An All-IP Software Radio Architecture under RTLinux

C. Bonnet, L. Gauthier, P.A. Humblet, R. Knopp, A. Menouni-Hayar, Y. Moret, A. Nordio, D. Nussbaum, M. Wetterwald

Corresponding Author:
Prof. R. Knopp
Mobile Communications Department
Institut Eurecom
B.P. 193
06904 Sophia-Antipolis, FRANCE
(+33) 4 93 00 29 06
raymond.knopp@eurecom.fr

Abstract

This paper presents an overview of a software radio architecture for testing quality-of-service (QoS) aware IP data services over a typical third-generation radio interface. The testbed is implemented using a hard real-time micro-kernel known as RTLinux, running beneath the Linux operating system for providing real-time ene-to-end functionality. The testbed runs on a variety of Intel Pentium-based computing platforms including laptops and high-end servers. Layers 1 and 2 are compliant with the 3GPP specifications for TDD operation and layer 3 provides an interconnection with an IPv6 core network. The intent is to study the impact of an IP core network and QoS constraints on the physical and link layers as well as the codesign of physical layer configurations and IP layer networking.

1. Introduction

This paper presents an overview of a hardware/software architecture for an IP (Internet Protocol) experimental software-radio testbed. The physical layer (PHY) as well as link (MAC/RLC) and network layers are implemented as hard real-time threads in the RTLinux POSIX micro-kernel running on standard PCs. Radio signal acquisition is performed using PCI bus technology. The primary goals of the platform are to study system level aspects related to

- radio interface configuration and algorithms
- Medium-Access (MAC) and Radio-Link (RLC) protocol design
- Quality-of-Service (QoS) control in all-IP radio networks
- Integration of IPv6 Mobility Management mechanisms
- Hardware/software co-design for modern radio systems

The radio uses the Time Division Duplex (TDD) transmission specifications from the 3GPP (see www.3gpp.org) for layers 1 and 2. Layer 3 deviates from 3GPP, in the sense that it provides a direct interconnection with an IP-based network at the base station. In this architecture the traditional base station becomes an IP-router or *Radio Gateway (RG)* and performs both routing from a single IP traffic stream towards multiple radio bearers and, at the same time, translation from IP *quality-of-service (QoS)* classes, using for example *DiffServ*, into 3GPP QoS classes and their corresponding physical layer configurations (i.e. error-coding, interleaving delay, modulation format.)

In section 3 we describe the basic architectures for RG's in addition to simpler mobile terminals are presented, along with specific details regarding implementation technologies. Section 3 deals with specifics related to the physical layer (Layer 1), including a brief summary of the implemented parts of the 3GPP specifications including details regarding execution times and software architectures. In section 4 we present an overview of the chosen implementation for layer 2, notably the Medium Access(MAC) and Radio-Link Layers (RLC). Section 5 outlines a proposal for a direct interconnection of the IP-layer to 3GPP layer 2 and discusses its implications on the networking entities.

2. Architectural Overview

In this section we describe the basic architectural characteristics of the experimental platform. The primary design choice was to implement the entire system in software, from the physical to networking layers. Furthermore, we have chosen to implement the system under the control of an open-source hard real-time operating system (RTOS), in this case RTLinux. There were several reasons for these choice:

1) because of it's open-source policy, we have complete control over the integration of our implementation with the rest of the operating system.
2) rapid development of code for the hard real-time environment is possible due to the high-level primitives offered by a POSIX multi-threaded environment.
3) The all-software approach is slowly becoming reality for base station applications where power consumption is less of an issue than production time
4) The approach extends directly to system-on-a-chip (SoC) architectures for low-power applications, where similar RTOS are being proposed.
5) The code is applicable to different types of host computers suitable for different applications, from laptops to PCs all the way up to high-end computational servers
6) Very little special purpose hardware, other than the RF and (simple) acquisition sub-systems is required.

7) Networking functionality and a rich enviroment for developing applications is provided under the operating system

The fourth point above requires some clarification. A primary characteristic of the architecture we propose is that all sub-systems (in this case processes or sub-routines) physically share the same memory space and common data-structures. From an organizational point-of-view this is very convenient, since the organizational complexity resulting from the flexibility offered by modern radio standards (e.g. 3GPP) is astounding. One only has to read the physical layer proposals of the 3GPP [2,3,4,5,6] to see that an all-software approach is an extremely tempting solution to implementing the system. Unfortunately power consumption issues make this approach impossible for hand-held devices (e.g. mobile phones, PDAs, etc.). Nevertheless, a SoC solution can potentially share the same appealing characteristics. Typically the SoC will be centered around a low-power processor core (e.g. StronARM or PowerPC) running an RTOS (e.g. RTLinux, eCos, etc.) which is complemented by a series of special-purpose co-processors (e.g. error-decoding co-processor, FIR Filter banks, FFT co-processors, etc.) for the application at hand. Here the "brains" or the organizational component (data structures, process scheduling, etc.) of the system are contained in the CPU and the "force" is contained in the co-processors. They all share the same memory space so that resources can be quickly shared and data-transfer is less of an issue between sub-systems. Our all-software development is easily ported to this type of architecture, since sub-routine calls for specific signal-processing tasks will be replaced by interrupts to trigger co-processors. Very little of the organizational component will require modification. Mixed FPGA and CPU architectures proposed by companies such as Xilinx (www.xilinx.com), Atmel (www.atmel.com) and Altera (www.altera.com) combine the organizational benefits of a CPU core and the raw computing benefits of FPGAs in a single low-power device, with the added benefit of being able to completely reconfigure even the computing portion of the system.

A real-time micro-kernel called RTX [11] can be used in a Windows NT enviroment and provides features comparable to RTLinux. It does not strive at POSIX compliance however. Tests of soft-modem performance for V90 and DSL systems running as background tasks in a PC have been reported in [12] below.

2.1 Testbed Components

The hardware portion of the current testbed for this architecture consists of 4 elements that are under software-control, namely

- a PCI-bus based data acquisition card (PMC/CARDBUS/PCI form-factors)
- an analog/digital interface (14-bit sampling)
- an up/downconversion RF card using TDD multiplexing

Several high-end radio interfaces are being produced in conjunction with Philips Semiconductors Sophia Antipolis, France (www.philips-semiconductors.com) for the 1900-1920 MHz band 3GPP TDD 3.84 Mchip/s system. The RF architectures are also applicable to the 1.28 Mchip variant that will be deployed in China. Details of the RF subsystem are beyond the scope of this paper. An earlier prototype of this system is described in [1].

The software portion of the platform is developed as an extension (i.e. a network device) to the Linux Operating System (www.linux.org) and makes use of a hard real-time micro-kernel known as RTLinux (www.rtlinux.org) for performing layer 1 and layer 2 3GPP physical and link layer processing. The details are described in sections 3. Network layer functionality is provided by the Linux Kernel's IP networking subsystem and open-source extensions (e.g. Mobile IPv6).

The basic hardware/software configurations for Radio Gateways (RG), a combination of a basestation and an IP router, and Mobile Terminals (MT) are shown in Figure 1, Figure 2.

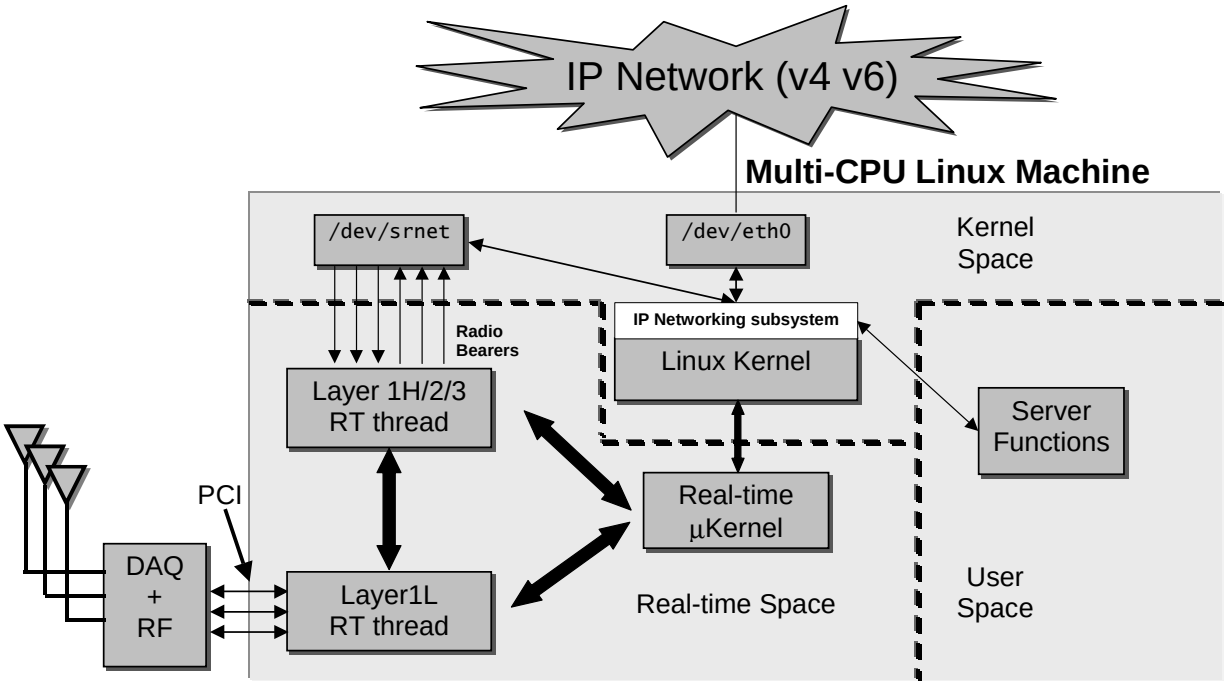


Figure 1: Radio Gateway Architecture

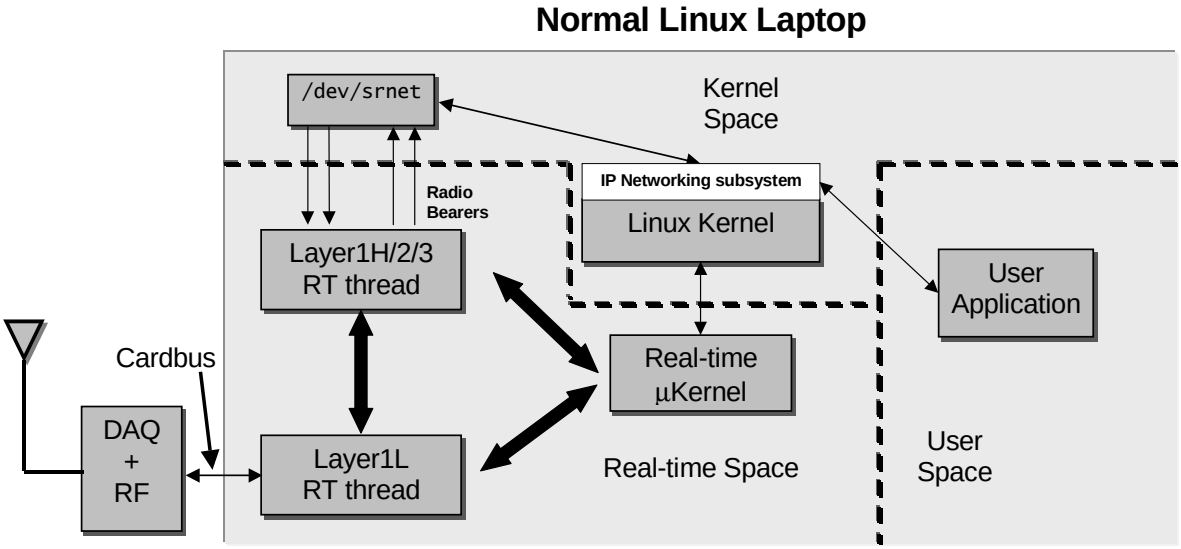


Figure 2: Mobile Terminal Architecture

The RG's network connection is IP based and is assumed to be connected to other RG's on an IPv4 or IPv6 network via a local ethernet using the standard Linux `/dev/eth0` ethernet device. Other networking entities such as mobility and security management servers and gateways to the Internet could also be connected to the IPv4/IPv6 backbone in order to control the traffic between RG's. This is the subject of several French RNRT (www.telecom.gouv.fr/rnrt) (@irs++, PLATON) as well as a European consortium projects (MOBYDICK, www-int.berkom.de/~mobydick) and will make use of the testbed described here.

