3.

4.1 ATMtunnel software

The ATMtunnel software developed in this master thesis project implements the specification described in section 2.3 as a POSIX[1] -compliant application written in C. The application has been thoroughly tested and verified and uses a single threaded `select()`-based approach for high predictability and easy debugging. The whole reception and transmission path within the application including buffer handling are $\mathcal{O}(1)$ with regard to the processing of an ATM cell.

The ATMtunnel application provides the following key features:

- ATM traffic is forwarded on the transport layer (AAL0) which provides support for all ATM Adaptation Layers.
- ATM forwarding is performed transparently making ATM reconfiguration unnecessary.
- ATM cells are forwarded through the tunnel as fast as possible.
- Delay jitter causing re-ordering in the UDP transport will activate the cell replay functionality to guarantee cell order.
- The maximum delay introduced by the cell replay handling is configurable.
- A properly received cell will not be delayed longer than the specified maximum delay.
- Cells arriving "too late" will be discarded as is allowed in the ATM transport layer while cell order is maintained.

[1] Portable Operating System Interface.

- Sequence number out-of-sync scenarios caused e.g. by network outage is addressed by the configurable reset timeout.
- ATM traffic is forwarded based on PVC. One tunnel instance per PVC enables different tunnel settings for different ATM transports.
- Configurable buffer fill delay for increased efficiency of UDP/IP network utilisation.
- The ATMtunnel can be configured to display statistics of forwarded cells, packets, cell loss, etc.

The ATMtunnel software has configurable behaviour and is invoked as follows:

```
ATMtunnel by Ralf Nyrén (C) 2006-2007

Usage: atmtunnel [-dvh] [options] [itf.]vpi.vci host port
Options:
  -p  max number of cells in UDP packet
  -w  max send delay waiting for cells to fill UDP packet
  -r  replay window size (cells)
  -D  replay window delay waiting for missing cells
  -T  replay window out-of-sync reset timeout
  -L  localaddr[:localport]
  -c  cell/packet counter print timeout
```

4.2 Reference delay distribution

Before the ATMtunnel software could be tested, the delay distribution and delay jitter had to be determined for the network proposed for ATM tunneling, see section 2.1. Figure 4.1 shows the results from measuring the Round Trip Time (RTT) of a network[2] within Ericsson. The RTT was measured using ICMP ping

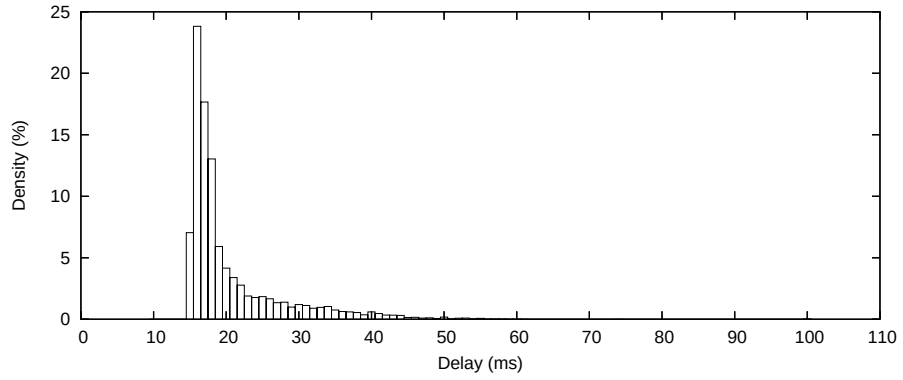

Figure 4.1. ICMP ping delay distribution measured within an Ericsson network

and resembles the delay distribution characteristics illustrated by figure 2.1. The measured delay distribution illustrated in figure 4.1 provides the reference for

[2]Ericsson IP network between Umeå and Kista (Stockholm).

the artificial delay used to simulate the effects of using the ATMtunnel software in a WAN. The mean value of the delay distribution was 20.5ms.

4.3 Test results

The tests were performed using an RNC simulator connected via two ATM tunneling endpoint machines to a Node B base station as described in section 2.2.

