A METHODOLOGY FOR ANALYZING POWER CONSUMPTION IN WIRELESS COMMUNICATION SYSTEMS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE,
FACULTY OF SCIENCE
AT THE UNIVERSITY OF CAPE TOWN
IN FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE BY COURSE WORK AND DISSERTATION

By
Mwelwa K. Chibesakunda
February 2004

Supervised by
Prof. Pieter S. Kritzinger

Abstract

Energy usage has become an important issue in wireless communication systems. The energy-intensive nature of wireless communication has spurred concern over how best systems can make the most use of this non-renewable resource. Research in energy-efficient design of wireless communication systems show that one of its challenges is that the overall performance of the system depends, in a coupled way, on the different submodules of the system i.e. antenna, power amplifier, modulation, error control coding, and network architecture. Network architecture implementation strategies offer protocol software implementors an opportunity of incorporating low-power strategies into the design of the network protocols used for data communication.

This dissertation proposes a methodology that would allow a software protocol implementor to analyze the power consumption of a wireless communication system. The foundation of this methodology lies in the understanding of the *formal specification* of the wireless interface network architecture which can be used to predict the performance of the system. By extending this hypothesis, a protocol implementor can use the formal specification to derive the power consumption behaviour of the wireless system during a normal operation (transmission or reception of data). A high-level formalism like state-transition graphs, can be used to track the protocol processing behaviour and to derive the associated *continuous-time Markov chains.*

Because of their diversity, *Markov reward models*(MRM) are used to model the power consumption associated with the different states of a specified protocol layer. The models are solved analytically using the Möbius *performance* and *dependability* tool. Using the MRM *accumulation* and *utilization* measures, a profile of the power consumption is generated. Results from the experiments on the protocol layers show the individual power consumption and utilization of the different states as well as the accumulated power consumption of the different protocol layers when compared. Ultimately, the results from the reward model solution can be used in the energy-efficient design of wireless communication systems.

Lastly, in order to get an idea of how wireless communication device companies handle issues of power consumption, we consulted with the wireless module engineers at Siemens Communication South Africa and present our findings on current practices in energy efficient protocol implementation.

Acknowledgements

This body of work would not have been possible if it was not for the inspiration and goodwill of those I consider noble, just and wise.

First and foremost, I would like to thank the good lord for blessing me with good health during the tenure of the masters degree.

To Professor Pieter Kritzinger, to whom this dissertation would not have been possible, I am forever grateful for taking me under the DNA wing and guiding me through the world of research. I have learnt invaluable lessons from you and being a member of this distinguished research group. In addition, I would also like to thank Dr Markus Siegle of University of Erlangen in Germany for his lectures in Stochastic and Markov theory.

To the PERFORM research group (Tod Courtney et al.), many thanks for helping me understand the Möbius tool. I predict very exciting times for you at University of Illinois.

To Kruger Murray at Siemens, thank you for all your input into my case study. You certainly helped us answer critical questions in our research.

To my proof readers Oksana, Lourens, Nico, Farrel, and Mwinji, I thank you for taking the time from your busy schedules to help me edit my write-up. Your comments were noted and many incorporated in this body of work.

To my friends Chad, Kumoyo, Cholwe, and Jack, many thank yous for supporting me mentally in getting to the ultimate goal. A special thank you to Binoy George for your invaluable comment, educated criticism, programming expertise and most of all your friendship.

Last, but not the least, to my family, Daddy Austin and sister Mwamba, without you non of this was ever imaginable. You are what motivates me to do the best I can. Though you were not here with me, sister Ngocha has kept me comforted. I thank God for all of you.

Contents

List of Figures

List of Tables

Chapter 1

Background and Problem Definition

1.1 Introduction

Power consumption has become a growing concern among designers of wireless communication systems. With increased mobility, the limited supply of battery power has become a known constraint to the continuous operation time of wireless devices [Eph02]. In addition, because of multimedia data and other diverse applications, mobile application software is now more processor intensive.

The challenge in designing energy-efficient wireless communication systems is that their overall performance depends, in a coupled way, on the following subsystems: *antenna*, *power amplifier*, *modulation*, *error control coding*, and *network architecture* [SWW$^+$02]. This dissertation confines itself to the area of network architecture protocols. The next section discusses some of the work that has been done in studying energy concerns in wireless communication systems.

1.1.1 Other Work

Research in the study of issues relating to energy efficiency in wireless communication systems revolve around the following areas,

1. Investigating energy consumption,
2. Power saving strategies, and

3. Energy efficient protocol design.

These categories are not mutually exclusive, albeit the ultimate goal is to develop systems that make the best use of available energy resources. We will now discuss each of these areas in turn in the following subsections.

Investigating Energy Consumption

Feeney in [Fee99] investigated a solid experimental basis for assumptions that could (or could not) be made in the design and analysis of network protocols operating in *ad hoc* wireless environments. This work investigated the implications of having either an *ad hoc* or *infrastructure type* network and how this would affect the power consumption of the wireless device. Later, Feeney and Nilsson in [FN01] collected detailed measurements of the energy consumption of an IEEE 802.11 wireless network interface also operating in *ad hoc* mode. The data collected was presented as a collection of linear equations for calculating the energy consumed in sending, receiving and discarding broadcast and point-to-point data packets of various sizes. Zorzi and Rao in [ZR01] analyzed the energy consumption performance of various versions of TCP, namely, *Tahoe*, *Reno* and *NewReno*, for bulk data transfer in an environment where channel errors are correlated. They investigated the performance of a single wireless TCP connection by modelling the correlated packet loss/error process (e.g., as induced by a multipath fading channel) as a *firstorder Markov chain.* Jones *et al.* in their comprehensive survey [JSAC01], presented the chief sources of power consumption with regard to network operations, as being classified into two types: *communication* related and *computation* related. Communication involves usage of the transceiver at the source, and destination nodes. Computation on the other hand, is mostly concerned with the protocol processing aspects.

Power Saving Strategies

A number of strategies have been suggested for saving power in wireless communication systems. Woesner *et al.* in [WJPSW98] outlined the main strategies adopted by known protocol standards like IEEE 802.11 and ETSI RES 10 HIPER-LAN in terms of power saving. For example, for the *Medium Access Control* (MAC) layer, power saving issues included optimized use of the *physical* layer services, optimized medium access protocol structure and optimized system design. Application-driven power management was presented by Kravets and Krishnan in [KK00]. In their work, they presented a novel transport level protocol for managing the suspend/resume cycle of mobile host's communication device in an effort to reduce power consumption. In [JSAC01], Jones *et al.* also stated that a significant amount of power can be saved by incorporating low-power strategies into the design of network protocols used for data communication. In addition, they conducted a survey of general conservation guidelines and mechanism that may be adopted for an energy efficient protocol design.

Energy Efficient Protocol Design

Work in energy efficient protocols has mostly been concerned with energy efficient protocol design and some of the constraints associated with it. While designing error control protocols, Zorzi and Rao in [ZR97] proposed a way of defining a metric for power based on the average number of correctly transmitted packets during the lifetime of the battery. This hypothesis was then applied by Chockalingam and Zorzi in [CZ98] to study the energy efficiency of a class of multiple access schemes, using the average number of correctly transmitted packets for a given amount of energy as an appropriate metric. What is interesting about their work is that they apply *Markov* theory to track the protocol evolution and compute the protocol throughput and energy performance. In [CSAK98] Chen *et al.* compared different designs of MAC protocols based on the transceiver usage perspective. They present a framework for comparison of energy consumption due to MAC related activities and apply this framework to a set of wireless protocols. Havinga and Smit in [HS01] identified the most prominent problems of wireless multimedia networking and present several solutions with a focus on energy efficiency, one of which focuses on involving all protocol layers of the system in the design. What is interesting about all these approaches to energy efficient design is that they attempt to optimize each protocol layer separately and may achieve global optimality only by coincidence [SWW$^+$02].

Our work concerns itself with the investigation of power consumption of the entire protocol stack of wireless communication systems and energy efficient protocol implementation. In the next section, we discuss the methodology we are proposing for analyzing power consumption in wireless communication systems.

1.2 A Methodology for Power Consumption Analysis

In this section, the methodology we are proposing is discussed in general and a layout of the rest of the dissertation is provided. The framework of the methodology we are proposing is illustrated in **Figure** 1.

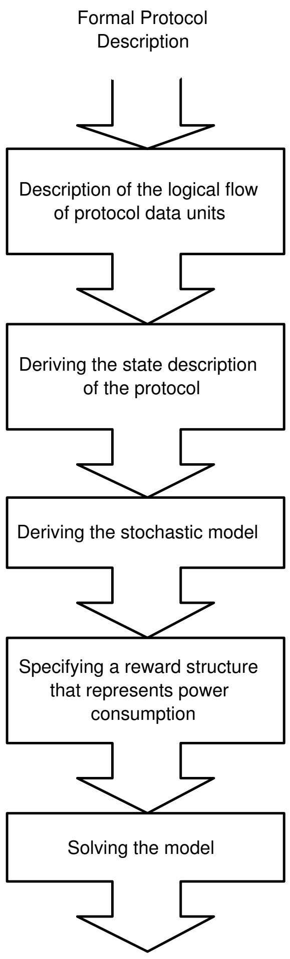

Formal Protocol Description
Description of the logical flow of protocol data units
Deriving the state description of the protocol
Deriving the stochastic model
Specifying a reward structure that represents power consumption
Solving the model

Figure 1: The Methodology Procedure in Outline

The following subsection discusses the different stages of the proposed methodology.

1.2.1 The Methodology

Formal specifications of protocols have for a long time been used for the prediction of the performance measures of protocols based on the OSI basic reference model [Kri86, Rud83]. As mentioned in **Section** 1.1 herein, the *network architecture* is one of the contributing factors to the energy-efficient design of wireless communication systems. In addition, there is a considerable amount of power consumed during the operations of the aforementioned. Though it is a significant subsystem, the *network architecture* poses a more difficult challenge in determining the empirical data associated with their power consumption because of the way the different layers interface with the hardware. We propose a methodology for analyzing the power consumption of a wireless communication system based on the functions and operations documented in the formal specification of their radio interface protocols. The following discussion describes the different parts of our methodology illustrated in **Figure** 1.

Description of Logical Flow of Data Units

Using the formal specification of a protocol, we can derive the logical flow of *protocol data units* (PDUs) as they progress through the layered architecture. While the PDUs make their way from *source layer* to *destination layer* (served and servicing layers), there are processes performed on these data units. Processes vary from a combination of instructions executing different protocol functions, to a number of messages needed to transfer a given amount or type of information [Svo89], among other things. Coupled together, these protocol processes contribute to power consumption.

Derivation of Protocol State Description

With an understanding of the processes associated with individual protocol layers, it can be deduced that behaviour of the protocol layer can be represented as a state transition graph where the vertices represent states and arcs represent state transitions. It should be pointed out here that this high level representation is protocol implementor dependent as the protocol specification documents do not provide hard and fast rules for this.

Derivation of Continuous-Time Markov Chain

Using the state transition graphs as our state space, we can derive a *Markov* representation of the protocol layer architecture [KB96]. We use the assumption made in [Hil03] that the behaviour of the real protocol system during a given period of time is characterized by the probability distributions of a stochastic process. The state transition diagrams in this case, indicate the evolution of the system in time and can be represented by the finite-state stochastic process which characterizes the dynamics of the protocol system.