The 3GPP to IP network interconnect is made via a homemade RTLinux driver `/dev/srnet`. Its basic functions include

- IP routing for a mobile (local-area) IP subnet
- IP-to-3GPP Quality-of -Service (QoS) translation
- Strict real-time implementation of 3GPP Layers 1 (PHY) and 2 (MAC/RLC)
- Minimal RRC signaling functionality to accommodate a set of mobile-IP management functions (attach, resource requests, authentication, identification, etc.)

The entities comprising the 3GPP device driver are collectively known as the **Access Stratum (AS)** in 3GPP terminology, whereas entities in the IP backbone are collectively known as the **Non-Access Stratum (NAS)**.

The computational complexity of the RG will depend on the number of channels and/or antennas that are required. The actual host computer can range from a dual-processor server machine, for a *pico-RG* application, to a powerful *symmetric multi-processor (SMP)* server (e.g. COMPAQ Proliant 6500,8500, www.compaq.com), for *micro-RG* cellular applications or advanced smart-antenna pico-RG systems. This type of multi-CPU architecture is shown in Figure 3 and makes use of several parallel PCI busses, each connected to a separate RF subsystem. In this case, antennas could first be used in a "smart"fashion, namely where each operates across the same frequency channel and joint processing is performed (e.g. beamforming, space-time coding, etc.) Additionally, they could each be used independently for several parallel frequency channels. This could be the case where operators share a common piece of equipment to reduce operation costs.

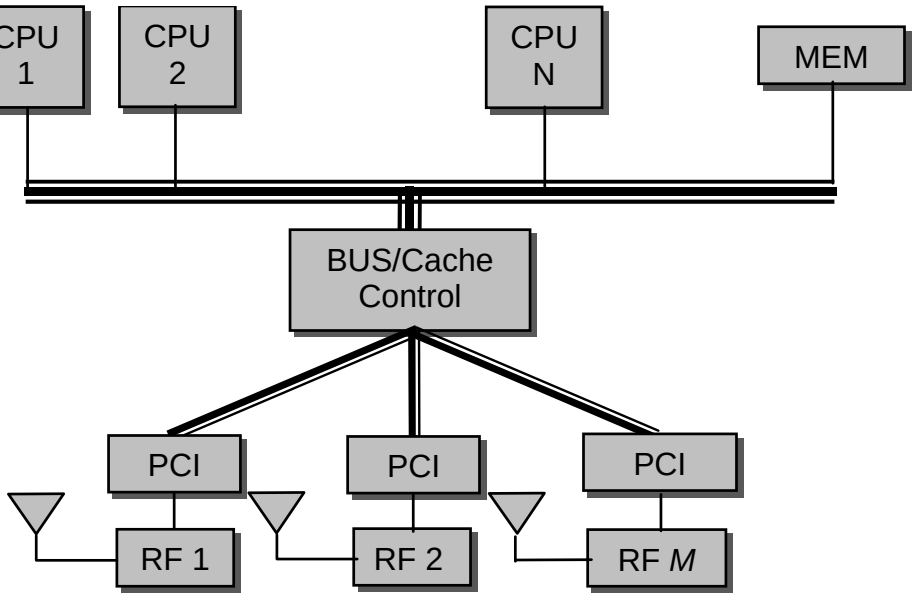


Figure 3: Multi-way SMP Architecture for Radio Gateways

We have chosen the generic PCI bus because it has become the *de facto* standard for data acquisition and comes in many shapes and sizes (e.g. PMC, CARDBUS) for different host environments. Other software-radio testbeds have followed a similar approach below[9][10]. For high-speed data acquisition, PCI64 (66 MHz/64-bit, 528 Mbytes/s peak) or PCI-X (100MHz/64-bit, 800 Mbytes/s peak) may be used instead of low-speed PCI32 (33 MHz/32-bit, 132 Mbytes/s peak). Acquisition systems can be readily designed using the variety of PCI cores (LogiCore PCI) for Xilinx Virtex series FPGAs (see www.xilinx.com). A prototype for this type of data acquisition system was described in [1].

The MT is a normal Linux machine (e.g. laptop) and may be connected to a second machine via ethernet for running special applications. This can be to run Windows applications for instance, or to show applications on PDAs.

The acquisition system for the MT currently uses a PCI bus furnished by a docking station. New acquisition cards are currently being developed using CARDBUS technology[1] that will fit in the laptop's PCMCIA slot and provide low-speed PCI-bus throughput.

2.2 Access Stratum Software Architecture

The elements of the radio interface protocol are shown in Figure 7, where it is seen that the radio interface is layered into three protocol layers:

- The physical layer (L1);
- The data link layer (L2);
- The network layer (L3).

L1 is responsible for

- Control/configuration of the hardware elements (Data Acquisition
- Basic signal processing (Layer 1L)
- Channel coding/decoding and multiplexing/demultiplexing (Layer 1H)

and operates under real-time constraints. We have split L1 into two sublayers, L1L and L1H. L1L is responsible for slot based signal processing (e.g. matched filtering, channel estimation, etc.) while L1H is responsible for frame-based signal processing (e.g. interleaving, error coding/decoding). L1H receives data from the upper-layers at most once per signaling frame (10ms) in blocks known as *transport channels*. These are further sub-divided into *transport channel blocks*. L1H multiplexes and codes these blocks to forms what are known as *physical channels*. Physical channels are raw data mapped onto specific radio channels which, as we will see shortly, are indexed by timeslots and channelization codes.

At the lowest level, the physical layer software interfaces to the hardware data acquisition system and is responsible for configuring DMA transfers of the incoming/outgoing sample streams. Interfaces for controlling the RF functionality (antenna switch, amplifier gains, oscillator frequencies, etc.) are also implemented.

L2 is responsible for

- Medium-access protocols (MAC) (dynamic channel allocation, bandwidth optimization)
- Radio Link layer (RLC) algorithms (retransmission protocols, segmentation, ciphering)

and also operates under real-time constraints, although somewhat less stringent.

L3 implements the signaling protocols that control the radio resource (e.g. user authentication and connection, QoS mapping, etc.). These are known as control-plane or *C-plane signaling* protocols. Furthermore, L3 provides the data interface to the IP backbone network, for user data or *U-plane information*, in the form of a standard Linux networking device (see Section 1). Few aspects of L3 operate under hard real-time constraints.

These software-only implementations of Layers 1,2,3 are developed under the real-time micro-kernel extension to Linux known as RTLinux (see www.rtlinux.org). RTLinux architecture is

[1] CARDBUS is a PCI signalling standard for laptop computers (see www.pc-card.com)

briefly discussed in section 3.2. All signal processing routines make use of the native Intel Pentium DSP instructions found in the various *single-instruction multiple-data* (SIMD) units (MMX, SSE1, SSE2) so that real-time performance can be achieved. This is described in section 3.3 followed by Layer 1 processing is treated in section 5.3.

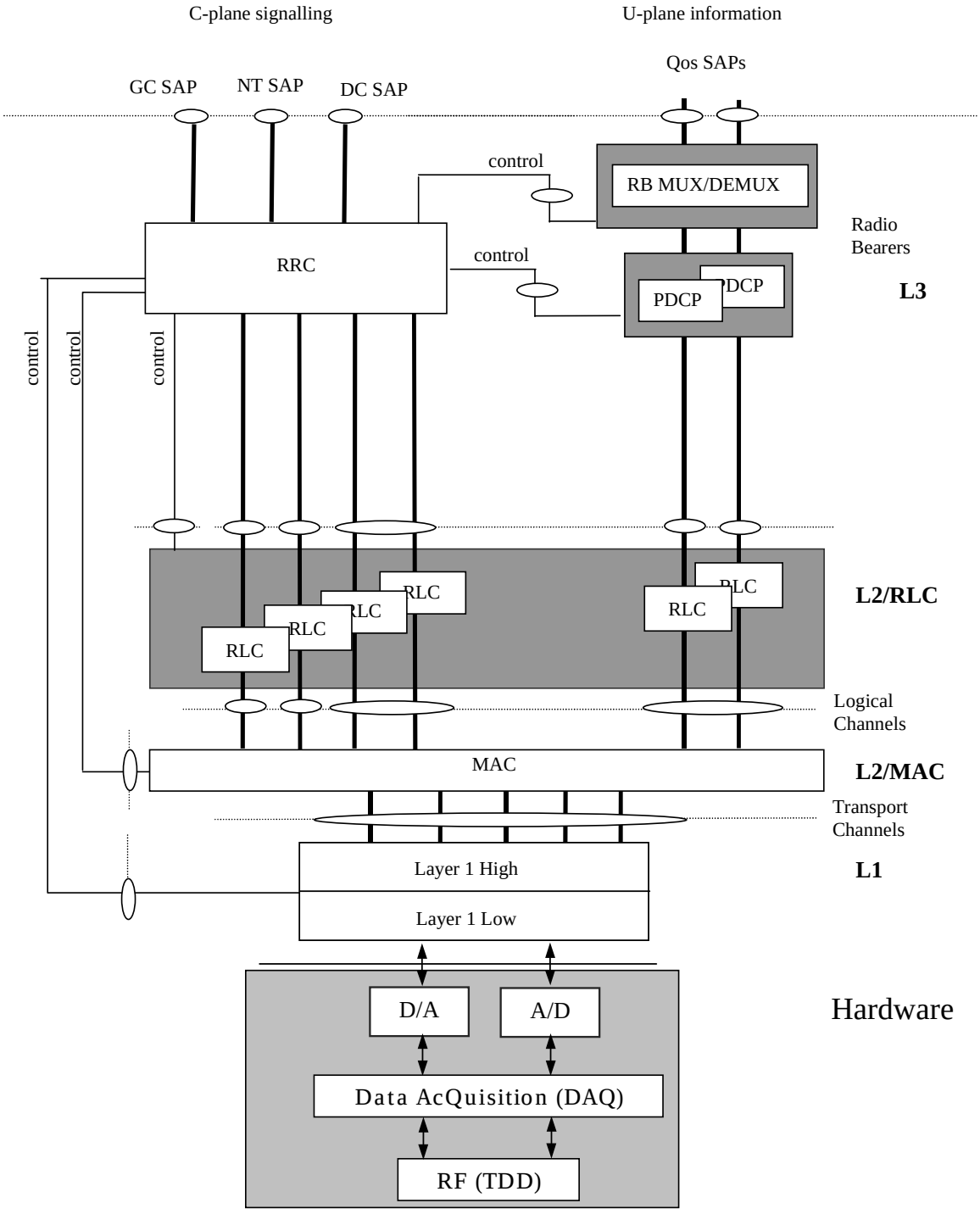


Figure 4: Access Stratum Layers

2.3 RTLinux

RTLinux, an extension to the Linux Operating System (see www.linux.org), supports real-time interrupt handlers and real-time periodic tasks with interrupt latencies and scheduling jitter close to hardware limits. Worst case interrupt latency on a modest PC is under 15 microseconds from the moment the hardware interrupt is asserted. Better hardware configurations produce better timings.