The effects of the ATM tunnel on the Iub were measured using the Time of Arrival ToA value of the Timing Adjustment control frame sent by the RBS as described in section 2.1.2. The ToA values correspond to the downlink transport frames sent from RNC simulator to RBS over the Iub.

The delay distribution of the real network described in section 4.2 was simulated using Linux `NetEm` with a constant delay parameter of 10ms and ±3ms delay jitter. This corresponds to an approximate round trip delay of at least 20 ± 6ms.

4.3.1 FACH transport

The traffic chosen for the tests described in this section were produced by repeatedly sending information on the Forward Access Channel (FACH), see section 1.2.3. Figure 4.2 shows the results of four tests performed to investigate the tunnel's influence on the Iub under different delay distribution scenarios:

4.2a Iub using regular ATM over OC3/STM-1, no tunneling involved.

4.2b Iub using ATMtunnel with a 100Mbit/s LAN without any artificial delay.

4.2c Iub using ATMtunnel with 10ms constant delay[3] in both uplink and downlink, i.e. an RTT of 20ms.

4.2d Iub using ATMtunnel with an average artificial delay of 10ms with ±3ms delay jitter. The delay distribution was chosen to approximately simulate the network characteristics of the Wide Area Network (WAN) described in section 4.2.

Figure 4.2a provides the reference ToA distribution which depends on the setup of the Receiving Window in the Node B and any constant transmission delay configured in the RNC simulator. The average ToA reference value is 32.0ms.

Changing the Iub to use the ATM tunnel gives a slightly lower ToA value in figure 4.2b. A lower ToA value implies higher delay on the Iub which is what should be expected when the tunnel is used. The mean ToA value in this case is 31.8ms which implies 0.2ms higher delay than the reference value. A typical switched 100Mbit/s LAN measures a ICMP ping delay of approximately 0.15–0.20ms.

In figure 4.2c a 10ms constant delay has been introduced by the use of `NetEm`. A wider delay distribution can be seen in this case and it should be noted that `NetEm` depends on the kernel timer with a precision of 1.0ms, see section 2.1.1. The measured ToA values have a mean of 21.1ms which implies a increased delay of 10.9ms.

[3]The delay configured in `NetEm` was 10ms but the actual delay is likely to be somewhat higher due to the nature of `NetEm`.