Specification of Markov Reward Model

In order to analyze the power consumption of the protocols, we propose the use of *Markov reward models* (MRMs). Markov reward models are commonly used for the *performance*, *dependability* and *performability* analysis of computer and communication systems. The motivation for using MRMs is twofold:

1. The evolution of any protocol with finite memory can be modelled as a *Markov chain* [ZR97]; and

2. The freedom to modify the reward structure allows the modeler to represent a wide variety of operating conditions [STR88].

Model Solution

Once we have the MRM, we are able to solve it using the *performability* tool *Möbius* [CCD$^+$01] developed by W.H. Sanders *et al.* of the *PERFORM* research group at the University of Illinois. The tool allows us to solve different MRMs based on the specification of reward rates and reward structures. In particular, the tool can be used to predict the following results over an interval of time:

1. Power consumption in a protocol state, and

2. Power consumption of the protocol layer.

The next section discuses the layout of the rest of this dissertation.

1.2.2 Dissertation Outline

The layout of the rest of this dissertation is as follows:

Chapter 2 provides an introduction to *Markov theory* and *Markov reward models*(MRMs). The chapter also presents the Möbius modelling tool and a brief description of how the different editors of the tool are used to model a MRM.

Chapter 3 explains the general principles of the layered network protocol architecture. An example of a wireless protocol architecture is provided in *Universal Mobile Telecommunications System* (UMTS).

Chapter 4 proposes a way of modelling power consumption as a *Markov reward model* (MRM). It explains the *reward structure* used to model power consumption.

Chapter 5 shows the application of the methodology to UMTS and solving of the derived model. It tackles the problem of deriving the *continuous-time Markov chains* from the state transition diagrams.

Chapter 6 illustrates and discusses the results of the experiments conducted in **Chapter 5**.

Chapter 7 presents conclusions and suggestions for future work.

In the study of energy-efficient design of wireless communication systems, the challenge lies in the fact that the overall performance depends, in a coupled way, on the following subsystems [SWW$^+$02]:

Chapter 2

Markov Theory and Markov Reward Modelling

2.1 Introduction

In the first section of this chapter, we provide the definition of *Stochastic* and *Markov processes*. We then provide a brief discussion on some of the properties of *Markov processes* and subsequently describe a type of *Markov process* known as *continous-time Markov chains*. *Markov theory* forms the basis of the hypothesis of tracking protocol evolution in time. *Markov* theory has been extensively covered in literature; here we refer mainly to [KB96],[Mar03],[Tri82].

In the next section, we define *Markov reward models* (MRMS) and we classify their known properties of interest. Lastly, we profile a *Performabilty* tool, Möbius, which has been used to analyze our derived *Markov* models; here we refer to [HT93],[DS01][CCD$^+$01]

2.2 Markov Theory

The following summary on the relevant *Markov* properties is taken from [KB96] and [Mar03].

2.2.1 Markov Processes

Markov processes are a class of stochastic processes whose conditional probability density function[Mar03] is such that

$$\begin{aligned} &P[\chi(t) = x|\chi(t_n) = x_n, \chi(t_{n-1}) = x_{n-1}, \chi(t_0) = x_0] \\ &= P[\chi(t) = x|\chi(t_n) = x_n],\ \ t > t_n > t_{n-1} > \ldots > t_0 \end{aligned} \tag{1}$$

The above condition is known as the *Markov property.* A *Markov process* is a stochastic process $\{\chi(t), t \in T\}$ for which this property holds. We will assume that $T = [0, \infty]$ in this dissertation and write $S = \{x_i : i \in \mathrm{N}_0\}$ [1]

Definition 2.1 Homogeneous Markov Processes. *A* Markov process $\{\chi(t)\}$ *is said to be* homogenous *or* stationary *if the following condition holds*

$$P[\chi(t+s) = x|\chi(t_n + s = x_n] = P[\chi(t) = x|\chi(t_n) = x_n] \tag{2}$$

The equation expresses that a homogenous *Markov process* is invariant to shifts in time. We will now provide some of the *Markov processes* properties that are of interest to us.

2.2.2 Markov Properties

Consider states $i, j \in S$. If there is a path from i to j, i.e., there exists an integer n (which may depend on i and j) such that

$$p_{ij}(n) > 0$$

where p_{ij} is the probability of a transition between state i and j.

Two states i and j are said to *communicate*, written $i \rightleftharpoons j$, if there is a path from state i to state j and vice versa.

Let $C[i] = \{j|i \rightleftharpoons j;\ \ j \in S\}, \forall i \in S$. We call $C[i]$ the *class* of state i.

Definition 2.2 *A MC is said to be* irreducible *if every state communicates with every other state.*

An irreducible MC clearly has only one class of states, i.e. $C[i] = C[j]\ \ \forall i, j \in S$.

[1] N denotes the set of positive integers and N_0 additionally includes the 0.

Definition 2.3 *A class C is said to be* transient *if there is a path out of C. That is, if $\exists i \in C$ and $k \in \overline{C}$ such that $p_{ik} > 0$. The individual states in a transient class are themselves said to be* transient.

Definition 2.4 *A class C is said to be* ergodic *if every path which starts in C remains in C. That is*

$$\sum_{j \in C} p_{ij} = 1, \quad \forall i \in C$$

The individual states in an ergodic class are called ergodic. An irreducible MC consists of a single ergodic class, i.e. $C[i] = S, \forall i \in S$.

Next write $f_j^{(m)}$ for the probability of a *Markov* process leaving a state j and *first* returning to the same state j in m steps. Clearly the probability of *ever returning* to state j is given by

$$f_j = \sum_{m=1}^{\infty} f_j^{(m)}$$

We now classify the states j of a MC depending on the value f_j of the state. Not surprisingly, if $f_j = 1$ we say the state is said to be *recurrent*; if a return is not certain, that is $f_j < 1$, then state j is said to be *transient*. Furthermore, if our MC can return to state j only at steps $\eta, 2\eta, 3\eta, \ldots$, where $\eta \geq 2$ is the largest such integer, then state j is said to be *periodic with period* η. If such an integer number η does not exist, then the state j is said to be *aperiodic*.

Knowing the probability $f_j^{(m)}$ of returning to state j in m steps, we can now define another interesting quantity, the *mean recurrence time* M_j of state j.

$$M_j = \sum_{m=1}^{\infty} m f_j^{(m)} \tag{3}$$

The mean recurrence time is thus the average number of steps needed to return to state j for the first time after leaving it.

We can further describe a state j to be *recurrent null* if $M_j = \infty$, whereas it is *recurrent nonnull* if $M_j < \infty$. Note that an irreducible MC can only have recurrent null states if the number of states is infinite.

With all this in mind, we can now state the following important theorem:

Theorem 2.1 *The states of an irreducible* MC *are all of the same type; thus they can be either*

- *all transient,*
- *all recurrent nonnull, or*
- *all recurrent null.*

Moreover, if periodic, then all states have the same period η.

2.2.3 Continuous Time Markov Chains

If the states of a *Markov process* are discrete and the process may change state at any point in time, the process is said to be a *Continuous Time Markov Chain* [Ste94]. The formal definition is as follows:

Definition 2.5 *The stochastic process $\{X(t)\}$ forms a* ***continuous-time Markov chain*** *(CTMC) if for all integers n, and for any sequence $t_0, t_1, ..., t_n, t_{n+1}$ such that $t_0 < t_1 < ... < t_n < t_{n+1}$, we have*

$$P[\chi(t) = x | \chi(t_n) = x_n, \chi(t_{n-1}) = x_{n-1}, ..., \chi(t_0) = x_0] = P[\chi(t) = x | \chi(t_n) = x_n] \qquad (4)$$

For a *continuous-time Markov chain*, the exponential distribution is the only distribution that satisfies the *Markov property.*

State Transition Diagrams

For small CTMCs the simplest way to illustrate a process is often in terms of its state transition diagram. In such a diagram, each state of a process is represented as a node in a graph. Arcs are then used to represent the transitions between states of the process. The arcs are then labelled by the *transition rates* between states.

Steady State Distribution

Performance analysis is usually concerned with the behaviour of systems over an extended period of time. For the models we will be considering, the following theorems explain steady-state distribution:

Theorem 2.2 *When a CTMC chain is irreducible, homogenous and all its states are recurrent nonnull, it can be shown that the following limit exists and is independent of the initial state distribution. That is,*

$$\lim_{t \to \infty} \pi_j(t) = \pi_j \tag{5}$$

The vector $\prod = (\pi_0, \pi_1, \pi_2, \ldots)$ is the steady state distribution of the CTMC. In addition, the CTMC is said to be ***ergodic*** *(c.f. Definition 2.4).*

When the CTMC is irreducible, aperiodic and homogeneous the following theorem helps us out.

Theorem 2.3 *In an irreducible and aperiodic homogeneous MC the limiting probabilities π_j always exist and are independent of the initial state probability distribution. Moreover, either*

1. *all states are transient or all states are recurrent null. In both cases $\pi_j = 0 \; \forall \; j$ and there exists* no *steady state distribution, or*

2. *all states are recurrent nonnull and then $\pi_j > 0 \; \forall \; j$, in which case the set $\{\pi_j\}$ is a steady state probability distribution and*

$$\pi_j = \frac{1}{M_j} \tag{6}$$

A recurrent nonnull MC is also referred to as an ergodic *MC and all the states in such a chain are ergodic.*

Theorem 2.4 *In a finite, homogeneous, irreducible, continuous-time Markov chain, the limiting probabilities $\{\pi_j\}$ always exist and are independent of the initial probability distribution. Moreover, the set $\{\pi_j\}$ is also a stationary probability distribution which can be uniquely determined from solving the set of equations:*

$$+ q_{jj}\pi_j + \sum_{i \neq j} q_{ij}\pi_i = 0 \tag{7}$$

$$\sum_j \pi_j = 1 \tag{8}$$

where the latter equation follows from the rules for probability conservation. In matrix form Eq. 7 may be expressed as

$$\pi Q = 0 \tag{9}$$

where Q [Ste94] represents the *infinitesimal generator matrix* of a *continuous-time Markov process* with n states. The entry in the jth column of the ith row of the matrix ($j \neq i$) will be q_{ij}, the mean transition rate. In other words, it will be the sum of the parameters labelling arcs connecting nodes i and j in the state transition diagram. The diagonal elements are chosen to ensure that the sum of the elements in every row is zero, i.e.

$$q_{ii} = - \sum_{j \in S, j \neq i}^{n} q_{ij} \tag{10}$$

The set of equations given by **Theorem** 2.4 is sometimes referred to as the set of *global balance equations.*

2.3 Markov Reward Models

A *Markov Reward model* (MRMs) consists of a *continuous-time Markov chain* $\{X(t), t \geq 0\}$ with finite state space S, and a reward function r where $r : S \rightarrow \Re$. For each state $i \in S, r(i)$ represents the reward obtained per unit time in that state. This type of MRM is called a rate-based MRM.