Real-time tasks in RTLinux can communicate with Linux processes either via shared memory regions or a FIFO interface. Thus, real-time applications can make use of all the powerful, non-real-time services of Linux, including:

- Networking
- Graphics
- Windowing systems
- Linux device drivers
- Standard POSIX functionality

2.3.1 Hard Real-Time Processing

The basic idea behind RTLinux is that Linux code that enables/disables interrupts is replaced with code which enables/disables *soft interrupts*. Hard interrupts are now handled by the real-time micro-kernel. If Linux is supposed to handle the interrupt it is passed on to the Linux kernel which services it as a soft interrupt. Due to this separation, the Linux kernel cannot disable interrupts to real-time threads. The result is that Linux drivers work as normal (provided they were written properly) and do not influence real-time threads. This allows for special real-time tasks to be added to the operating system with guaranteed performance without affecting (except for execution speed) normal Linux tasks. The basic structure of the RTLinux extension is shown in Figure 5.

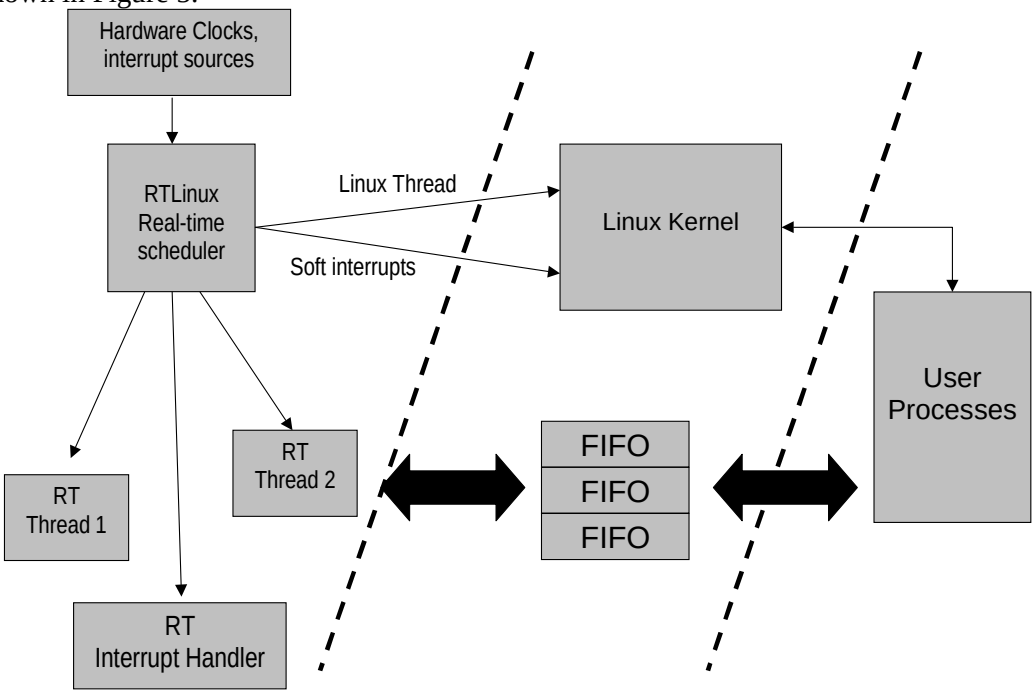


Figure 5: RTLinux Micro-Kernel

2.3.2 POSIX-like API

Although not strictly POSIX compliant, RTLinux strives at a POSIX-like interface. It provides

- a real-time thread library (pthreads)
- real-time schedulers
- real-time external interrupt handling
- SMP (symmetric multi-processing) functions

The SMP functions allow the programmer to pin threads (and in current versions interrupt handlers as well) to a specific CPU. This allows for partitioning of tasks in a multiprocessor environment, which is particularly, appealing for DSP applications.

2.4 Intel SIMD Programming

Our implementation makes use of the native Intel SIMD fixed-point instructions (MMX,SSE) for obtaining maximum processor efficiency on Pentium 3 processors. Our code is written for the GNU gcc C compiler, with assembly intrinsics for MMX/SSE instructions. Future upgrades will include optimized code for the new Pentium 4 and Itanium architectures (www.intel.com) which have many new SIMD instructions for the 128-bit data-path.

Routines typically make use of

- MMX packed 16-bit arithmetic (multiply, add, multiply+add)
- loop unrolling
- software pipelining

By making use of these techniques, a simple FIR filter with 16-bit arithmetic can achieve a computational efficiency of 8/3 multiply/adds per clock cycle. See the Intel whitepapers (www.intel.com) for example code employing these techniques. Sample code for an upsampling FIR filter used in our transmitter is shown in Figure 6. This code implements the following filter:

$$s[n] = \sum_{i=0}^{11} c[n-i] h_{n \bmod 4}[i]$$

and makes use of various parallel 16-bit operations instructions (`movq, pmulhw, paddsw`). The inner loop on the filter taps is completely unrolled. Some execution times for typical algorithms used in 3GPP systems will be given in Section 0

```
void Do_Tx_Flt(mmx_t *c, mmx_t *s,mmx_t *h)
{
 int i;

 for (i=0;i<SLOT_LENGTH;i++)
 {

  movq_m2r(h[0],mm2);
  pxor_r2r(mm0,mm0);
  movq_m2r(c[i],mm3);


  movq_m2r(h[1],mm4);
  pmulhw_r2r(mm2,mm3);

  movq_m2r(c[i-1],mm5);
  paddsw_r2r(mm3,mm0);


  movq_m2r(h[2],mm2);
  pmulhw_r2r(mm4,mm5);

  movq_m2r(c[i-2],mm3);
  paddsw_r2r(mm5,mm0);

// … … … …

  movq_m2r(h_mmx[11],mm4);
  pmulhw_r2r(mm2,mm3);

  movq_m2r(c[i-11],mm5);
  paddsw_r2r(mm3,mm0);

  pmulhw_r2r(mm4,mm5);
  paddsw_r2r(mm5,mm0);
  movq_r2m(mm0,s[i]);
 }
}
```

Figure 6: Example of Optimized Code

3. Layer 1 Software Architecture

3.1 3GPP TDD 3.84 Mchips/s Physical Layer

Here we give a very brief summary of the 3GPP TDD 3.84 Mchip/s physical layer. Detailed information can be found in [2,3,4,5,6,7]. 3GPP TDD 3.84 Mchip/s is a time-slotted CDMA system. Data frames last 10ms and are divided into 15 times slots of equal time duration (667 μs) as shown in Figure 9.

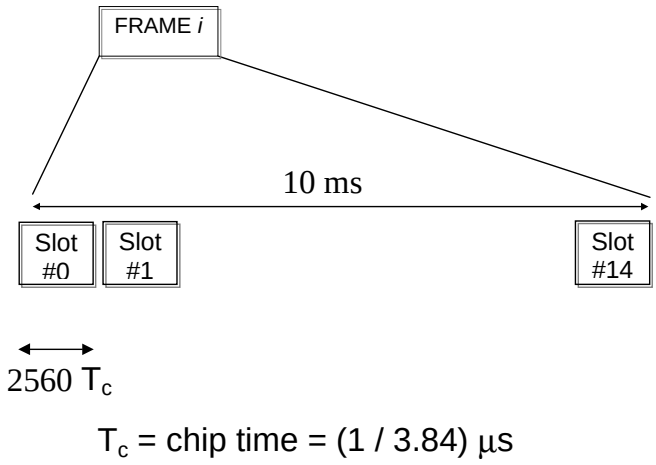


Figure 7: Frame Structure

Slots can be dynamically allocated for either uplink or downlink transmission which allows for the possibility of asymmetric resource allocation for the two traffic directions.

The basic transmission entity in a slot is a burst. A burst is comprised of

- physical channels (dedicated data, control data, primary/secondary synchronization, RG synch)
- a midamble
- a guard-interval
- Transport Format control bits (TFCI)
- Power Control bits (TPC)

Bursts are further subdivided into 2560 QPSK symbols (chips). The basic burst types are shown in Figure 8.

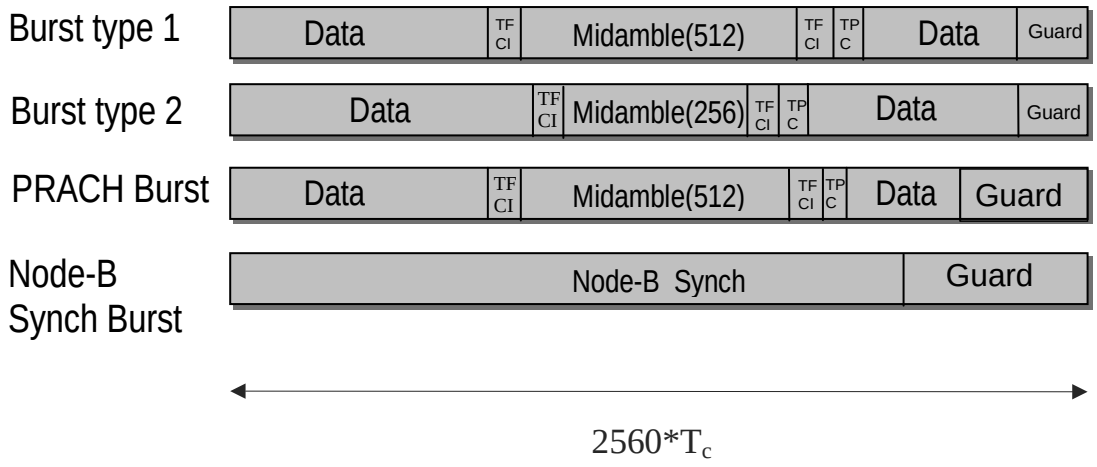


Figure 8: 3GPP TDD 3.84 Mchip/s Burst Structures

Physical channels are indexed by their slot number orthogonal channelization (spreading) code. Spreading is used to allow several users to share a particular timeslot and as a means of achieving multi-level coding. On the uplink the maximum number of users sharing a particular timeslot is 8, and users can use at most 2 channelization codes in parallel. On the downlink up to 16 parallel channelization codes can be used. Spreading is achieved through the use of Orthogonal Variable Spreading Factor codes which are based on Hadamard sequences. On the

downlink the allowed spreading factors are 16 and 1, whereas on the uplink factors of 1,2,4,8,16 are permitted.

3.2 Implemented Phyiscal Channels

Physical channels are indexed by their slot number and orthogonal channelization (spreading) code. Spreading is used to allow several users to share a particular timeslot and as a means of achieving multi-level coding.