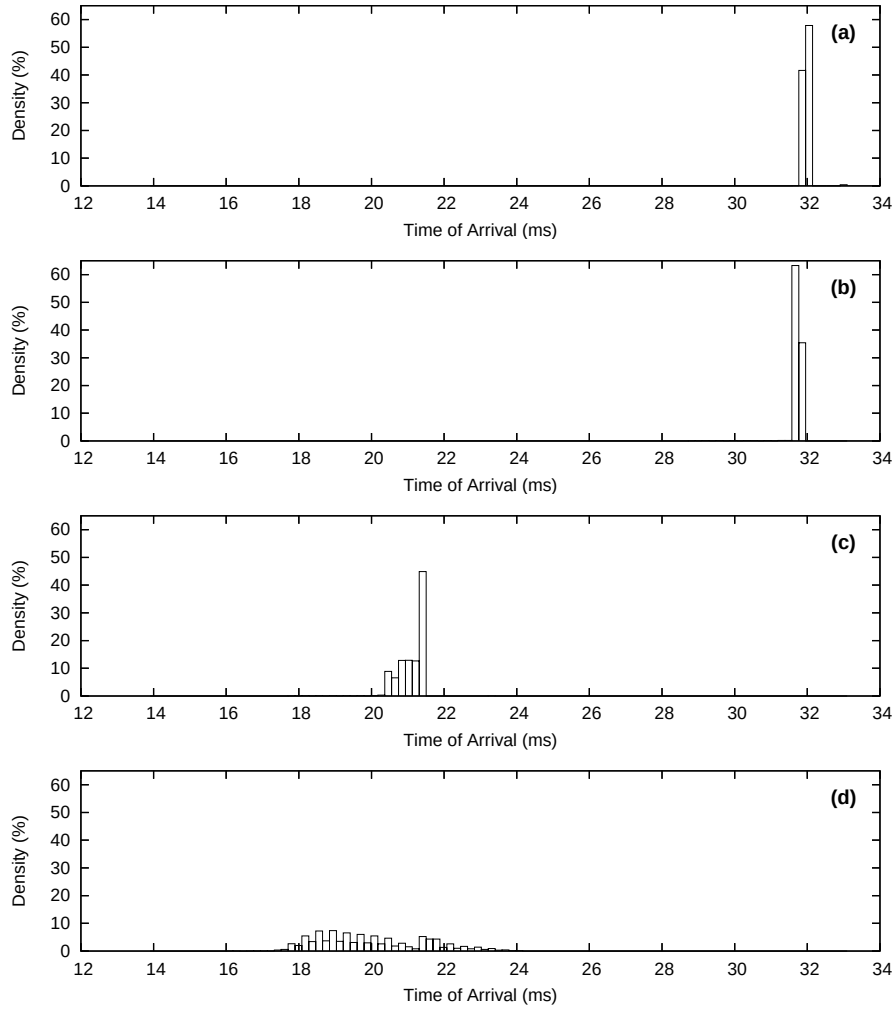

Figure 4.2. Time of Arrival (ToA) of downlink FACH frames measured by Node B. Figure 4.2a shows the reference downlink ToA distribution of the Iub using regular ATM transport, i.e. no tunneling involved. The delay imposed by the ATMtunnel and IP network is the difference between the reference ToA distribution in figure 4.2a and the results in figure 4.2b, 4.2c and 4.2d. Figure 4.2b illustrates the delay imposed on the Iub by using the ATMtunnel in a LAN environment. Figure 4.2c shows the effects of the ATMtunnel when an artificial constant delay of 10ms was added to the IP network using Linux `NetEm`. In figure 4.2d a WAN was simulated using an average artificial delay of 10ms with ±3ms delay jitter

A 10ms delay and ±3ms jitter were used when measuring the Time of Arrival shown in figure 4.2d. The mean value is 20.0ms which corresponds to a 12.0ms average delay. The 2.0ms delay difference as compared to the `NetEm` setting is likely to be explained by the replay window handling in the ATMtunnel and the precision problems in `NetEm`. The ToA value varies from 8.4ms to 24.5ms although the majority represents a delay jitter of approximately 6ms distributed between 17.8ms and 23.8ms.

All FACH frames sent by the RNC simulator during the described tests were successfully transmitted on the radio interface by the RBS.

4.3.2 Dedicated channel packet data transport

Using the ATMtunnel software over an IP network is likely to require some additional transport security. The test presented in this section uses an ATM tunnel setup as described in section 2.2 with the addition of a Virtual Private Network (VPN) installed between the ATM tunnel endpoints. The VPN software used was OpenVPN[4] which is capable of forwarding the ATM tunnel IP traffic using UDP/IP.

The test illustrated in figure 4.3 used packet data traffic at a speed of 384kbit/s on a dedicated channel, i.e. a transport channel dedicated to a particular User Equipment (UE). During the test an artificial delay distribution with an average of 10ms and ±3ms delay jitter was used. The delay was created using Linux `NetEm` with the same settings as in section 4.3.1, figure 4.2d.

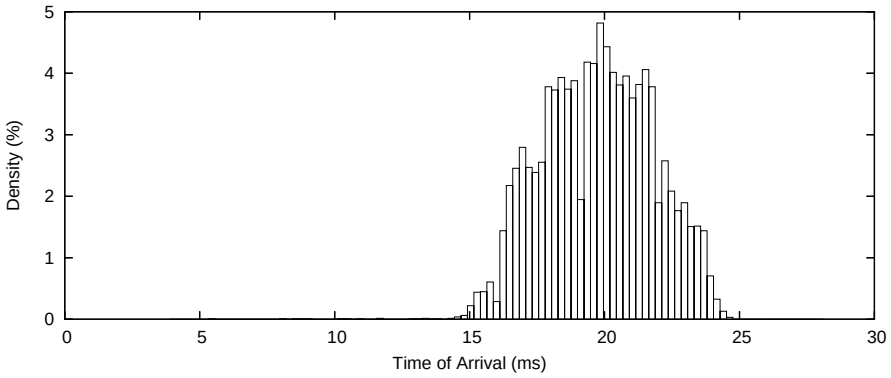

Figure 4.3. Time of Arrival (ToA) of downlink data frames sent on a dedicated transport channel at 384kbit/s. The ATM traffic on the Iub was tunneled over a Virtual Private Network (VPN) using the ATMtunnel software. The IP network used by the VPN was setup with an average artificial delay of 10ms with ±3ms delay jitter