There are four types of measures [HT93] supported by MRMs and they are listed below:

- *Steady state measures*;
- *Transient measures*;
- *Cumulative measures*; and
- *Performability measures.*

Steady state measures express the long run gain per unit time of the system. They are computed from the steady state behaviour of the *Markov chain. Transient measures* (or instant-of-time measures) express the rate at which gain is received from the system at any particular time t. They are computed from the transient behaviour of the system.

Cumulative measures express the overall gain that is received from a system over some finite time interval. They are computed when transient measures are integrated over a specified time interval $[0, t]$. Lastly, *Performability measures* are the a distribution of cumulative measures that express whether a prespecified gain y can be received from the system in some finite time interval $[0, t]$.

We use MRMs because they allows us the freedom to modify the reward structure to represent a variety of situations[STR88]: in this case, power consumption of the state space of the radio interface protocols of a wireless communication system.

2.4 The Möbius Modelling Tool

To analyze the *Markov chains* we have derived, we use the *Möbius Modelling tool* [CCD+01] developed by William H. Sanders *et al.* of the *Performability Engineering Research Group* (PERFORM) of the University of Illinois at Urbana-Champaign. *Möbius* is a multi-formalism modelling tool used to study the performance/dependability of a stochastic system. The aim of the tool is to provide support for many different modelling formalisms. These include traditional performance modelling paradigms such as networks of queues and *stochastic Petri nets* (SPNs), newer approaches such as *stochastic process algebras* (SPAs), *Markov processes* and *stochastic automata networks* (SANs), and more specialized approaches such as fault trees and combined block diagrams. We selected this tool because it supports the high level specification of *continuous-time Markov chains* through the *Buckets and Balls* formalism [Ste94].

The following sections will now provide a brief account of the architecture of the tool followed by the steps to analyzing a performabilty model with the tool below.

2.4.1 Möbius Architecture

The Möbius tool has a number of components that are divided into two categories: *model specification* components and *model solution* components. We use figure 2 from [CCD+01] to describe the components of the tool.

Model specification. Models are specified through a front-end, which consists of a set of graphical user interfaces running within a main Java interface. The tool is organized as a series of editors, classified according to model types. The editors and their functions are as follows:

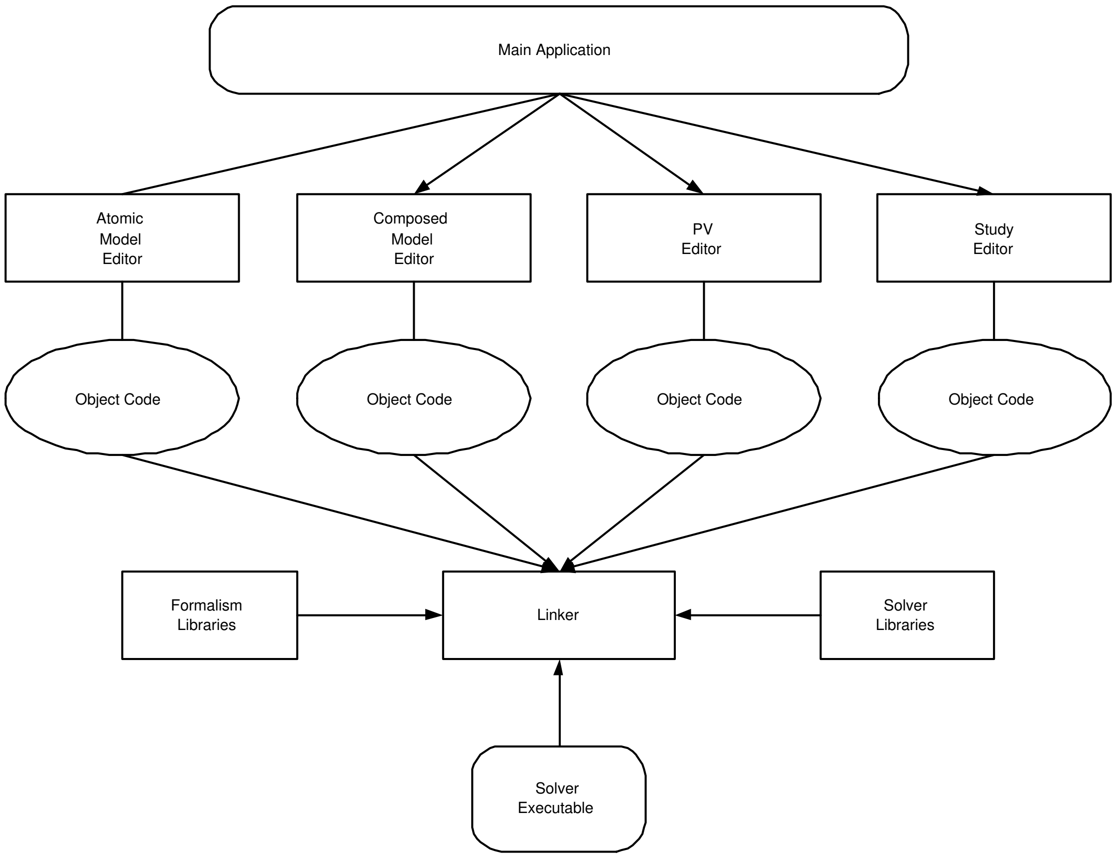


Figure 2: Möbius Tool Architecture

- *Atomic model editor.* This editor is used to design the model based on the performance formalism the user is implementing i.e. *Markov chains*, *stochastic Petri nets*, or *stochastic process algebras.* For example, CTMCs can be modelled using the *Buckets and Balls* formalism [Ste94]. A bucket represents a state, the arcs represent the transitions and the ball enables a state.
- *Composed model editor.* This editor allows a modeler to adopt a hierarchical approach to modelling, by constructing submodels as meaningful units and then placing them together in a well-defined manner to construct a model of a system. The result of this is a system of submodels that are able to read from and write to each other. This form of state sharing is is known as *equivalence sharing.*
- *Performance Variable editor*

  This editor allows *reward models* to be built upon atomic and composed models, equipping them with the specification of a performance measure. Currently, the tool has one type of reward model implemented: a *performance variable* (PV). A PV allows for the specification of a measure on one or both of the following:

  - the states of a model, giving a *rate reward* PV
  - action completions, giving an *impulse reward* PV

  For our analysis, we shall only consider a rate reward PV and the specification is explained in the next paragraph.

  A rate reward is a function of the state of the system at an instant of time. A performance variable can be specified to be measured at instant of time, measured in steady state, accumulated over a period of time, or accumulated over a time-averaged period of time. Once the rate reward is defined, the desired statistics on the measure must be specified. The options include solving for the mean, variance, or distribution of the measure, or the probability of the measure falling within a specified range.
- *Study editor* This editor allows for the selection of a desired analytical/numerical study. The tool supports a number of analytical/numerical solvers. The first step in analytic solution with Möbius is the generation of a state space, done by the Möbius state-space generator. After the state space is generated, any of the following analytical/numerical methods, listed below, may be employed to solve for the required performance variables:

1. *Transient solver, trs*,
2. *Iterative Steady State solver, iss*,
3. *Accumulated Reward solver, ars*,
4. *Adaptive transient solver, ats*,
5. *Deterministic Iterative Steady State solver, diss*, and
6. *Advanced Deterministic Iterative Steady State solver.*

Table 1 from [PER94] gives the measure and model classes for each of the above listed analytic solvers.

Analytic Solvers				
Model Class	Steady-state or Transient	Instant-of-time or Interval-of-time	Mean, Variance or Distribution	Applicable Analytic Solver
All activities exponential	steady-state	Instant-of-time	Mean, Variance, and Distribution	*dss* and *iss*
	Transient	Instant-of-Time	Mean, Variance and Distribution	*trs*
		Interval-of-time	Mean	*ars*
			Distribution	*pdf*
Exponential and deterministic activities	Steady-state	Instant-of-time	Mean, Variance and Distribution	*diss*

Table 1: Models and measures versus the applicable analytic solver

Model solution. To solve the specified *performability variable*, Möbius offers two solutions:

1. Discrete-event simulation,
2. Analytical evaluation.

Discrete-event simulation can be used to solve models with arbitrarily large state spaces. It often requires that the simulation be run a number of times before suitable results can be produced. For this reason, we shall not use simulation in our model analysis. Analytical evaluation on the other hand requires that the state-space and state transition matrix of the stochastic process be generated to solve the model. In addition, the activities of the process have to be exponentially distributed and possess the *memoryless* property of a *Markov* process.

2.4.2 Stochastic Processes

The behaviour of a system can often be characterized by enumerating all the states that the system may enter and by describing how the system evolves from one state to another over time. In its most general form, such a system can be represented by a *stochastic process*.

Chapter 3

Network Architecture

3.1 Introduction

The first step in understanding how concurrent communication software is implemented in a wireless system involves understanding the *network architecture* of the system. The objective of this chapter, is to provide the general principles of *network architecture* design. In this first section of the chapter, we discuss how the *Open Systems Interconnection reference* model is the basic framework by which most network protocols are designed and standardized. This will include understanding the purpose of protocol hierarchies, architecture specifications, and the issues associated with protocol engineering. For this we will refer mainly to [Kri86], [Svo89], [Zim80], [Tan96], and [Kri03].

In the latter section, we provide an example of a radio interface protocol architecture in *Universal Mobile Telecommunications System* (UMTS). We focus on the *user equipment* (UE) side of the protocol architecture. This will include illustrations and a description of how *protocol data units* (PDUs) progress through the selected protocol layers during normal operation. Here we refer mostly to the *European Telecommunications Standards Institute 3rd Generation Partnership Project*, also known as the ETSI 3GPP series of *Technical Specification* (TS) documents [vR02e][vR02g][vR02h][vR02a].

3.2 Network Architecture Theory

3.2.1 The OSI Reference Model

As long ago as 1976 the *International Standards Organization* (ISO) recognized the need for some standardized way in which different computer systems could communicate with each other. Because of the way digital information is transformed from the useful user application data to the actual transmitted bits, they proposed to describe the architecture of the network software in layers. These layers would be an *open system* that subsequently facilitated the interconnection of dissimilar systems. What developed from this, is what is now known as the *Open System Interconnection Reference Model*, or OSI reference model for short, whose fundamental purpose is to act as a reference architecture for network design. **Figure** 3 from [Kri03] illustrates this model

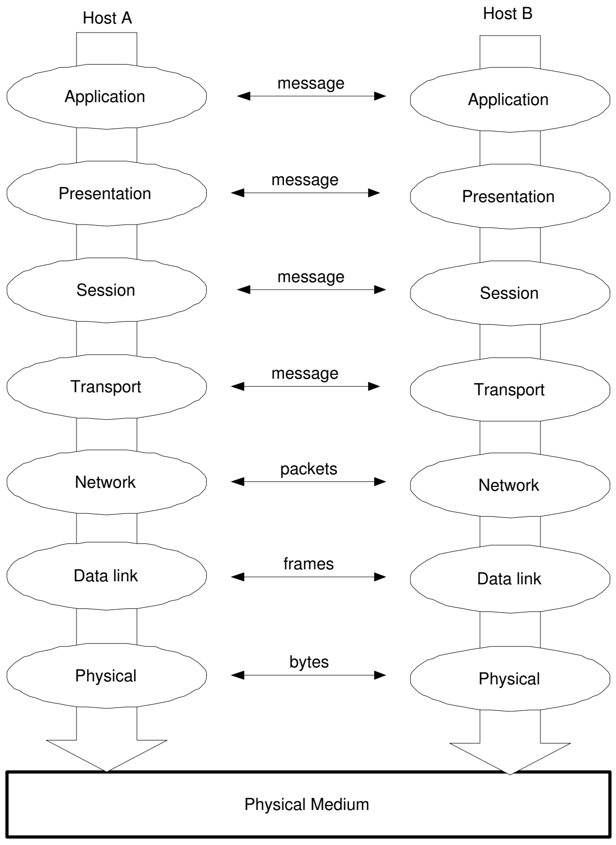


Figure 3: The ISO OSI Reference Model

3.2.2 Protocol Hierarchies

The purpose of having a layered architecture is to reduce the design complexity of the communicating systems. With this schema, the networks are organized as a series of layers or levels, where each one is built upon the one below it. The number of layers, the name of each layer, the contents of each layer, and the function of each layer differ from network to network. Between each pair of adjacent layers is an interface. The interface defines which services the lower layer offers the upper one.