On the uplink (UL) the maximum number of users sharing a particular timeslot is 8, and users can use at most 2 channelization codes in parallel. On the downlink (DL) up to 16 parallel channelization codes can be used. Spreading is achieved through the use of Orthogonal Variable Spreading Factor (OVSF) codes which are based on Hadamard sequences.

3.2.1 Channelization Codes

Channelization codes are simply an organization of Hadamard codes of varying lengths which allow users (or data streams of one user) of potentially different data rates to be mutually orthogonal. These are known as OVSF codes. Figure 3 shows the OVSF code tree for spreading factors 1,2, and 4. Each level of the tree contains the 2^{Q-1} rows of a Hadamard matrix. In addition to being orthogonal to all sequence at the same level, the organization of the tree allows any sequence to be orthogonal to any other sequence which is not a direct ancestor

or descendant in the tree.

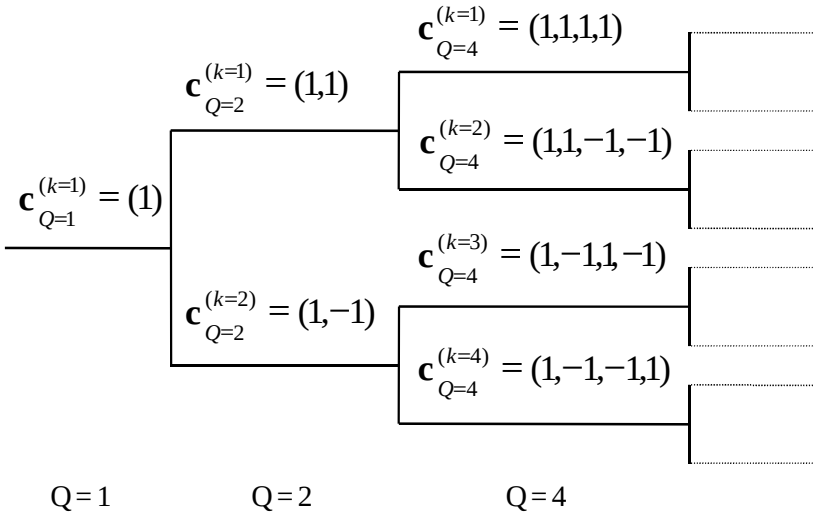


Figure 9: OVSF Code Tree

3.2.2 Scrambling codes

Scrambling is used to reduce interference from signals originating in neighbouring cells. Each cell is assigned a scrambling sequence of length 16 chips. This is a major difference from other CDMA systems such as UMTS/FDD and IS-95 which use very long scrambling codes. Because of this short scrambling sequence, advanced receiver architectures (i.e. equalizers, multiuser receivers) can be implemented at reasonable computational cost.

3.2.3 Modulation Pulse-Shaping

QPSK signaling is used in UMTS TDD. Pulse-shaping is achieve through the use of a root-raised cosine filter with a spectral roll-off factor of .22. The resulting channel bandwidth is 5 MHz.

3.2.4 Dedicated Physical Channels (DCH)

Dedicated physical channels are used to transmit user traffic. Any combination of burst types, channelization codes and TFCI formats are allowed. On the downlink the allowed spreading factors are 16 and 1, whereas on the uplink factors of 1,2,4,8,16 are possible.

3.2.5 Primary common control physical channel (P-CCPCH)

The BCH or *broadcast channel* is mapped onto the Primary Common Control Physical Channel (P-CCPCH). The position (time slot / code) of the P-CCPCH is known from the Physical Synchronisation Channel (PSCH), see subclause . This channel always uses a spreading factor of 16 and channelization code 0. Burst type 1 is used without TFCI.

3.2.6 Secondary common control physical channel (S-CCPCH)

PCH (Paging Channel) and FACH (Forward Access Channel) are mapped onto one or more secondary common control physical channels (S-CCPCH). In this way the capacity of PCH and FACH can be adapted to the different requirements. The S-CCPCH is implemented for control channels.

3.2.7 The physical random access channel (PRACH)

The RACH (Random-access Channel) is mapped onto one uplink physical random access channel (PRACH). This channel is used by the mobile station during the connection phase with the basestation and for passing control information once connected. Because of its importance, the PRACH is implemented for control information and short user data packets.

3.2.8 The synchronisation channel (SCH)

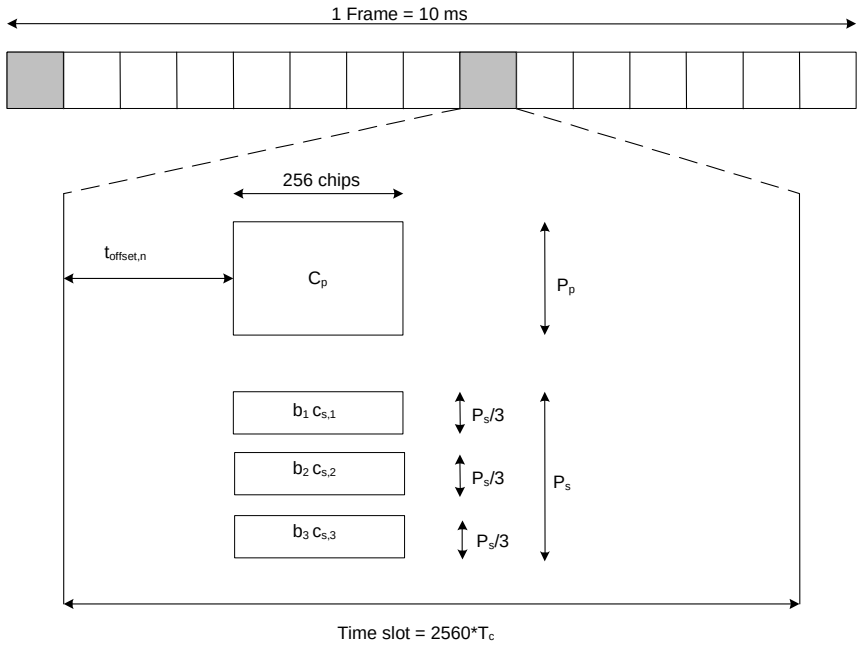
The SCH is used for two purposes in UMTS/TDD. First to acquire initial frame synchronization (using the primary synchronization code) and second to derive the code group of a cell (using the secondary synchronization code). In order not to limit the uplink/downlink asymmetry the SCH is mapped on one or two downlink slots per frame only.

There are two cases of SCH and P-CCPCH allocation:

Case 1) SCH and P-CCPCH allocated in TS#k, k=0….14

Case 2) SCH allocated in two TS: TS#k and TS#k+8, k=0…6; P-CCPCH allocated in TS#k.

The position of SCH (value of k) in frame can change on a long term basis in both cases. Due to this SCH scheme, the position of P-CCPCH is known from the SCH. Figure 14 is an example for transmission of SCH, k=0, of Case 2.



$b_i \in \{\pm 1, \pm j\}$, $C_{s,i} \in \{C_0, C_1, C_3, C_4, C_5, C_6, C_8, C_{10}, C_{12}, C_{13}, C_{14}, C_{15}\}$, i=1,2,3;

Figure 10: SCH signaling

As depicted in figure 4, the SCH consists of a superposition of a primary and three secondary code sequences each 256 chips long. Typically a mobile station will detect several candidate base stations using the primary synchronization sequence and select the one with the strongest signal. Since base stations are required to be synchronized, the time-offset of the SCH is used to ensure that the SCH from neighbouring cells do not overlap in time, thus allowing the mobile station to distinguish them. The time offset $t_{offset,n}$ is one of 32 values, depending on the code group of the cell. In addition to determining the code group (and thus t_{offset}) the secondary synchronization sequences allow the mobile station to determine the frame number modulo-2. This is needed to correctly pass the demodulated P-CCPCH data to the channel decoding functions, since the BCH is interleaved across 2 frames.

3.3 Layer 1L software architecture

Layer 1 DSP routines operate on buffers written to and read from the external PCI acquisition system. The two DMA memory buffers are organized as shown in Figure 10 and contain 1UMTS frame's (10ms) worth of samples.

As shown in Figure 4, layer 1 is subdivided into 2 parts, L1L and L1H. L1L is responsible for modulation and L1H for channel coding and multiplexing. The basic transmitter functions of L1L are shown in Figure 12. The higher layers (include L1H) provide data to be modulated to L1L in a form that can be described by the following data structures:

```
// uX == unsigned X-bit integers
// sX == signed X-bit integers

typedef struct L1C_PhyChan {
  u32 MobileFlag;      /* indicates if channel must be looked at by a mobile
*/
  u16 Data_amp;        /* channel amplitude */
  u16 Mid_amp;         /* midamble amplitude for each slot and channel */
  u8  BurstFormat;     /* index of the burst format */
  u8  CCTrCh;          /* CCTrCh (on reception) */
  u8  Sequence_index;  /* the channelization sequence index ( < SF) */
  u8  Mid_rot_index;       /* midamble index for each channel and slot for
rotating,from 0 to 2 */
  u8  Midamble_index;
  u16 TFCI;
  u8  Channel_Status;/* On transmission, or for L1H on reception */
  DATA_PTR DataPtr;    /* Pointers to a common pool of buffers */
} L1C_PhyChan

typedef union{
  s8  *Rx;   /* this has to be signed */
  u32 *Tx;
} DATA_PTR;

/* Physical burst formats, from 25-221
   Entries 0 to 89 are exactly the uplink formats from Table 5b.
   Entries 90 to 109 are the downlink formats from Table 5a
   with added TPC fields (always 0) and GP (always 96). */
typedef struct L1C_BF {
  u8 SF;
  u8 ML;    /* Midamble length, divided by 256 */
  u8 GP;    /* Guard period */
  u8 NumTFCIBits;
  u8 NumTPCBits;
  u16 NumData;  /* Total number of data bits */
  u16 NumData1; /* Num of data bits in first half */
} L1C_BF;

typedef struct L1C_SHARE {
  u8 N_channels[L1C_SLOTS_FRAME];       /* number of channels for each slot */
  u16 Synch_prim_amp[L1C_SLOTS_FRAME];   /* primary synch amplitude */
  struct L1C_PhyChan PhyChan[L1C_SLOTS_FRAME][L1C_MAX_CHANNELS];
} L1C_SHARE ;
```

Figure 11: Data Structures for Describing 3GPP/TDD Physical Channels

These structures define the interface between L1L and L1H, and exactly reproduce the 3GPP standards described in [3,4,5]. Since they are common to both L1L and L1H, they have been given the prefix L1C (for common). The structure `L1C_BF` describes the format of the burst in terms of the size of the different elements it comprises (as shown in Figure 8). This is taken from the 3GPP tables 5a and 5b from [3]. Each physical channel is described by L1C_PhyChan and contains information such as amplitudes, burst formats, channelization indices, Transport format indicators (TFCI), pointers to raw data (DataPtr.Tx) and matched filter output (DataPtr.Rx), etc. The entire frame is described in L1C_SHARE and contains the number of channels for each slot, the amplitude of the primary synchronization signal in a slot, and the complete physical channel descriptions.