The Time of Arrival distribution shown in figure 4.3 has a mean value of 19.8ms implying an average delay of 12.2ms, using figure 4.2a as reference. Compared to the results presented in figure 4.2d an additional average delay of 0.2ms can be seen when the VPN was used.

[4] `http://openvpn.net/`

Chapter 5

Concluding remarks

This master thesis project proves the concept of tunneling ATM traffic over an IP network for the latency sensitive communication between RNC and RBS in UTRAN.

Comparison of Iub latency between pure ATM based transport and tunneled transport shows an additional delay of approximately 0.2ms in a LAN environment. Typical packet delay in a 100Mbit/s LAN environment is 0.1–0.2ms including IP stack processing implying the developed ATMtunnel software to cause insignificant delay.

Simulation of a real LAN/WAN situation using the Free Software Linux kernel IP stack `NetEm` functionality show predictable results. Based on investigation of an Ericsson WAN, potentially usable for ATM tunneled IP based remote connection of RNC and RBS, it can be concluded that the ATMtunnel software could indeed be used to connect physically remote test equipment for test and verification purposes.

The ATMtunnel software combined with tools such as `NetEm` have proved to be a very useful and cost effective system for test and verification of transmission related issues.

5.1 Limitations

The ATM tunnel setup described in this report does not relay the network synchronisation mechanisms [17] provided by the physical media of the ATM transport, i.e. the fiber optic OC3/STM-1 link. An RBS connected to an RNC using the ATMtunnel software can therefore not rely on the Iub to provide a clock reference signal. However, an external clock source can be provided to an RBS by use of for example an atomic reference clock or a GPS receiver. An RBS test environment is likely to have plenty of reliable clock sources.

5.2 Future work

The impact of the ATMtunnel on the Iub communication could be further investigated for various network characteristics. For example a more realistic delay distribution function could be implemented for use with `NetEm`. A Wide Area Network (WAN) test with a real RNC would certainly also be interesting.

The tests described in this report only used a fraction of the available Iub bandwidth. The performance of the ATMtunnel software during higher bandwidth scenarios could be investigated using traffic models producing extensive ATM load.

There exist other solutions for tunneling ATM traffic over an IP network which would be interesting to compare to the current approach used in this master thesis project. An example is the Layer 2 Tunneling Protocol Version 3 (L2TPv3) which includes an ATM cell relay mode [23] similar to the techniques used in the ATMtunnel software. However, at the time of writing the author could not find a freely available implementation of L2TPv3 with support for ATM over L2TPv3.

Chapter 6

Acknowledgements

The author would like to thank Per Hurtig and Caroline Muzikants at Ericsson for access to and test support in a real UMTS Radio Access Network. The author would also like to thank Marie Dahlberg and Mikael Larsson, RBSIoV, TietoEnator for supporting this master thesis project.

References

[1] 3GPP. *Technical Specification Group Radio Access Network; Physical channels and mapping of transport channels onto physical channels (FDD) (Release 6)*. 3rd Generation Partnership Project, Dec. 2005. 3GPP TS 25.211 V6.7.0.

[2] 3GPP. *Technical Specification Group Radio Access Network; Base Station (BS) radio transmission and reception (FDD) (Release 6)*. 3rd Generation Partnership Project, Dec. 2006. 3GPP TS 25.104 V6.14.0.

[3] 3GPP. *Technical Specification Group Radio Access Network; Synchronisation in UTRAN Stage 2 (Release 6)*. 3rd Generation Partnership Project, Dec. 2006. 3GPP TS 25.402 V6.5.0.

[4] 3GPP. *Technical Specification Group Radio Access Network; UTRAN overall description (Release 6)*. 3rd Generation Partnership Project, Dec. 2006. 3GPP TS 25.401 V6.9.0.