When network designers decide how many layers to include in a network and what each one should do, one of the most important considerations is having clearly defined interfaces between the layers. Having clearly defined interfaces, in turn, requires that each layer performs a *specific collection of well understood functions.* In addition to minimizing the amount of information that must pass between layers, clear cut interfaces also make it simpler to replace the implementation of one layer with a completely different one because all that is required of the new implementation is that it offers exactly the same set of services to the next layer higher up as the old implementation did. According to Liba Svobodova in [Svo89], an OSI system can be structured in many ways, such as by grouping layers in a way similar to the Internet model.

3.2.3 Network Architecture Specification

The set of layers and protocols is called the *network architecture.* The specification of the architecture must contain enough information to allow an implementor to write the program for each layer so that the program will correctly obey the appropriate protocol. Neither the details of the implementation nor the specification of the interface are part of the architecture.

When describing the operation of any protocol layers, it is important from the outset to discriminate between:

1. the **services provided** by the layer,
2. the **protocol** (i.e. logical operation of the layer), and
3. the **services used** by that layer.

This is important because only then can the function of each layer be defined in the context of the other layers. This also has the effect that a person implementing one protocol layer needs only to have knowledge of the services the layer is to provide to the layer above, the internal operation (protocol) of the layer, and the services that are provided by the layer below to transfer the appropriate items of information, called *Protocol Data Units* or PDUs associated with the protocol to the corresponding layer in a remote computer.

Generally, the OSI standards for each individual layer are presented in two documents: the *service definition* [Svo89] and the *protocol specification* [BS80]. These documents consist of an informal description of the functions, definitions of variables, predicates and actions.

3.2.4 Protocol Engineering Issues

For any protocol implementation, it is desirable that the communications systems software has to be correct and efficient. To achieve this, the process of protocol engineering involves the following procedures: *Textual specification*, *Formal description*, *Implementation*, and generation of the *Executable application*. **Figure** 4 shows the outline of the different stages of the protocol engineering process which are discussed in turn in the following subsections.

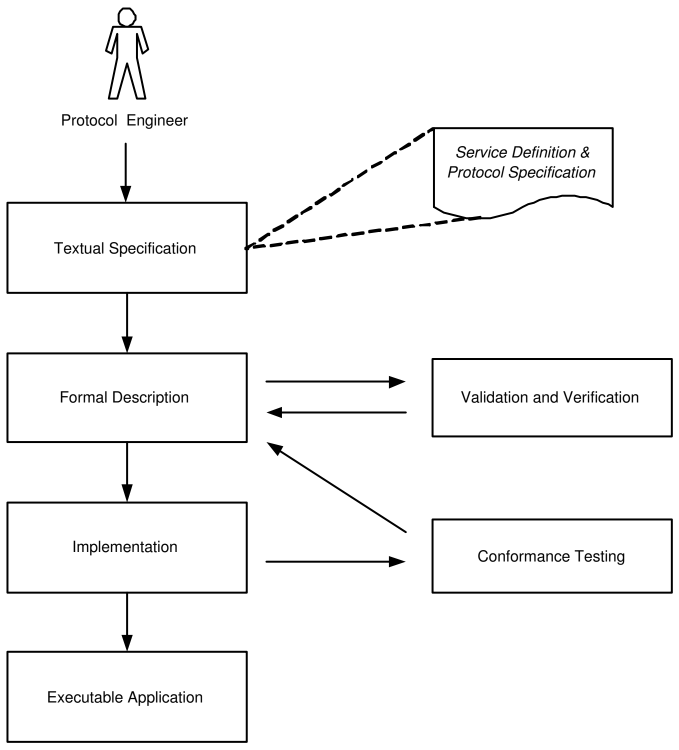


Figure 4: The Protocol Engineering Process

Textual Specification

In **Section** 3.2.3 of this dissertation, we discussed the two important documents of protocol specification: the *service definition* and the *protocol specification.* These are the blueprints the protocol implementor uses in designing the protocol. At this stage, the protocol implementation is purely at the implementor's discretion and any problems that arise out of these specifications are considered as a local implementation matter.

Formal Description

Because *Concurrent Communicating Systems* (CCS) are non-deterministic, protocol implementation is very difficult. A solution to this problem is the use of *Formal Description Techniques* (FDTs) [Rud85], which provide a well-defined set of concepts that can easily be communicated between human developers. In addition, FDTs provide an unambiguous, clear and concise description of the services, protocols or other aspects of a CCS. In addition, they are used to determine the correctness of the specification, validate the system specification, and aid in judging the future system. Examples of well known FDTs include the *Specification and Description Language* (SDL) [HBS94], and *Estelle* [BD87].

Implementation

Implementation of the protocol is mostly done by automated derivation from the formal specification of the protocol. According to Svobodova in [Svo89], not only does this ensure the implementation conforms to the standard but also speeds up the implementation process, though there are still many problems to be solved before an automated implementation process becomes widely feasible. Hence automated implementations are more focused on a single layer. On the other hand, the interfaces between the adjacent layers are hand-coded. The hand-coded part includes the generation and analysis of PDUs.

In order to have an efficient implementation of the protocol, *conformance testing* is performed on the software. This ensures that the software being implemented is "valid" in the sense that it implements the standard it claims to implement. At this stage the whole process is recursive as the implementor may have to return to the *formal description* of the protocol.

Executable Application

Once the protocol has been verified as being correct, deadlock free, and validated to ensure it adheres to the standards, it is ready for incorporation in the layered architecture of the system. Because of this layered architecture, a particular protocol layer can easily be replaced when a better implementation of the protocol is available.

3.3 An Example of A Radio Interface Protocol

In this section we provide an example of a radio interface protocol architecture in *Universal Mobile Telecommunications System* (UMTS). **Figures** 5 and 6 from [Gra02] illustrate the UMTS and *UMTS Terrestrial Radio Access Network* (UTRAN) architectures and in particular, where the radio interface under consideration is situated. A brief outline of these architectures is provided in the following discussion.

The architecture of a UTRAN consists of one or several *Radio Network Systems* (RNS), which correspond to a *Base Station Subsystem* (BSS) common in GSM radio access networks. A RNS is comprised of a *Radio Network Controller* (RNC), which controls several

Node Bs. The term *Node B* refers to a site where several radio base stations are co-located, with each such radio base station serving one cell. A *Node B* is connected to a RNC via an interface, denoted as *Iub* Interface. There is also an interface between RNCs, denoted as *Iur* Interface. Such an interface does not exist in the GSM access networks. In GSM, there is only a small amount of communication between different RNS entities needed, which is handled through the Core Network but which requires radio access network depend functions in the Core Network Domain.

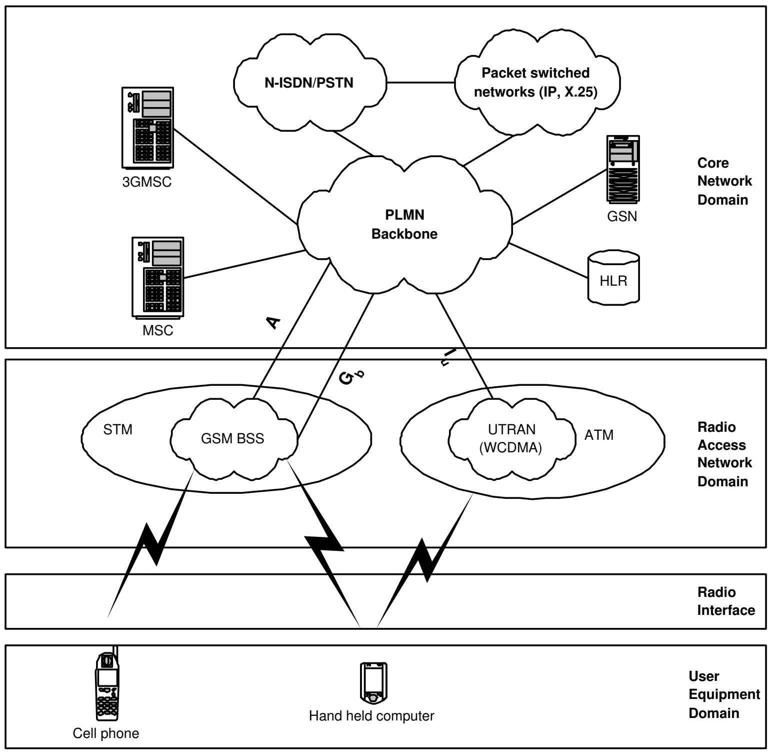


Figure 5: The UMTS Architecture

In this architecture, the *User Equipment* (UE) can simultaneously be served by mutiple Node Bs. If a UE is connected to Node Bs of different RNS, the *Iur* interface is needed to exchange the data with the Serving RNC (S-RNC), which at a given time provides the connection with the Core Network via an interface, denoted as *Iu* interface.

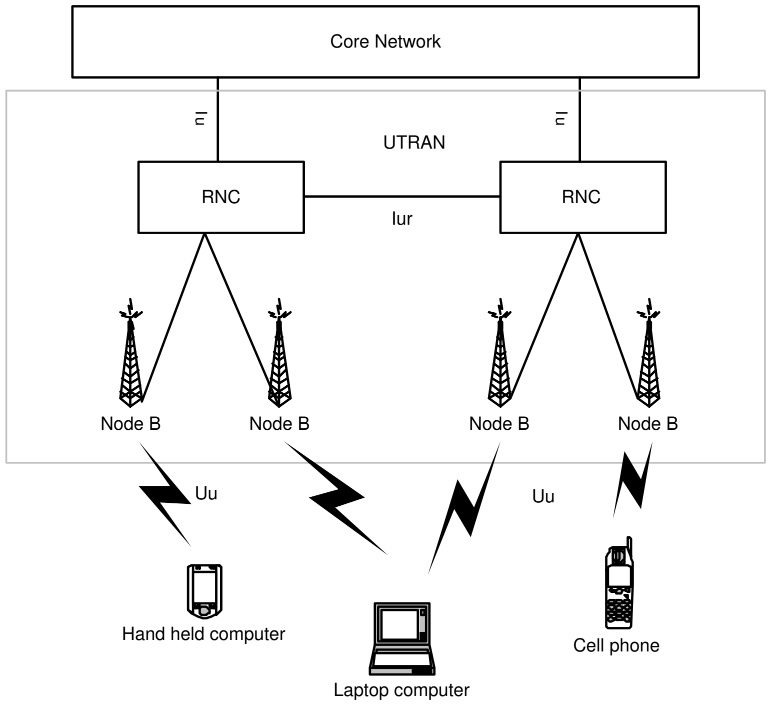


Figure 6: UTRAN Architecture

In the following subsection, we discuss the protocol description based on the ETSI 3GPP series of specification documents [vR02e][vR02g][vR02h][vR02a].