L1L interprets the L1C data structures to generate the sample streams as shown in Figure 12. The raw data sequences are first spread (including scrambling) and then modulated using a root-raised cosine pulse-shaping filter. For passband signals, 4 samples/symbol are used and upconversion to $f_s / 4$ is performed. For complex baseband equivalent signals, the raw symbol sequences are upsampled by a factor of 2 sample/symbol.

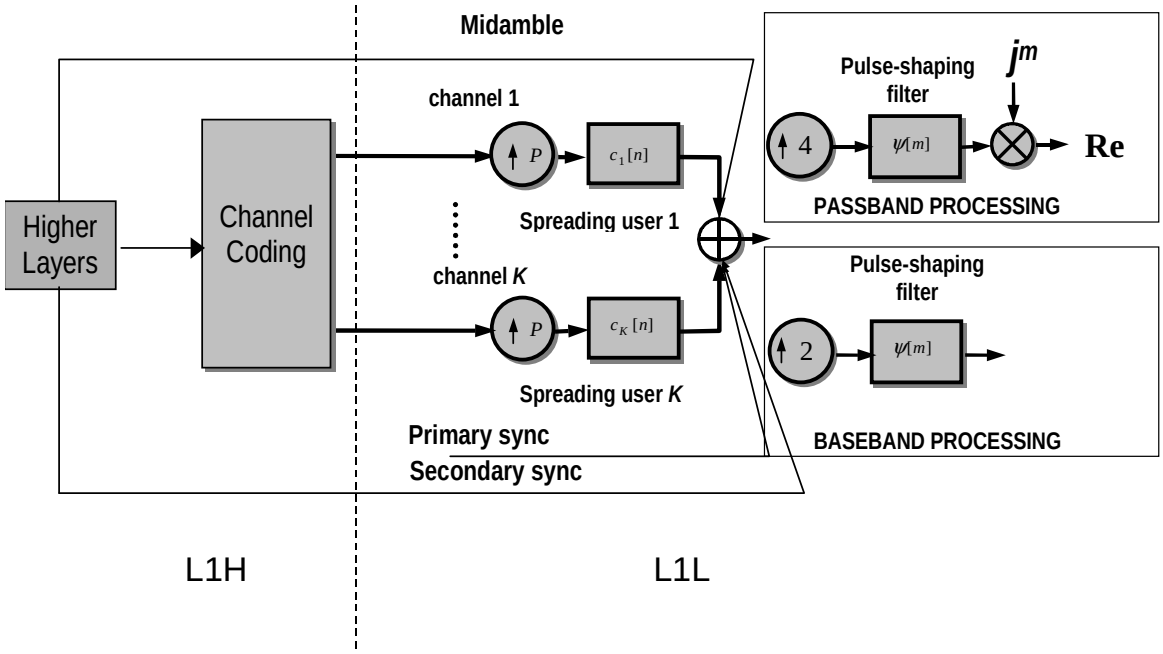


Figure 12: Layer 1L Transmitter

The basic receiver structure for both RGs and MTs is shown in Figure 10. The receiver for a MT operates in 2 modes. The first is the initial acquisition of the RGs synchronization signal. This operation comprises correlation and energy detection. Once synchronized, frame/slot timing is acquired and demodulation of the different parallel channels can be accomplished.

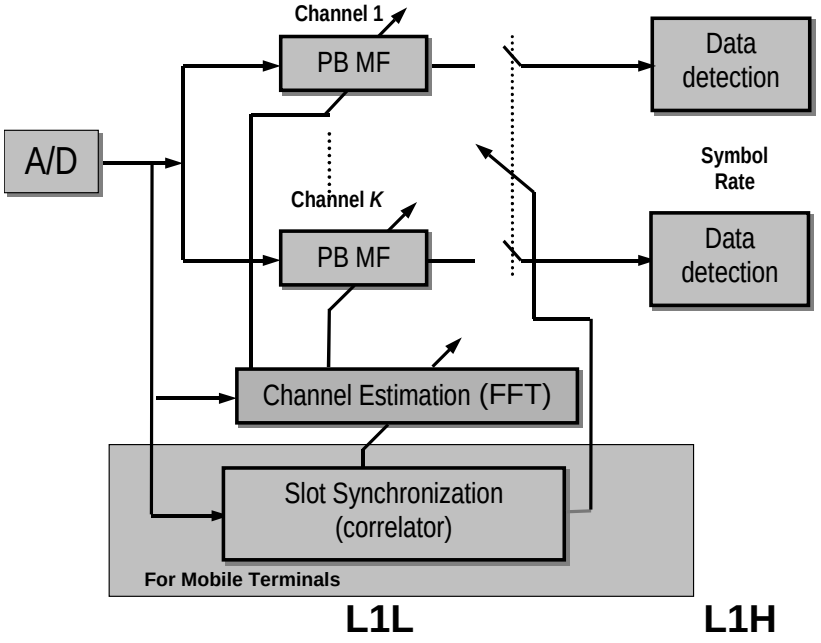


Figure 13: Layer 1L Receiver

For each slot, the impulse response of the channel is estimated using a frequency-domain least-squares approach based on the FFT as described in [1]. This is possible due to the cyclic structure of the 3GPP midamble sequences. Once estimated, the channel impulse is convolved with the channelization sequence of the different channels to obtain a bank of matched filters.

These filters are then used to derive the sufficient statistics for each data symbol on each channel using 8-bit soft-decisions. These are passed to the higher layers to perform de-interleaving and soft-decision error decoding. The channel estimate is also used to adjust slot timing due to frequency offset between the RG and MT sampling clocks.

The receiver for an RG is somewhat simpler since initial synchronization is not needed. At a later stage, a synchronization (and timing adjustment) algorithm will be required to synchronize several RGs. This will be accomplished using the 3GPP Node-b synchronization burst. The main difference is that the RG must demodulate more parallel streams and perform joint channel estimation of up to 8 users. Again this is made possible due to the cyclic properties of the midamble sequences via the FFT.

3.4 Layer 1H overview

Figure 14 illustrates the overall concept of transport-channel coding and multiplexing characteristic of 3GPP systems. Data arrives to the coding/multiplexing unit in form of transport block sets, once every transmission time interval. The transmission time interval is transport-channel specific from the set {10 ms, 20 ms, 40 ms, 80 ms}. The following coding/multiplexing steps can be identified:

- Addition of CRC information to each transport block
- Transport Block concatenation / Code block segmentation
- channel coding (convolutional, turbo, uncoded)
- radio frame size equalization
- interleaving
- radio frame segmentation
- rate matching
- multiplexing of transport channels
- bit scrambling
- physical channel segmentation
- mapping to physical channels

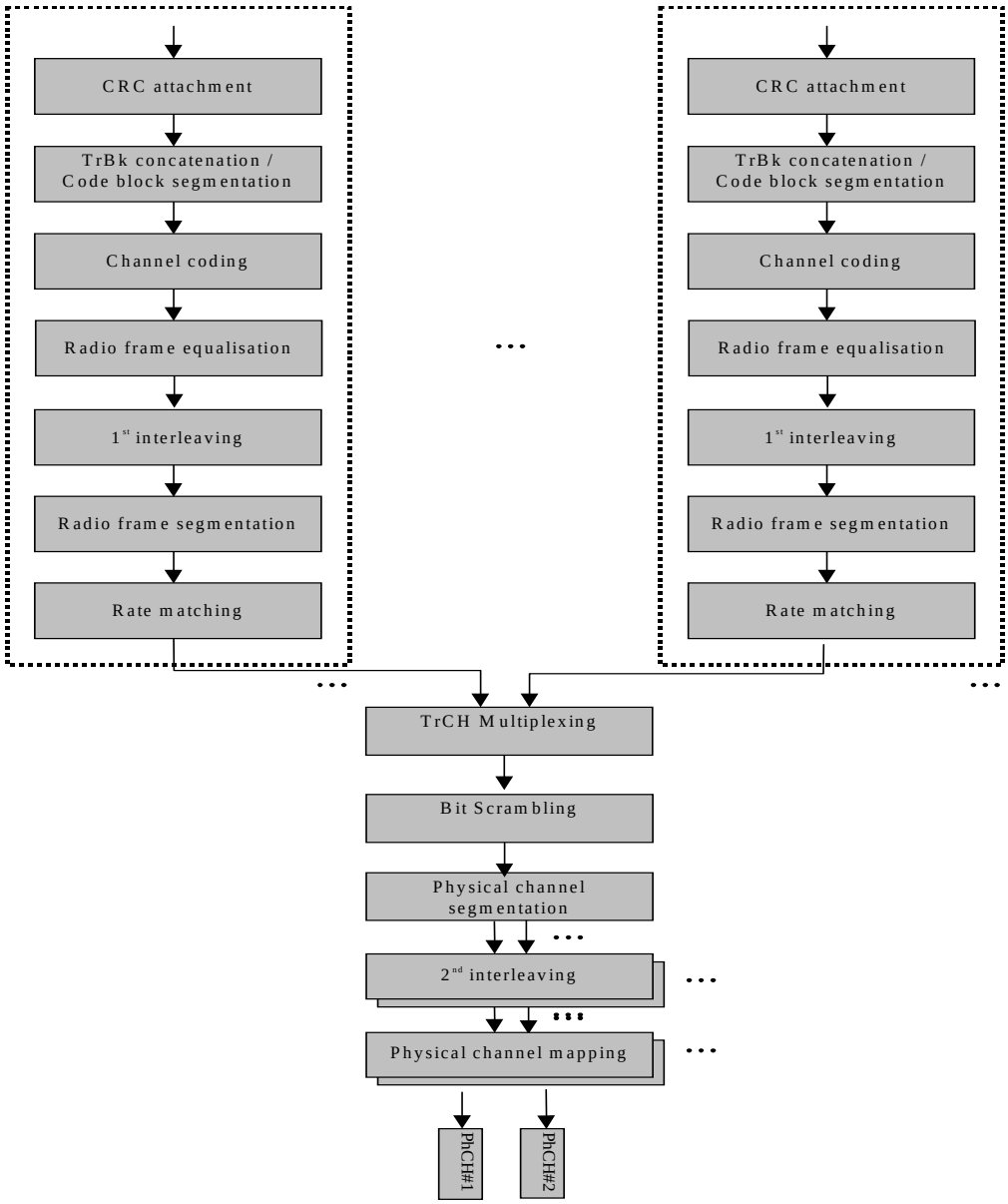


Figure 14: 3GPP Transport channel multiplexing structure

The main computational burden related to L1H lies in the channel decoder for the receiver. Three of the channel coding methods require either a soft-decision Viterbi decoder on a 256-state trellis (convolutional code), or a soft-in soft-out 8-state iterative decoder (turbo decoder). We have currently only implemented the convolutional coding options of 3GPP. Aside from error decoding, the remainder of L1H operations in both transmit and receive are insignificant with respect to computational complexity.