[5] Almesberger, W. *Linux ATM API Draft, version 0.4*, 19 July 1995.

[6] Andersson, L., and Swallow, G. The Multiprotocol Label Switching (MPLS) Working Group decision on MPLS signaling protocols. RFC 3468 (Informational), Feb. 2003.

[7] ATM Forum. *Inverse Multiplexing for ATM (IMA), Specification 1.1*, Mar. 1999. ATM Forum AF-PHY-0086.001.

[8] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and Weiss, W. An Architecture for Differentiated Service. RFC 2475 (Informational), Dec. 1998. Updated by RFC 3260.

[9] Bucari, C. Private Communication. PROSUM Networking Products, June 2006.

[10] Debian. Debian GNU/Linux – The Universial Operating System. Web page, 19 Nov. 2007. `http://www.debian.org/`.

[11] Free Software Foundation. The Free Software Definition. Web page, 1 Nov. 2007. `http://www.fsf.org/licensing/essays/free-sw.html`.

[12] Grossman, D. New Terminology and Clarifications for Diffserv. RFC 3260 (Informational), Apr. 2002.

[13] Hemminger, S. Network Emulation with NetEm. *linux.conf.au* (Apr. 2005). `http://developer.osdl.org/shemminger/netem/LCA2005_paper.pdf`.

[14] Holma, H., and Toskala, A. *WCDMA for UMTS: Radio Access for Third Generation Mobile Communications*, revised ed. John Wiley & Sons, Ltd., Chichester, West Sussex, England, 2001.

[15] Hubert, B., Graf, T., Maxwell, G., van Mook, R., van Oosterhout, M., Schroeder, P. B., Spaans, J., and Larroy, P. *Linux Advanced Routing and Traffic Control HOWTO*, 29 Oct. 2003. `http://lartc.org/lartc.pdf`.

[16] Hurtig, P. Private Communication. Ericsson, RBSIoV, Oct. 2006.

[17] ITU-T. *Synchronization layer functions*. International Telecommunications Union, Geneva, Switzerland, July 1999. ITU-T Recommendation G.781.

[18] ITU-T. *Physical/electrical characteristics of hierarchical digital interfaces*. International Telecommunications Union, Geneva, Switzerland, 29 Nov. 2001. ITU-T Recommendation G.703.

[19] ITU-T. *Network node interface for the synchronous digital hierarchy (SDH)*. International Telecommunications Union, Geneva, Switzerland, 14 Dec. 2003. ITU-T Recommendation G.707/Y.1322.

[20] Kasera, S. *ATM Networks: Concepts and Protocols*. McGraw-Hill Communications, New York, NY, USA, 2006.

[21] PROSUM Networking Products. ATM Network Interface Cards for Fiber Optic and UTP 155 Mbps Connections. Web page, 14 Nov. 2007. `http://www.prosum.net/atm155_E.html`.

[22] Rosen, E., Viswanathan, A., and Callon, R. Multiprotocol Label Switching Architecture. RFC 3031 (Proposed Standard), Jan. 2001.

[23] Singh, S., Townsley, M., and Pignataro, C. Asynchronous Transfer Mode (ATM) over Layer 2 Tunneling Protocol Version 3 (L2TPv3). RFC 4454 (Proposed Standard), May 2006.

[24] Sköld, J., Lundevall, M., Parkvall, S., and Sundelin, M. Broadband data performance of third-generation mobile systems. *Ericsson Review*, 1 (2005).

[25] Stevens, W. R. *TCP/IP Illustrated, Volume 1*. Addison Wesley Longman, Inc., 1994.

[26] Vainshtein, A., and Stein, Y. Structure-Agnostic Time Division Multiplexing (TDM) over Packet (SAToP). RFC 4553 (Proposed Standard), June 2006.

[27] Wang, Z. *Internet QoS: Architectures and Mechanisms for Quality of Service*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[28] Örjan Kling, L., Åke Lindholm, Marklund, L., and Nilsson, G. B. CPP – Cello packet platform. *Ericsson Review*, 2 (2002).