3.3.1 Radio Interface Protocol Architecture

The radio interface of UMTS is arranged in a series of protocol layers like most communication networks. It is layered into three protocol layers[vR02a], which include:

1. the *Physical layer* (L1),
2. the *Data link layer* (L2), and
3. *Network layer* (L3).

Layers 2 and 3 are further broken down into other sublayers. Layer 2 consists of the following sublayers: *Medium Access Control* (MAC), *Radio Link Control* (RLC), *Packet Data Convergence Protocol* (PDCP), *Broadcast/Multicast Control* (BMC), and *Link Access Control* (LAC). Layer 3 consists of the following sublayers: *Radio Resource Control* (RRC), *Mobility Management* (MM), and *Call Control* (CC). **Figure** 7, extracted from the *Radio interface protocol architecture diagram* in [vR02a], illustrates the overall architecture of the radio interface.

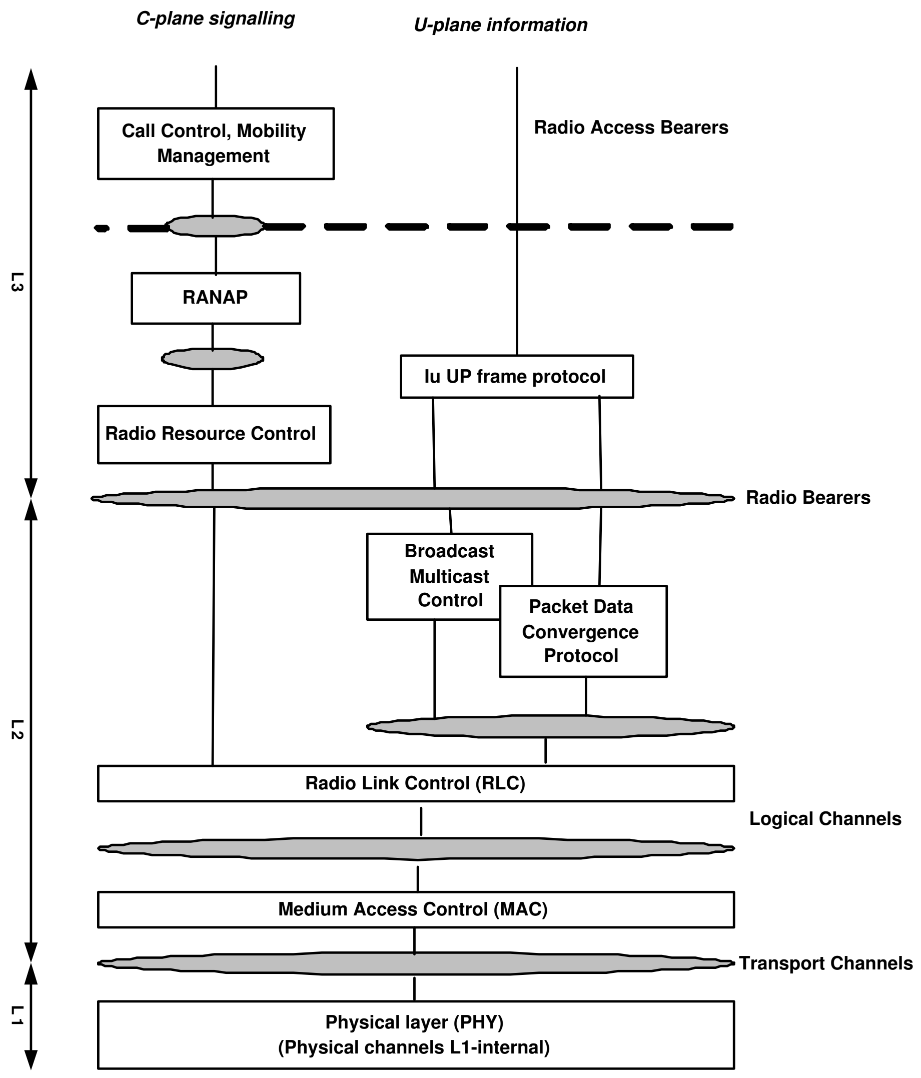


Figure 7: Architecture of the Radio Interface Protocol

UMTS is a connection-oriented transmission system. Layer 3 and RLC are divided into what are know as Control- and User- planes [Gra02]. The control plane protocols are responsible for handling and transfer of network-internal control information including connection setup. The user plane protocols are responsible for processing and transfer of the actual user application data, e.g. speech or video data, file transfer data, etc.

3.3.2 Services and Functions of the Protocol Layers

In addition to data transport service, each of the radio interface protocols provides additional services to upper layers of the protocol stack. From the moment either entity (mobile device or UTRAN) initiates communication, there is a path that is followed from the application initiating communication (i.e. sending a message) right through to the peer-end application receiving this message. Associated with this path, are various states assumed by each of the protocols in order to accomplish the transmission.

Higher layers

The higher layers in UMTS refer to the application layers above layer 3. UMTS specifies two modes of operation for the UE: Idle mode and UTRAN Connected mode. The UE procedures in idle mode include [Gra02]:

- cell search,
- cell selection/reselection,
- initial access, and
- registration to Core Network (CN).

Layer 3

This layer is responsible for the setup of connections, and the routing of data through the network i.e. *Internet Protocol* (IP). The following are protocols associated with this layer:

1. **Call Control (CC)** provides call management functions, such as setup, maintenance, release of so-called *signaling connections* between the Core Network and a mobile (UE). Through CC all signalling between a mobile and external networks is handled. The rudimentary CC procedures are call establishment procedures, call clearing procedures, call information phase procedures, and other miscellaneous procedures.

2. **Mobility Management (MM)** is responsible for mobility management on radio access level (RNS level). It manages user-specific databases needed to establish the

optimal connection between the Core Network and the Radio Access Network nodes. In addition, Mobile system selection (whether e.g. UMTS or GSM shall be used, operator selection etc.) and authentication are handled by the MM. Depending on how they can be initiated, three types of MM procedures can be distinguished: MM common procedures, MM specific procedures, and MM connection management procedures.

3. **Radio Resource Control (RRC)** provides mobility management on cell level, and radio resource management (admission control, handover control). Basically, it handles all signalling procedures between UE and UTRAN, and configuration procedures of the lower layers (L1 and L2)in both UE and in the network.

Layer 2

This layer is responsible for the setup of connections between network nodes, physical medium access control, flow control, error detection, and managing retransmissions. The following are the protocols associated with this layer:

1. **Broadcast-Multicast Protocol (BMC)** manages the distribution of messages received from the Cell Broadcast Center to the desired cells on the network side. It has scheduling procedures for generating scheduling information which enables BMC at the UE side to control *Discontinuous Reception* (DRX), so that the UE can read only those messages to which it had subscribed.

2. **Packet Data Convergence Protocol (PDCP's)** has the main function of TCP/IP or UDP header compression, i.e. it removes all redundant information from the headers which does not need to be sent repeatedly over the radio interface. Header compression procedures are specific to this protocol. Associated with this procedure are three modes of PDCP data transfer: Transparent data transfer, Data transfer without sequence numbering, and Data transfer with sequence number. These modes are differentiated by the headers of the IP packets.

   Figure 8 illustrates the logical flow of data through the protocol. The services the PDCP provides to higher layers are:

   - transfer of user data;

- maintenance of PDCP SDU sequence numbers.

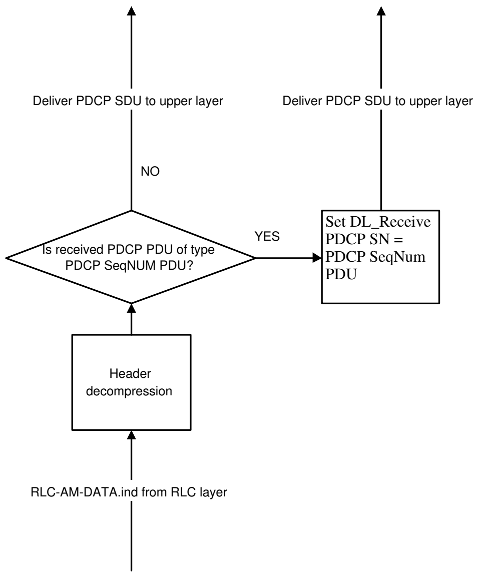

Deliver PDCP SDU to upper layer
Deliver PDCP SDU to upper layer
NO
YES
Is received PDCP PDU of type PDCP SeqNUM PDU?
Set DL_Receive PDCP SN = PDCP SeqNum PDU
Header decompression
RLC-AM-DATA.ind from RLC layer

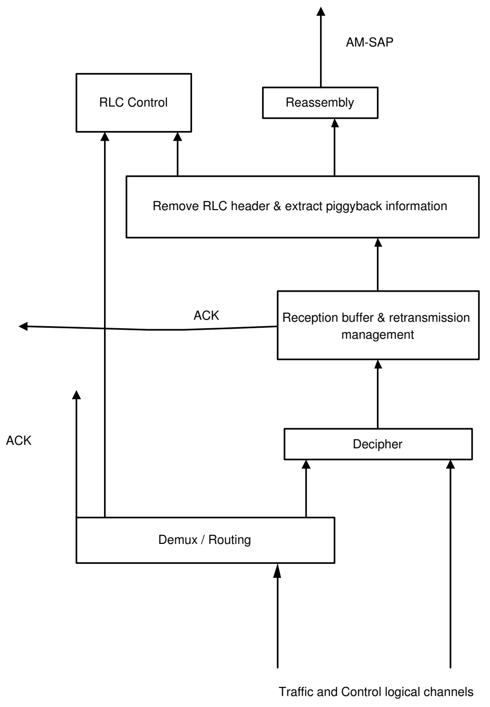
Figure 8: Logical Flow Through the Packet Data Convergence Protocol

3. **Radio Link Control (RLC)** protocol performs at the transmitting side segmentation of the higher layer protocol data units into smaller blocks suitable for radio transmission. At the receiving side it reassembles the small blocks back into higher layer data units. Segmentation and Re-assembly procedures are the basic procedures of this protocol. Additional procedures are available but they depend on the transmission mode the UE or UTRAN are in. Three modes of transmission are specified for the RLC: Transparent (TM), unacknowledged (UM) and acknowledged (AM) transmission.

   - In *transparent mode*, operations include segmentation, transmission buffering, receiver buffering, and reassembly.
   - In *unacknowledged mode*, operations include segmentation and concatenation, ciphering, adding RLC header, transmission buffering, receiver buffering, removing RLC header, deciphering, and reassembly.
   - In *acknowledged mode*, operations include segmentation / concatenation, ciphering, adding RLC header, retransmission buffer and management, multiplexing,

transmission buffering, adding piggyback information, de-multiplexing / routing, receiver buffering and retransmission management, removing RLC header and extracting piggyback information, deciphering, and reassembly.