3.5 Real-time Thread Scheduling

As shown in Figure 1 Figure 2 the 3GPP TDD radio interface is implemented using real-time POSIX threads. Scheduling of these threads is synchronized to the hardware via signals coming from the acquisition system with an accuracy of 66.67µs. A typical L1L thread would have the form shown in Figure 8.

```
void *L1L_thread(void *param) {

    // Do some initilization

    do {   // loop until we get a signal to exit

      wait_for_scheduling();  // wait to be woken
                              // up by the scheduler

        DO_L1L_DSP(slot_count);     // do L1L DSP

        if (slot_count == LAST_SLOT_IN_FRAME)
          Wake_up_L1H();    // wake up frame-based thread

             // now TEST for error condition

        EXIT = get_exit();

        // compute the wakeup time of the next thread
        // using information from timer and acquisition hardware

        if (EXIT == 0) {
          start_time = get_time() +  OFFSET_COMPUTED_FROM_HARDWARE;

       // schedule next dsp_thread_j at start_time

          schedule(L1L_thread,start_time);

        slot_count++;
      }
    }
    while (EXIT == 0);

}
```

Figure 15: L1L thread

The L1L thread is scheduled to execute every slot (i.e. 666 μs) and performs either or both of the transmitter for the next slot and receiver for the previous slot. This scheduling is adjusted to account for the drift between the hardware clock and the system clock using timing signals provided by the hardware. At the end of each frame, the L1L_thread schedules the execution of the frame-based processing (L1H, L2, L3). On transmit the following frame is prepared and on receive the previous frame is processed. The overall thread scheduling scenario is shown in Figure 16 where the different thread priorities can be seen. The remaining CPU time is given to the non real-time mode running Linux user and kernel-space processes including the IP networking subsystem.

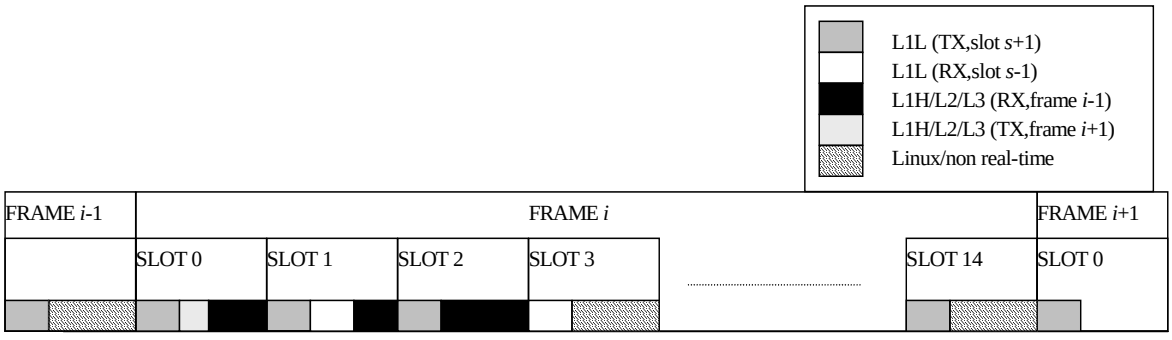


Figure 16: Real-time Thread Scheduling

3.6 Typical Execution Times

In table we provide an indication of the execution time of the various signal-processing blocks previously described. The reference machine is a single-processor 800 MHz Pentium 3 running

RTLinux 3.0 under RedHat Linux 7.1. We note that on the transmission end, the signal processing time per slot requires less than half the slot duration. Aside from Viterbi decoding in L1H, the same is true on reception. The CPU load required by Viterbi decoding depends heavily on the data rate. In the example shown in the table, we sustain a rate of 1 Mbit/s for blocks of length 1000 bits on a single processor, so that coded data rates on the order of 400 kbit/s should be feasible with the remaining processing on a modest PC.

Operation	Sub-operations	Time (μs)
Transmission	Spreading/Scrambling Pulse-shape filtering	Insignificant 200 per slot
Reception	Primary Synchronization	350-400 per slot
Reception	Channel estimation (FFT, 3 channels) Matched Filtering (spreading 16) Matched Filtering (no spreading) Viterbi decoding (1000 output bits)	80 per slot 20 per slot 300 per slot 1000

Table 1: Typical Execution Times

4. Implementation of Layers 2 and 3

As shown in Figure 4, Layer 2 is subdivided into the following layers: Medium Access Control (MAC), Radio Link Control (RLC), and Packet Data Convergence Protocol (PDCP). Layer 3 and RLC are divided into Control (C-) and User (U-) planes. PDCP exists in the U-plane only.

In the C-plane, Layer 3 is partitioned into sublayers where the lowest sublayer, denoted as Radio Resource Control (RRC), interfaces with layer 2 and terminates in the backbone IP network; it provides the AS Services to higher layers. The higher layer signaling such as Mobility Management (MM) and Call Control (CC) is assumed to belong to the NAS, and therefore not in the scope of this paper.

Each RLC/MAC/PDCP block in Figure 4 represents an instance of the respective protocol. Service Access Points (SAP) for peer-to-peer communication are marked with circles at the interface between sublayers. The SAP between MAC and the physical layer provides the transport channels discussed previously. The SAPs between RLC and the MAC sublayer provide the logical channels. The RLC layer provides three types of SAPs, one for each RLC operation mode (unacknowledged-UM, acknowledged-AM, and transparent-TM). PDCP is accessed by PDCP SAP. The service provided by L2 is referred to as the radio bearer. The C-plane radio bearers, which are provided by RLC to RRC, are denoted as signaling radio bearers. In the C-plane, the interface between 'Duplication avoidance' and higher L3 sublayers (CC, MM) is defined by the General Control (GC), Notification (Nt) and Dedicated Control (DC) SAPs.

Also shown in the figure are connections between RRC and MAC as well as RRC and L1 providing local inter-layer control services. An equivalent control interface exists between RRC and the RLC sublayer, between RRC and the PDCP sublayer. These interfaces allow the RRC to control the configuration of the lower layers. For this purpose separate Control SAPs are defined between RRC and each lower layer (PDCP, RLC, MAC, and L1).

The RLC sublayer provides ARQ functionality closely coupled with the radio transmission technique used. There is no difference between RLC instances in C and U planes. There are primarily two kinds of signaling messages transported over the radio interface - RRC generated signaling messages and NAS messages generated in the higher layers. On establishment of the signaling connection between the peer RRC entities three or four UM/AM signaling radio bearers may be set up. Two of these bearers are set up for transport of RRC generated signaling messages - one for transferring messages through an unacknowledged mode RLC entity and the

other for transferring messages through an acknowledged mode RLC entity. One signaling radio bearer is set up for transferring NAS messages set to "high priority" by the higher layers. An optional signaling radio bearer may be set up for transferring NAS messages set to "low priority" by the higher layers. Subsequent to the establishment of the signaling connection zero to several TM signaling radio bearers may be set up for transferring RRC signaling messages using transparent mode RLC.

4.1 MAC Services and Functions

This subclause provides an overview on services and functions provided by the MAC sublayer.

4.1.1 MAC Services to upper layers

- **Data transfer**. This service provides unacknowledged transfer of MAC SDUs between peer MAC entities. This service does not provide any data segmentation. Therefore, segmentation/reassembly function should be achieved by upper layer.
- **Reallocation of radio resources and MAC parameters**. This service performs on request of RRC execution of radio resource reallocation and change of MAC parameters, i.e. reconfiguration of MAC functions such as change of identity of UE, change of transport format (combination) sets, change of transport channel type. In TDD mode, in addition, the MAC can handle resource allocation autonomously.
- **Reporting of measurements**. Local measurements such as traffic volume and quality indication are reported to RRC.

4.1.2 Transport Channels

The MAC layer provides an interface to L1 consisting of transport channels. The following transport channels are implemented in the testbed.

BCH (Broadcast Channel)- A downlink channel used for broadcasting basic system information, notably radio channel configuration data.

FACH (Forward Access Channel)- A downlink control channel for AS/NAS signalling (connection establishment, data channel establishment) as well as short user packets and paging.

RACH (Random-Access Channel) – An uplink control channel for AS/NAS signalling (connection establishment, data channel establishment) as well as short user packets.

DCH (Dedicated Channel)- A bi-directional channel for either dedicated traffic or control information.

4.1.3 Logical Channels

The MAC layer provides data transfer services on logical channels. A set of logical channel types is defined for different kinds of data transfer services as offered by MAC. Each logical channel type is defined by what type of information is transferred.
A general classification of logical channels is into two groups:
- Control Channels (for the transfer of C-plane information);
- Traffic Channels (for the transfer of U-plane information).

The configuration of logical channel types is depicted in Figure 17.

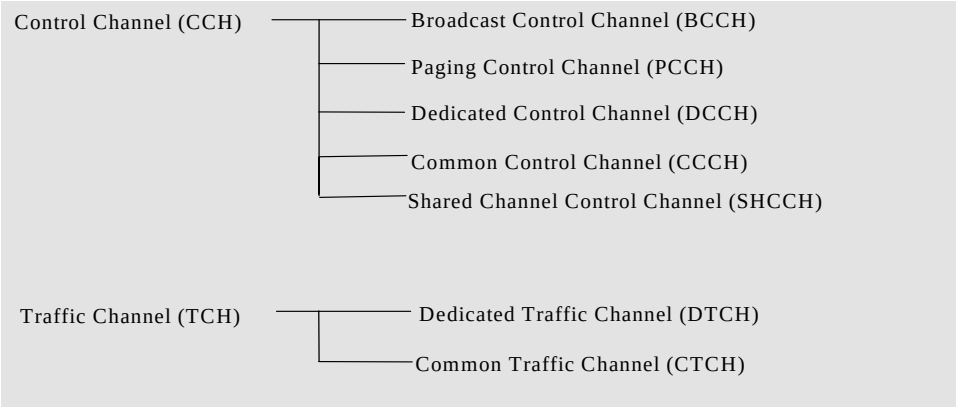


Figure 17: Logical channel structure

4.1.4 Control Channels

Control channels are used for transfer of C-plane information only. The following control channels are implemented.

- **Broadcast Control Channel (BCCH)**

A downlink channel for broadcasting system control information.

- **Common Control Channel (CCCH)**

Bi-directional channel for transmitting control information between network and UEs. This channel is commonly used by the UEs having no RRC connection with the network and by the UEs using common transport channels when accessing a new cell after cell reselection.

- **Dedicated Control Channel (DCCH)**

A point-to-point bi-directional channel that transmits dedicated control information between a UE and the network. This channel is established through RRC connection setup procedure.

4.1.5 Traffic Channels

Traffic channels are used for the transfer of U-plane information only. The following U-plane channels are implemented.

- **Dedicated Traffic Channel (DTCH)**

A Dedicated Traffic Channel (DTCH) is a point-to-point channel, dedicated to one UE, for the transfer of user information. A DTCH can exist in both uplink and downlink.

- **Common Traffic Channel (CTCH)**

A point-to-multipoint unidirectional channel for transfer of dedicated user information for all or a group of specified UEs.

4.1.6 Mapping between logical channels and transport channels

The logical/transport channel mappings as seen from the UE and Network sides are shown in Figure 18 and Figure 19 respectively.

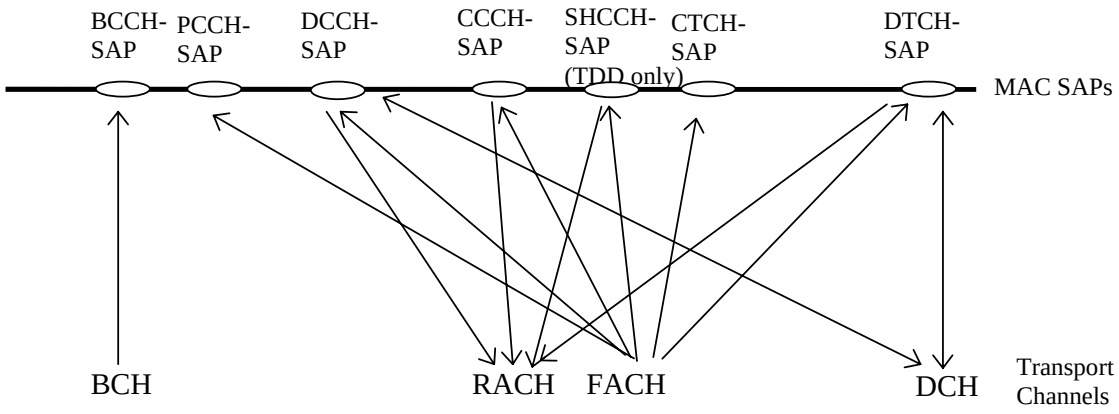


Figure 18: Logical channels mapped onto transport channels, seen from the UE side

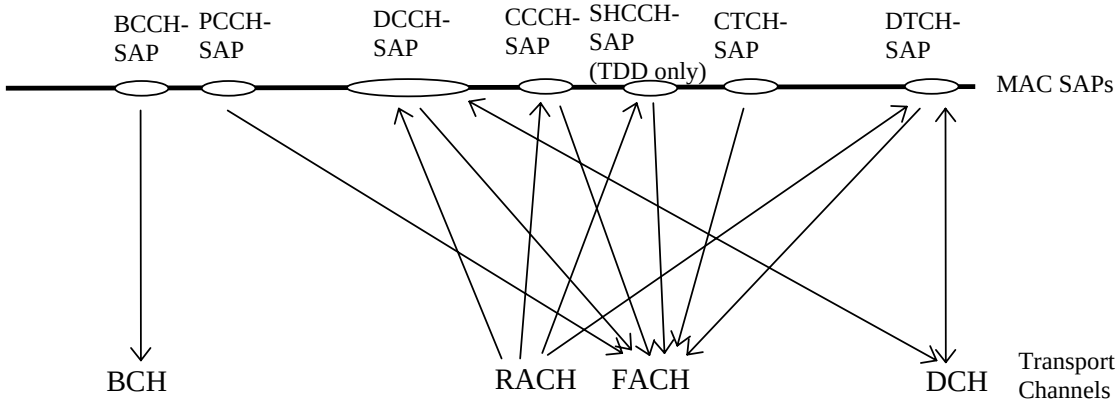


Figure 19: Logical channels mapped onto transport channels, seen from the UTRAN side

4.1.7 MAC functions

The functions of MAC include:

- **Mapping between logical channels and transport channels.** The MAC is responsible for mapping of logical channel(s) onto the appropriate transport channel(s).
- **Selection of appropriate Transport Format for each Transport Channel depending on instantaneous source rate**. Given the Transport Format Combination Set assigned by RRC, MAC selects the appropriate transport format within an assigned transport format set for each active transport channel depending on source rate**.** The control of transport formats ensures efficient use of transport channels.
- **Priority handling between data flows of one UE**. When selecting between the Transport Format Combinations in the given Transport Format Combination Set, priorities of the data flows to be mapped onto the corresponding Transport Channels can be taken into account. Priorities are e.g. given by attributes of Radio Bearer services and RLC buffer status. The priority handling is achieved by selecting a Transport Format Combination for which high priority data is mapped onto L1 with a "high bit rate" Transport Format, at the same time letting lower priority data be mapped with a "low bit rate" (could be zero bit rate) Transport Format. Transport format selection may also take into account transmit power indication from Layer 1.
- **Priority handling between UEs by means of dynamic scheduling**. In order to utilize the spectrum resources efficiently for bursty transfer, a dynamic scheduling function may be applied. MAC realizes priority handling on common and shared transport channels. Note that

for dedicated transport channels, the equivalent of the dynamic scheduling function is implicitly included as part of the reconfiguration function of the RRC sublayer.
NOTE: In the TDD mode the data to be transported are represented in terms of sets of resource units.

- **Identification of UEs on common transport channels**. When a particular UE is addressed on a common downlink channel, or when a UE is using the RACH, there is a need for inband identification of the UE. Since the MAC layer handles the access to, and multiplexing onto, the transport channels, the identification functionality is naturally also placed in MAC.
- **Multiplexing/demultiplexing of upper layer PDUs into/from transport blocks delivered to/from the physical layer on common transport channels**. MAC should support service multiplexing for common transport channels, since the physical layer does not support multiplexing of these channels.
- **Multiplexing/demultiplexing of upper layer PDUs into/from transport block sets delivered to/from the physical layer on dedicated transport channels**. The MAC allows service multiplexing for dedicated transport channels. This function can be utilized when several upper layer services (e.g. RLC instances) can be mapped efficiently on the same transport channel. In this case the identification of multiplexing is contained in the MAC protocol control information.
- **Traffic volume measurement**. Measurement of traffic volume on logical channels and reporting to RRC. Based on the reported traffic volume information, RRC performs transport channel switching decisions.
- **Transport Channel type switching.** Execution of the switching between common and dedicated transport channels based on a switching decision derived by RRC**.**
- **Access Service Class selection for RACH transmission**. The RACH resources (i.e. timeslot and channelisation code for TDD) may be divided between different Access Service Classes in order to provide different priorities of RACH usage. Each access service class will also have a set of back-off parameters associated with it; the network may broadcast some or all of which. The MAC function applies the appropriate back off and indicates to the PHY layer the RACH partition associated to a given MAC PDU transfer.

4.2 RLC Services and Functions

4.2.1 Services provided to the upper layer

- **Transparent data transfer**. This service transmits upper layer PDUs without adding any protocol information, possibly including segmentation/reassembly functionality.

Unacknowledged data transfer. This service transmits upper layer PDUs without guaranteeing delivery to the peer entity. The unacknowledged data transfer mode has the following characteristics:

- Detection of erroneous data: The RLC sublayer shall deliver only those SDUs to the receiving upper layer that are free of transmission errors by using the sequence-number check function.
- Immediate delivery: The receiving RLC sublayer entity shall deliver a SDU to the upper layer receiving entity as soon as it arrives at the receiver.

Acknowledged data transfer. This service transmits upper layer PDUs and guarantees delivery to the peer entity. In case RLC is unable to deliver the data correctly, the user of RLC at the transmitting side is notified. For this service, only in-sequence delivery is supported. In many cases upper layer protocol can restore the order of its PDUs. As long as the out-of-sequence properties of the lower layer are known and controlled (i.e. the upper layer protocol will not immediately request retransmission of a missing PDU) allowing out-of-sequence delivery can save memory space in the receiving RLC. But simulations have proved that out of sequence delivery generate a significant overhead in upper layers. The acknowledged data transfer mode has the following characteristics:

- Error-free delivery: Error-free delivery is ensured by means of retransmission. The receiving RLC entity delivers only error-free SDUs to the upper layer.

- Unique delivery: The RLC sublayer shall deliver each SDU only once to the receiving upper layer using duplication detection function.
- In-sequence delivery: RLC sublayer shall provide support for in-order delivery of SDUs, i.e., RLC sublayer should deliver SDUs to the receiving upper layer entity in the same order as the transmitting upper layer entity submits them to the RLC sublayer.

Maintenance of QoS as defined by upper layers. The retransmission protocol shall be configurable by layer 3 to provide different levels of QoS. This can be controlled.

Notification of unrecoverable errors. RLC notifies the upper layer of errors that cannot be resolved by RLC itself by normal exception handling procedures, e.g. by adjusting the maximum number of retransmissions according to delay requirements. There is a single RLC connection per Radio Bearer.

4.2.2 RLC Functions

Segmentation and reassembly. This function performs segmentation/reassembly of variable-length upper layer PDUs into/from smaller RLC PDUs. The RLC PDU size is adjustable to the actual set of transport formats.

Concatenation. If the contents of an RLC SDU cannot be carried by one RLC PDU, the first segment of the next RLC SDU may be put into the RLC PDU in concatenation with the last segment of the previous RLC SDU.

Padding. When concatenation is not applicable and the remaining data to be transmitted does not fill an entire RLC PDU of given size, the remainder of the data field shall be filled with padding bits.

Transfer of user data. This function is used for conveyance of data between users of RLC services. RLC supports acknowledged, unacknowledged and transparent data transfer. QoS setting controls transfer of user data.

Error correction. This function provides error correction by retransmission (e.g. Selective Repeat, Go Back N, or a Stop-and-Wait ARQ) in acknowledged data transfer mode.

In-sequence delivery of upper layer PDUs. This function preserves the order of higher layer PDUs that were submitted for transfer by RLC using the acknowledged data transfer service. If this function is not used, out-of-sequence delivery is provided.

Duplicate Detection. This function detects duplicated received RLC PDUs and ensures that the resultant upper layer PDU is delivered only once to the upper layer.

Flow control. This function allows an RLC receiver to control the rate at which the peer RLC transmitting entity may send information.

Sequence number check. This function is used in unacknowledged mode and guarantees the integrity of reassembled PDUs and provides a mechanism for the detection of corrupted RLC SDUs through checking sequence number in RLC PDUs when they are reassembled into a RLC SDU. A corrupted RLC SDU will be discarded.

Protocol error detection and recovery. This function detects and recovers from errors in the operation of the RLC protocol.

Polling. This function is used when an RLC transmitter requests a status report of an RLC receiver.

Status transmission. An RLC receiver uses this function to transmit status reports to a RLC transmitter in order to inform about which PDUs that have been received and not received.

SDU discard. This function allows an RLC transmitter to discharge RLC SDU from the buffer.

Estimated PDU Counter (EPC) mechanism. This function is used for scheduling the retransmission of status reports in the receiver side.

Suspend/resume function. Suspension and resumption of data transfer.

Stop/continue function. Stop and continue of data transfer.

Re-establishment function. Re-establish an acknowledged or unacknowledged mode RLC entity.