Figure 9 from [vR02h] illustrates the logical flow of data through the protocol layer.



AM-SAP
RLC Control
Reassembly
Remove RLC header & extract piggyback information
ACK
Reception buffer & retransmission management
ACK
Decipher
Demux / Routing
Traffic and Control logical channels

Figure 9: Logical Flow through the Radio Link Control layer

4. **Medium Access Control (MAC)** controls the usage of the transport channels which are provided by the physical layer. It receives data on logical channels from the RLC, which it then multiplexes and maps onto the transport channels. In addition to multiplexing and mapping procedures, the MAC also performs priority control procedures, switching between transport channel procedures, and ciphering procedures in transparent mode RLC.

   Figure 10 from [vR02g] illustrates the logical flow of data through the protocol. The services that the MAC provides to the upper layers are [vR02g]:

   - *Data transfer*: This service provides unacknowledged transfer of MAC SDUs between peer MAC entities without data segmentation.

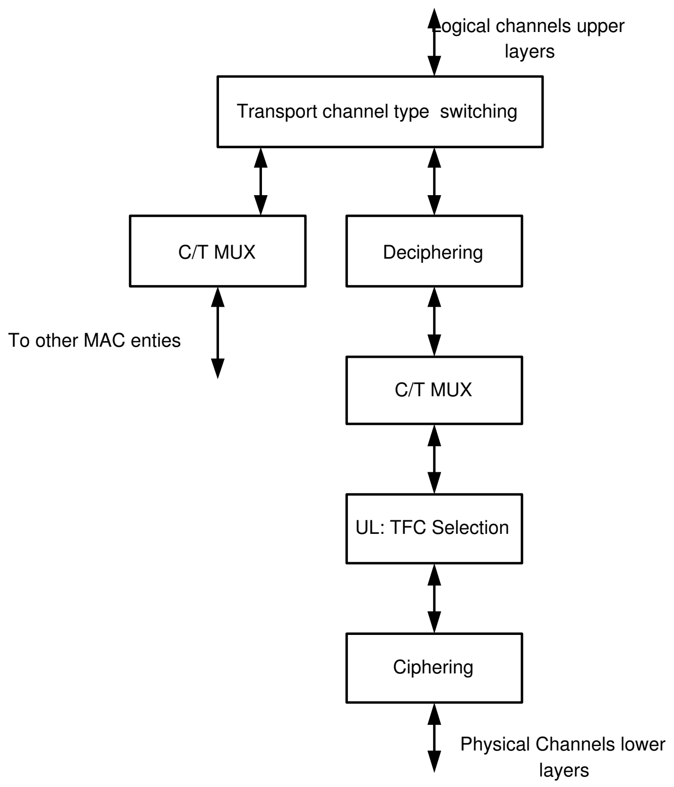

Logical channels upper layers
Transport channel type switching
C/T MUX
Deciphering
To other MAC enties
C/T MUX
UL: TFC Selection
Ciphering
Physical Channels lower layers

Figure 10: Logical flow through the MAC 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.
- Reporting of measurements: Local measurements are reported to RRC.

Layer 1

This layer is responsible for the transmission of data over the physical medium, synchronization, forward error correction, and modulation of the radio signal. The following discussion is a description of the different functions and procedures associated with this layer.

1. **Physical layer (PHY,L1)** offers information transfer services to MAC and higher layers. The following are some of the important procedures performed by the physical layer:

    - FEC encoding/decoding of transport channels;

- measurements and indication to higher layers (e.g. Frame Error Rate (FER), Signal-to-Interference Ratio (SIR), interference power, and transmission power);
- macrodiversity distribution/combining and soft handover execution;
- error detection on transport channels;
- multiplexing of transport channels and demultiplexing of coded composite transport channels;
- rate matching;
- mapping of coded composite transport channels on physical channels;
- modulation and spreading/demodulation and despreading of physical channels;
- frequency and time (chip, bit, slot, frame) synchronization;
- closed-loop power control;
- power weighting and combining of physical channels;
- RF processing;
- support of Uplink Synchronization;
- timing advance on uplink channels.

3.3.3 Protocol Coupling

This subsection describes how the different protocol layers work together to achieve communication between the *User Equipment* UE and the UTRAN through the different states of the radio interface protocols. We have chosen two common modes of operation: *Idle* and *UTRAN Connected mode* and we refer mostly to [Gra02] and the specification documents [vR02d],[vR02b],[vR02h],[vR02e], and [vR02c].

Idle mode

When a UE is switched on, it first performs a cell search procedure to locate a *public land mobile network* (PLMN) from the application protocol. The application protocol sends a signal to the Call Control protocol to perform a call establishment procedure. This procedure initiates a connection request through the *connection manager-Radio Resource Control* (RLC). The RRC performs the connection establishment procedures which specify the Common Control Channel as the logical channel and the *Random Access Channel*

(RACH) and *Forward Access Channel* (FACH) as the physical channels for communication with the UTRAN. The *Radio Link Control* (RLC) then prepares the connection request packets through the RLC transparent mode segmentation procedures, which break down the higher layer protocol data units into smaller smaller blocks suitable for radio transmission. The RLC then relays the requests packets to the *Medium Access Control* (MAC) protocol through the *Common Control Channel* (CCCH). The MAC then performs multiplexing procedures on the request packets, and maps them on the RACH for transmission to the physical layer. The *Physical protocol* (PHY) performs a series of procedures that ensures that the (UE) has acceptable communication conditions with the base station. This includes preparation and sending of beacons know as "RACH preambles" to the base station. The purpose of this phase is that first the base station shall obtain synchronization to this signal and indicates its successful acquisition to the UE on the *Acquisition Indicator Channel* (AICH). The process of sending the preambles involves the preparation of *Physical Random Access Channel* (PRACH) packets that are then modulated and spread in the respective radio frequency. During this transition, open loop power control procedures are performed to ensure that packets transmitted are received and decoded by the receiving base station. Once the UE has sent the initial access message in idle mode, it waits for a reply from the network on the AICH. The reply could be either a "RRC Connection Setup" or a "RRC Connection Setup Reject" message. Upon receiving the reply signal from the UTRAN, the signal is de-spread and demodulated to extract the reply packets. The packets are then decoded and demultiplexed for mapping onto the FACH. Measurement indicators are also sent to the RRC and the MAC with information on the Signal-to-Interference ratio(SIR), interference power, and transmission power on the physical channel. The MAC receives the reply signal on the FACH and performs error handling, error reporting to the RLC and RRC, traffic measurements and ciphering on the packets received. The FACH signal is then demultiplexed and mapped onto a CCCH. The RLC receives the reply signal on the CCCH and performs reassembly of small blocks protocol data units back into the higher layer units. An indication signal is then sent to the RRC on the response from the UTRAN. The RRC subsequently notifies the application layer on whether or not to remain in *idle mode* or enter *UTRAN Connected Mode.* **Figure** 11 from [Wil99] illustrates the protocol architecture for initial access.

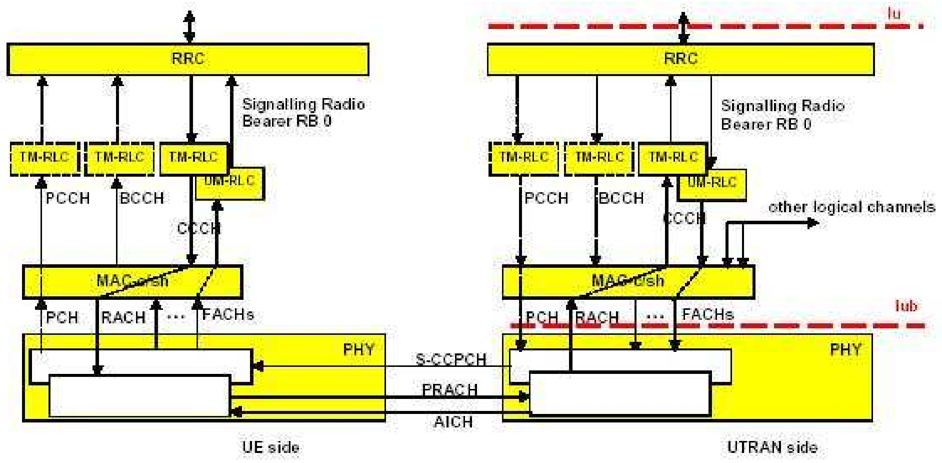


Figure 11: Protocol Architecture for Initial Access (Arrows down: Network side; arrows up: UE side)

UTRAN Connected mode: Downlink at the UE

This section describes how the different protocol layers handle requested HTTP packets on the downlink. Focus is on the User-plane protocols that are responsible for processing and transfer of the actual user application data [Gra02]. Of equal importance are the indication signals, sent from the lower layer protocols to the upper layer protocols, which carry the packets up to the application layer. It is understood that the basic units of protocol communication are:

1. *Protocol Data Units (PDUs)* used for peer-to-peer communication;
2. *Service Data Units (SDUs)* used for layer-to-layer communication.

Once a mobile has entered into UTRAN connected mode, the peer-to-peer communication between UE and Node B is extended to include the SRNC as a peer entity. The following user-plane protocol layer entities participate in peer-to-peer communication to achieve transmission of application layer packets (i.e. HTTP):

- Physical layer;
- MAC layer;
- RLC layer;

- RRC layer;
- PDCP layer;
- IP layer;
- TCP layer.

Figure 12 from [Gra02] shows the architecture of radio and transport protocols participating in the transmission of packet data.