5. Interface between layer 2 and IP

5.1 Architecture of the UMTS bearers

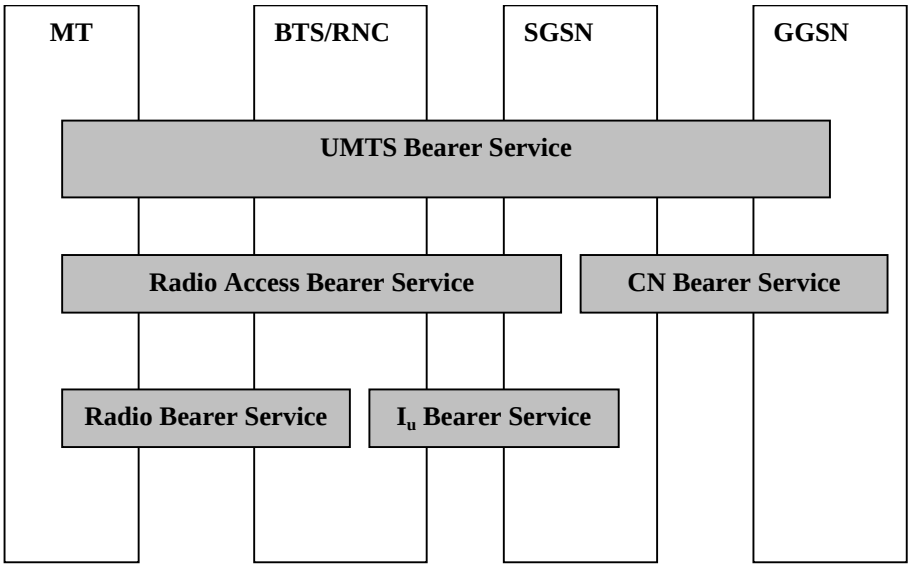


Figure 20: 3GPP network entities

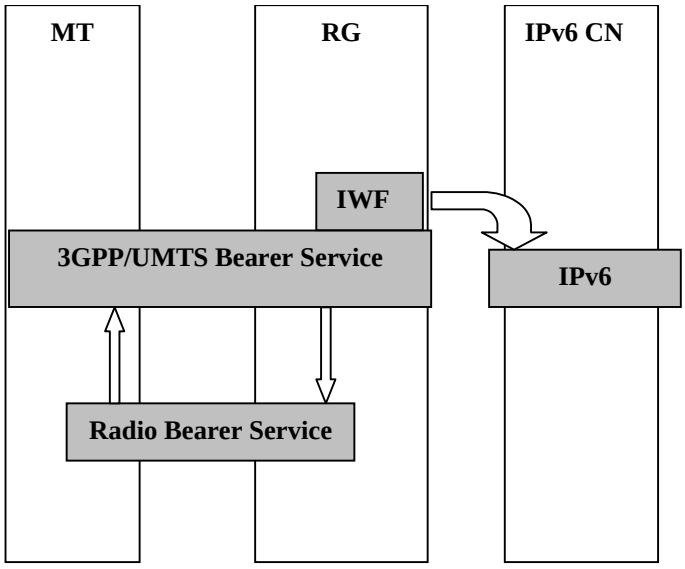


Figure 21: IPv6 CN proposal

Figure 20 shows the different networking entities of the standard 3GPP/UMTS specifications. Similarly in Figure 21 we show the architecture considered here based on an IP backbone. We can see from the figures that running IP to the Radio Gateway has an strong impact on the overall network architecture of the system, since we short circuit some basic networking entities (i.e. RNC/SGSN/GGSN). An interface that remains completely unmodified is the Radio Bearer Service. UMTS/3GPP bearer service is also required *a priori*. The need for the Radio Access Bearer Service, however, is questionable. The CN Bearer Service as well as the Iu Bearer Service is definitively not present in the proposed all-IP architecture since the corresponding interfaces do not exist. **This has an important impact on the various interface levels that the RG must offer.**

5.2 Global functions of the Radio Gateway.

The Radio Gateway is not only a Node-B or basestation since the Node B in the standard definition includes only the physical layer of a base station without complete MAC, RLC, PDCP and RRC protocol layers. The RG is composed of

- The physical layer + MAC layer
- An access to data traffic via the RLC (and optional PDCP) layer which is responsible to convey data over “Radio access Bearers”
- A control function (RRC) which provides an interface to setup and release radio bearers for signaling and data.

One can therefore view the Radio Gateway as a Node B + a subset of the RNC (Radio Network Controller.)

5.3 Elementary Procedures

5.3.1 Initialization phase

Let us first consider what is provided by the RG at initialization phase. The Mobile Terminal is first switched on. Then it enters the “Idle” state where it listens to the synchronization channel (SCH) and then to the BCH. After this first synchronization phase the MT is “physically” attached to the RG.

Then, on behalf of the NAS the MT connects to the RG. The primitive “UE Side Initiated Connection Establishment Request” is received by the MT RRC, which in turn sends a message “RRC connection request” to its peer RG RRC entity. This is acknowledge by the corresponding “response” message and primitive. From now on, the MT is in “connected” state meaning it has at hand a dedicated signaling radio bearer. The only service provided by the RG at this point is NAS signaling messages from/to the MT to/from the RG. This is done via the primitives “Data Transfer request”, “Data Transfer Indication” provided by the RRC.

After invoking these primitives, the NAS signaling message is sent via a DCCH logical channel. This logical channel is in turn mapped to either a RACH/FACH or DCH transport channel. The mapping is done according to a resource management strategy driven by the RG RRC taking into account traffic load conditions.

So at this point the MT is considered by the RG as being either in “Cell FACH connected” or “Cell DCH connected” state. Note that at this point no radio access bearer has been allocated for data transfer, where “data” means here any data you may want to send on a DTCH (dedicated Traffic channel)

5.4 Radio access bearer establishment

In order to transfer data over a radio bearer with a define QoS, the MT requires a radio access bearer then the NAS (IP) layer of the MT must send a message (via “data transfer request”) to the RG NAS in order to request a Radio Access bearer with a given QoS profile. The RG upper layer asks the RG RRC to open a radio bearer via the primitive “IF side initiated Radio Access Bearer establishment request” which in turn triggers a “radio bearer request” message to be sent from the RG RRC to the MT RRC peer identity. The corresponding “responses” are sent back. The NAS part of the RG receives the response from the RRC (“IF side initiated Radio Access Bearer Confirmed”) so that the RG NAS can acknowledge the NT NAS request. Note that when the radio access bearer is requested directly by the Infrastructure, we have a simplified procedure where the request is sent directly without a NAS handshaking (but it may necessitate Paging). From this point onward, we have a DTCH that is established with a given QoS.

5.5 Radio access bearer release

This is done with the same principles as for establishment. The Infrastructure (IF) is always responsible for that procedure. So all is triggered via the NAS if the MT is initiating the procedure or without NAS handshaking if IF is initiating the procedure. The RRC primitive is “IF side initiated Radio Access Bearer Release Request” and the corresponding message is “radio bearer release”.

5.6 IP QoS Classes mapping to UMTS Radio Bearer Mapping

3GPP documents specify that the mapping between application QoS attributes and the 3GPP bearer service attributes is rather an operator and/or implementation issue. Therefore, a mapping mechanism is required between the IP-based applications QoS parameters and the 3GPP QoS parameters.

IP QoS work is being conducted within Internet Engineering Task Force (IETF). This work focuses on two main streams for QoS control, namely *Integrated Services* architecture (IntServ) [13] and *Differentiated Services* architecture (DiffServ) [14]. Integrated services architecture tries to provide the QoS in an end-to-end manner like ATM. It allows users to communicate their QoS requirements to routers on the data path by means of a signalling protocol. Differentiated Services architecture specifies the IP header bit usage to differentiate between different QoS classes. The main objective of the DiffServ is to specify a QoS mechanism based solely on the contents of the IP header fields rather than on an end-to-end signalling protocol. We assume that our applications use either IntServ or DiffServ in order to specify their requirements in terms of QoS.

The following table depicts a possible mapping between 3GPP QoS classes and DiffServ QoS classes.

DIFFSERV	3GPP
Expedited Forwarding	Conversational Class
Assured Forwarding – Gold Class	Streaming Class
Assured Forwarding – Bronze Class	Interactive Class
Best Effort	Background Class

Table 2: Mapping of Differentiated Services Classes to UMTS QoS Classes

These mappings were studied in the context of another RNRT-funded project, SAMU (http://samu.crm-paris.com/Emain.htm).

First suppose that a mechanism such as DiffServ is used. (which may not be the case). This issue is related to the definition of a correspondence between a DiffServ Codepoint and a radio Bearer QoS and has an impact on discrimination of IP packets. Two cases must be considered. The first case is when no radio bearer is established. In this case, we suppose that the IWF will have to ask for a radio bearer with a given QoS deduced from scanning the Codepoint of the first IP packet to be transmitted. The second case is where the IP layer interacts with PDCP/RLC. Here the IP layer invokes the DATA request/indication primitive to send and receive data in the U-plane.

5.7 Other IP-based Services

5.7.1 Broadcast

This service is offered by RRC. All that is required from the upper layers is the NAS message to be sent and the value of the repetition period. More specifically, Mobile IPv6 router advertisement messages will be broadcast through this mechanism.

5.7.2 Paging

This service is also offered via RRC. More detailed information has to be provided on this service depending on the interaction at IP level. IP level paging may be required when all mobile terminal movements are not immediately reflected to the mobile terminal's home agent.

5.8 Handoff issues

There is no concept of soft handoff in 3GPP TDD. (no simultaneous Tx/Rx with several RGs). The RGs have to be loosely frame synchronized and the slot allocation must be identical. In this case the MT may demodulate several RG beacons and perform measurements at the radio level. We propose that the MT RRC layer reports to the NAS layer those RGs that offer a radio-link of acceptable quality and their correspoding identities. Identities may be "layer 2" identities, in which case we suppose that the network is pre-configured with the association "layer 2" identity and IP address.

6. Conclusions

This article focused on the overall software architecture of a 3GPP software radio for direct interconnection with an IPv6 backbone network. We described the basic hardware/software interface which consists of a PCI bus and low-level real-time processes for implementing the 3GPP TDD specifications for layers 1-3. This implementation was done under the hard real-time POSIX extension to the Linux operating system known as RTLinux. The architecture is portable to variety of processing platforms ranging from laptops to high-end servers for advanced multi-antenna basestations or IP radio-gateways (RG). Moreover, the software architecture can be adapted to system-on-a-chip (SoC) designs by transfering some algorithmically intensive routines into co-processors sharing the same system bus, while retaining the organizational benefits of a RTOS. We have described some of the novelties related to interconnecting 3GPP with an IPv6 backbone and their consequences on networking architecture.

7. References

[1] C. Bonnet, G. Caire, A. Enout, P. A. Humblet, G. Montalbano, A. Nordio, and D.Nussbaum, T. Höhne, R. Knopp, B. Rimoldi, “An Open-Architecture Software Radio Platform for University Research and Teaching”, Annales de Télécommunications, May 2001.

[2] 3GPP TS 25.201: "Physical layer - general description", ETSI 2001.

[3] 3GPP TS 25.221: "Physical Channels and Mapping of Transport channels onto Physical Channels (TDD)", ETSI 2001.

[4] 3GPP TS 25.222: "Multiplexing and channel coding (TDD)", ETSI 2001.

[5] 3GPP TS 25.223: "Spreading and modulation (TDD)", ETSI 2001.

[6] 3GPP TS 25.224: "Physical layer procedures (TDD)", ETSI 2001.

[7] 3GPP TS 25.302: "Services Provided by the Physical Layer", ETSI 2001.

[8] 3GPP TS 25.331: "RRC Protocol Specification", ETSI 2001.

[9] Vanu Bose, Mike Ismert, Matt Welborn, and John Guttag. “Virtual Radios”. In IEEE Journal on Selected Areas in Communications, vol. 17, no. 4, pages 591-602, April 1999.

[10] Massimiliano Laddomada, Fred Daneshgaran, Marina Mondin, Ronald M. Hickling, “A PC-based Software Receiver using a Novel Front-End Technology,” IEEE Communications Magazine, pages 136-145, August 2001.

[11] Bill Carpenter, Mark Roman, Nick Vasilatos, and Myron Zimmerman. “The RTX Real-Time Subsystem for Windows NT”. In Proceedings of the USENIX Windows NT Workshop, Seattle, WA, pages 33-37, August 1997.

[12] Michael B. Jones and Stefan Saroiu, “Predictability Requirements of a Soft Modem,”, In Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, Cambridge, MA, June 2001.

[13] J. Wroclawski, "The Use of RSVP with IETF Integrated Service", RFC 2210, September 1997.

[14] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.