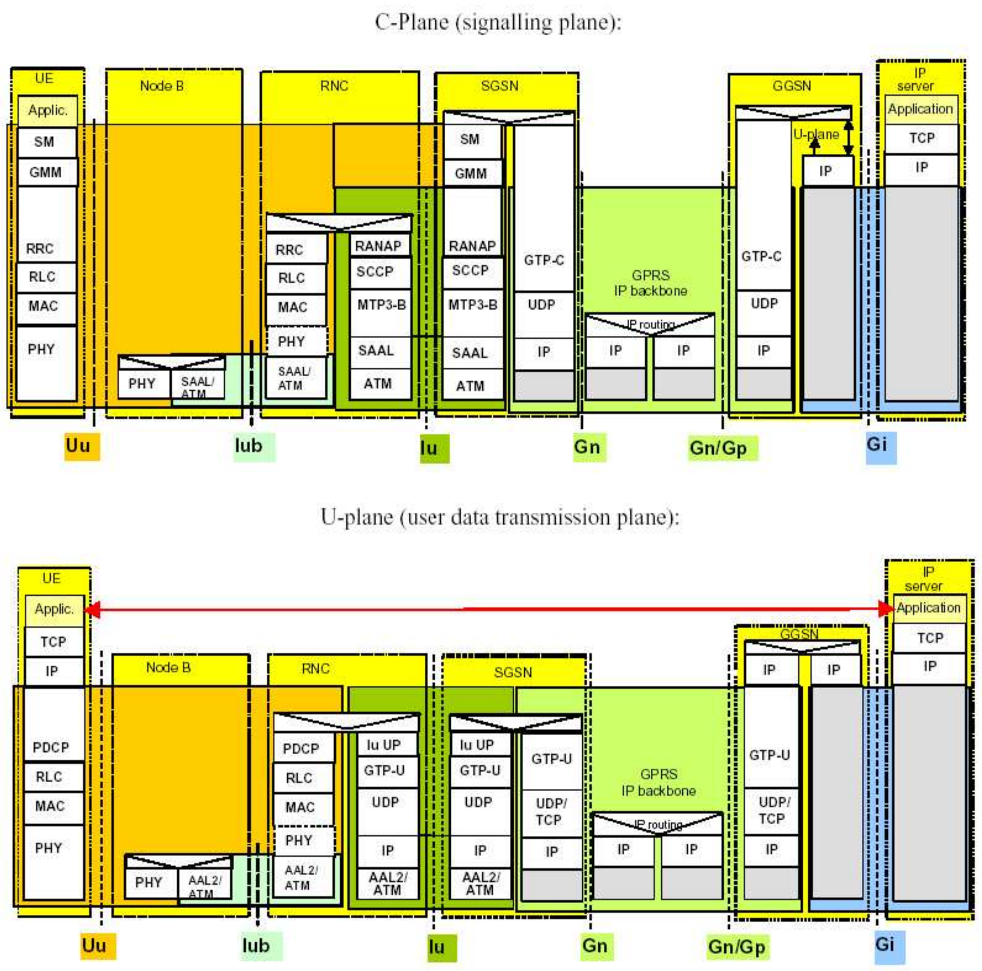


Figure 12: Radio and Transport Protocols Involved in Packet Data Transmission

Layer 1 (Physical)

The UE receives bits on a number of physical channels[1]. Demodulation and despreading of the transport channels is performed yielding the transport blocks of the transmission. Measurements of FER, SIR, interference power, and transmission power are performed during reassembly of the transport blocks and results are subsequently added to every transport block. Together, these processes yield the following primitives [vR02e]:

- *PHY-DATA-IND*[2] (sent to MAC);
- *PHY-STATUS-IND* (sent to MAC);
- *CPHY-MEASUREMENT-IND* (sent to RRC);
- *CPHY-OUT-OF-SYNC-IND* (sent to RRC);
- *CPHY-ERROR-IND* (sent to RRC);

The parameters for the L1 SDUs (MAC PDUs) include:

- Transport format indicator (TFI);
- Transport Block Set (actual transmitted data);
- CRC check result;
- Process Id.

The SDUs for the MAC and RRC are subsequently mapped to control and traffic channels.

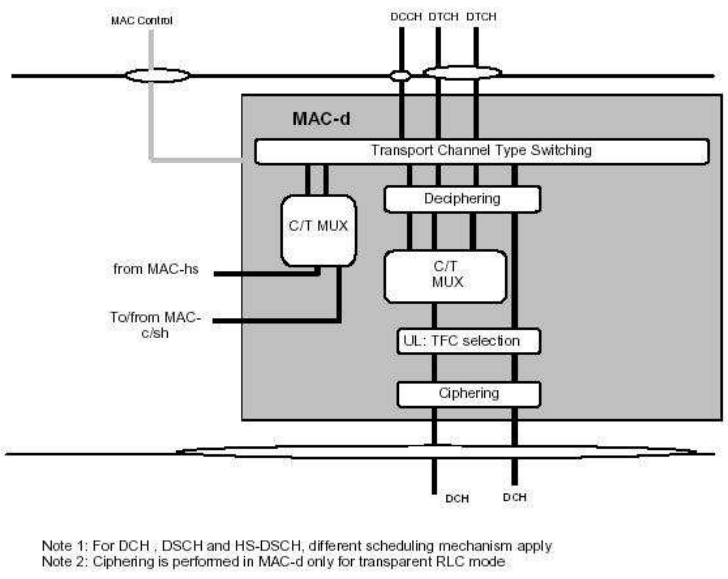
Layer 2 (*Medium Access Control*)

The MAC layer receives and indication signal from the physical layer. This layer consists of the following entities that could receive the indication signal, depending on which physical channel was used:

- *MAC-b*;
- *MAC-c/sh*;

[1]This is determined by the duplexing technique used: Frequency division duplexing (FDD)or Time division duplexing (TDD).

[2]Nomenclature for physical layer service primitives.



Note 1: For DCH , DSCH and HS-DSCH, different scheduling mechanism apply
Note 2: Ciphering is performed in MAC-d only for transparent RLC mode

Figure 13: UE Side MAC Architecture / MAC-d Details

- *MAC-d* (Illustrated in **Figure** 13 from [vR02g].);
- *MAC-hs.*

Each of these entities is responsible for mapping logical channels to transport channels. Of interest to us, is the *MAC-d* entity which maps the Dedicated Control Channels (DCCH) and Dedicated Traffic Channels (DTCH)on Dedicated Channel (DCH) which is carrying the data packet.

The delivered lower layer transport block sets are demultiplexed and subsequently mapped onto upper layer PDUs (RLC PDUs). In addition, the MAC entities perform measurements on the amount of traffic being received and the results reported to the RRC via the control channels. If transparent mode transfer is used by the RLC, then deciphering may also be performed on the PDUs. The primitives produced by these procedures include:

- *MAC-DATA-IND* (sent to the RLC);
- *MAC-STATUS-IND* (sent to the RLC);
- *CMAC-MEASUREMENT-IND* (sent to the RRC);
- *CMAC-STATUS-IND* (sent to the RRC).

Layer 2 (*Radio Link Control*)

Figure 14 from [vR02h] illustrates the flow of data in RLC when the protocol is in *Acknowledged Mode* (AM). For Internet traffic, the RLC employs AM transmission [Gra02] for "non-real time" packet data. The receiving side of the AM-RLC entity receives Acknowledged mode data (AMD) and Control PDUs through the configured logical channels from the lower layer. AMD PDUs are routed to the Deciphering Unit, where AMD PDUs (minus the AMD PDU header) are deciphered (if ciphering is configured and started), and then delivered to the Reception buffer. The AMD PDUs are placed in the Reception buffer until a complete RLC SDU has been received. The Receiver acknowledges successful reception or requests retransmission of the missing AMD PDUs by sending one or more STATUS PDUs to the AM RLC peer entity, through its transmitting side. If a Piggybacked STATUS PDU is found in an AMD PDU, it is delivered to the Retransmission buffer and Management Unit at the transmitting side of the AM RLC entity, in order to purge the buffer of positively acknowledged AMD PDUs, and to indicate which AMD PDUs need to be retransmitted.

Once a complete RLC SDU has been received, the associated AMD PDUs are reassembled by the Reassembly Unit and delivered to upper layers through the AM-SAP. RESET and RESET ACK PDUs are delivered to the RLC Control Unit for processing. If a response to the peer AM RLC entity is needed, an appropriate Control PDU is delivered, by the RLC Control Unit to the transmitting side of the AM RLC entity. The received STATUS PDUs are delivered to the Retransmission buffer and Management Unit at the transmitting side of the AM RLC entity, in order to purge the buffer of positively acknowledged AMD PDUs, and to indicate which AMD PDUs need to be retransmitted [vR02h].

The primitives sent to higher layers include:

- *RLC-AM-DATA-IND*;
- *RLC-UM-DATA-IND*;
- *RLC-TM-DATA-IND*;
- *CRLC-STATUS-IND.*

Layer 2 (*Packet Data Convergence Protocol*)

The *Packet Data Convergence Protocol* (PDCP) entity is responsible for header decompression of network protocols for IPv4. When the PDCP entity receives the PDCP PDUs (RLC

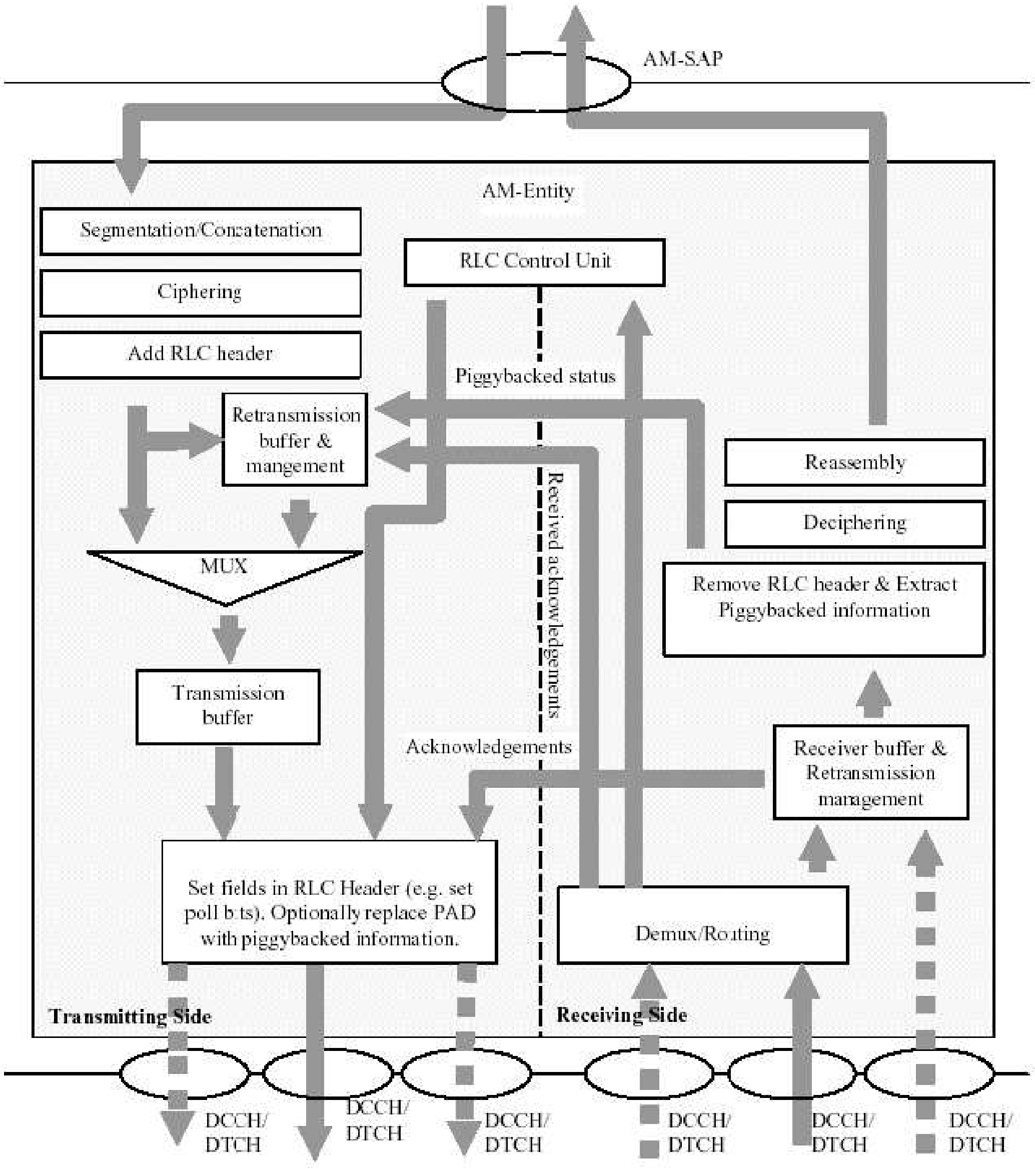

AM-SAP
AM-Entity
Segmentation/Concatenation
Ciphering
Add RLC header
RLC Control Unit
Piggybacked status
Retransmission buffer & mangement
MUX
Transmission buffer
Received acknowledgements
Acknowledgements
Set fields in RLC Header (e.g. set poll bits). Optionally replace PAD with piggybacked information.
Reassembly
Deciphering
Remove RLC header & Extract Piggybacked information
Receiver buffer & Retransmission management
Demux/Routing
Transmitting Side
Receiving Side
DCCH/ DTCH
DCCH/ DTCH
DCCH/ DTCH
DCCH/ DTCH
DCCH/ DTCH
DCCH/ DTCH

Figure 14: Model of an RLC Acknowledged Mode Entity

SDUs) from the lower layer, it performs header decompression of the PDCP PDU to obtain the PDCP SDU. This PDCP SDU is basically the data packets sent from the higher layer on the network side. It is subsequently delivered to the upper layer in the order received from the lower layer. The primitive sent to the upper layer is the *PDCP-DATA-Ind*. **Figure** 15 from [vR02c] illustrates the PDCP data transfer over AM RLC.

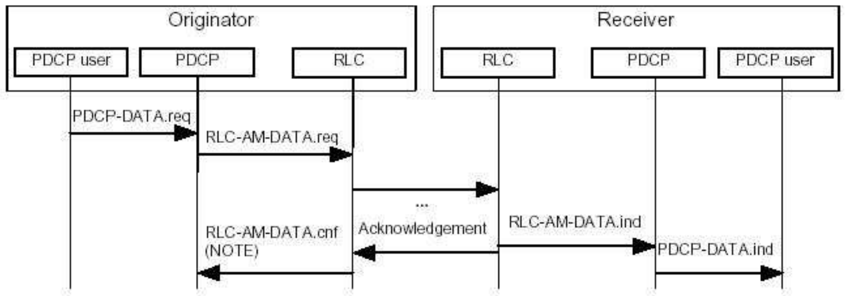


Figure 15: PDCP Data Transfer over Acknowledged Mode RLC

Chapter 4

Power Consumption as a Reward

4.1 Introduction

In this chapter we will apply the theory of *Markov reward models* (MRMs) for analyzing the power consumption of protocol processes. We use a *continuous-time Markov chain* to capture the progressive behaviour of a protocol layer, i.e the *functions* and *procedures* that process the protocol data units in a layer. With this behaviour, we associate a type of measure that accurately describes the characteristics of power usage in a protocol layer. Based on this premise, we explain how the expected power consumption behaviour can be predicted by modelling using MRMs.

In the next section, we discuss protocol behaviour based on information from the *formal specification documents*.

4.2 Protocol Behaviour

Formal specification documents provide information on the logical flow of protocol data units through a network architecture. The network architecture in this case represents the layered protocol stack. To the protocol implementor writing the program for each layer, the specification provides the *services* of the protocol, its *functions* and the *protocol data units* (PDU) exchanged between adjacent layers. With this information, the implementor is able to distinguish between the different states of a protocol whilst in execution. We understand this because while a program is in execution, different defined functions of the program will be called depending on what service the protocol is providing or receiving from its adjacent layer. Alternatively, the protocol implementor can use *Formal Description Techniques* (FDTs) [Kri03] to perform a meta-execution of the protocol specification to evaluate the protocol. Most FDTs, for example SDL [HBS94], are able to generate the program source code for protocol execution that can be used to explore the behaviour of the protocol during its execution.

For a protocol in execution, we can assume that it has continuous-time behaviour. We make this assumption because of the unpredictable nature of the wireless communication medium. A typical protocol layer can at any time be sending protocol data units, or be performing error correction due to retransmission. We are able to model this behaviour by using *Markov theory* [ZR97]. Our approach to deriving the *Markov* chain is determined by the fact that whilst a layer is in an arbitrary protocol state (protocol function in execution), there is a amount of time associated with being in that state before the next state change. This is known as the *sojourn time.* We assume the distribution of the layer being in an arbitrary state is *exponential* and that subsequent state changes are determined by the present state: in other words, it possesses the memoryless property of a *Markov chain.*

The *Markov* chain that has the properties discussed thus far is the *continuous-time Markov chain.* The protocol processes in this case represent the state space S, where $S = \{i, i \in \mathrm{N}_0\}$. For a protocol with n states, we let q_{ij} be the *mean transition rate* from state i to state j and Q be the $n \times n$ generator matrix.

In the next section, we explain how we can use *Markov reward models* (MRMs) to model power consumption at the protocol layer.

4.3 Power as a Reward

In this section, we present our approach to modelling power consumption in protocol layers as rewards of a *Markov reward model* (MRM). We establish power consumption as a possible metric of interest in stochastic representation of protocol processes.

4.3.1 Type of Measure

For any process that contributes to the transmission or reception of a data in the wireless communication system, there is an amount of power that is consumed from the battery. With the level of abstraction the protocol specification documents provide, it is a non-trivial task formulating a suitable metric for power. In addition, based on extensive investigation on our part, getting the exact power consumption values associated with protocol layer states is more difficult because such information is not documented. However, we can make the following hypothesis from the protocol behaviour. While the protocol layer is in a particular state, we expect a certain amount of time to be spent in that state. In addition, while in that state, it is obvious that a certain amount of power to be used in order to complete that task. Alternately, if we couple the consumption of power by the different states, power consumption can be viewed as a measure which depends on the accumulated behaviour of all states of a protocol over the utilization interval of that protocol: Power is the reward associated with being in a state. These stochastic behaviors combined with the reward functions form *Markov reward models* [Qur96]. For our purpose, power can be classified as an *interval-of-time* variable for the analysis of *Markov reward models.* The *accumulated reward* earned up in time is defined by [G. 90] and [CMST88] as follows:

Definition 4.1 *If r_i is the reward rate associated with the structure-state i: then the vector $\boldsymbol{r}$ defines the reward structure. A real-valued reward rate r_i is associated to each state $i \in S$. Assuming that the reward rates are non-negative and have no absorbing states, S_A:*

$$\forall i \in S,\ (r_i \in \mathrm{N}, i \in \mathrm{N}, r_i \in R, r_i \geq 0)\ \wedge\ (i \in S_A \Rightarrow r_i = 0) \tag{11}$$

The accumulated reward earned up in time t is defined by

$$Y(t) = \int_0^t r_{X(\tau)} d\tau. \tag{12}$$

4.3.2 Reward Definition

Now that we have specified the measure of interest, we have to select a type of reward that is suitable for the analysis of the accumulated behaviour of the system. There are two types of rewards that are common to *Markov reward models*: *rate rewards*, which assign time-dependent values to state occupancies, and *impulse rewards*, which assign time-independent values to state transitions. Power consumption is a time-dependent value therefore we shall confine ourselves to *rate rewards* because they are suitable for the interval-of-time measure we are interested in.

By interpreting rewards as energy levels, we have derived the following as our definition of power consumption in relation to rewards:

Definition 4.2 *Let c_i, $i \in \mathrm{N}_0$, where i represents the state index, represent power consumption as a reward in the* Markov Reward Model *(MRM).*

Though this definition of power consumption provides a way of predicting how the actual power of the device would be consumed by the different protocol states S, it is not sufficient in modelling power consumption of the layer because it does not incorporate parameters such as the the number of packets and their frame size, which would aid in providing a more realistic model. Alternatively, we can assume this to be a coefficient that is part of the overall consumption of the protocol layer. In [FN01], Feeney and Nilsson describe the energy consumed by the network interface when a host sends, receives or discards a packet under error free conditions by the following linear equation:

$$Energy = m \times size + b \tag{13}$$

The constant coefficients m and b are used to represent values for various operations i.e. packets discarded and packets received. They further state that there is a fixed component associated with device state changes and channel acquisition overhead and an incremental component which is proportional to the size of the packet. Since the protocol architecture is coupled with the wireless interface [SWW$^+$02], we can intuitively assume that this description is analogous to the protocol layers as well. We categorize the aforementioned modelling of power consumption as the *Power Factor Analysis*.

Another way of using the MRM framework is by using it as a pure performance model [GPT00] to conveniently describe power consumption as a measure of interest. For our purpose, we can analyze the power utilization of the different protocol states. We assume that once the protocol layer is in operation, sending and receiving protocol data units, and the effects of the initial bias have been overcome, the network architecture will enter into steady state behaviour. What this implies is that the model exhibits regularity and predictability over its state space. When the model is in *steady state*, we denote the steady state probabilities i.e., π_i, for the probability that the model is in state i. The utilization, ρ, can be computed based on a binary reward assignment. This is a special MRM where if a particular resource is occupied in a given state, the reward rate 1 is assigned, otherwise reward rate 0 indicates the idleness of the resource. With reward structure $r_0 = 1$ and $r_i = 0$ for $i = 1, 2, 3, \ldots, n$ the utilization becomes:

$$\rho = \sum_{i \in S} r_i \pi_i = \pi_0 \tag{14}$$

This allows us to determine the probability vector π and therefore the fraction of time the protocol is in a particular state. This information proves useful when the protocol implementor wants to determine which of the protocol states the layer spends the most time in and what contribution, to the overall power consumption, that particular state makes. We categorize the aforementioned modelling of power consumption as the *Power Level Analysis*.

Lastly, MRMs can be used to capture the accumulated behaviour of a reward based on the reward specification defined for the model. In this case, if we specify a reward structure that describes the consumption of power by each protocol state, the combined *accumulated reward* of all the states of a specific layer can be used to compare the power consumption characteristics of the various protocol layers specified in a particular layered architecture. This information would allow the protocol implementor to distinguish which protocol layers of a particular implementation are power intensive. We categorize the aforementioned modelling of power consumption as the *Power Consumption Comparison* analysis.

4.4 Model Assumptions

In this section we discuss the different assumptions used for deriving the *Markov reward models* from the protocol layers. Hereafter, we validate these assumptions based on information gathered during an interview with a *protocol engineer* from Siemens Communications South Africa.

4.4.1 Stochastic Behaviour and Parameter Values

The most fundamental assumption we have made in this work so far is the **stochastic hypothesis** [Hil03]: "*The behaviour of the real network architecture during a given period of time is characterized by the probability distributions of a stochastic process.*" Although it can never be conclusively proved, evidence from the meta-execution of the protocol suggest that this is true. In contrast, the assumption that all protocol state time are exponentially distributed, is less likely to fit with observations of real systems. We make this assumption nevertheless because the exponential distribution is the only one we can use if we wish to assume that the stochastic process characterizing our network architecture is a *Markov process*. The *Markov* assumption that future behaviour of the system is only dependent on its current state, not on its past history is a reasonable assumption for computer and communication systems, especially if one chose the states carefully.

Depending on the application, we can assign different meanings to states, state-transitions, rates, and rewards [HT93]. In our case, we use *Markov reward models* because they have the property of allowing us to assign values, such as transition rates and rate rewards, that we can associate with being in a particular state. What they do not provide us is the exact values of the energy units that would be consumed by a protocol state. Unfortunately, this information is hard to access in a real implementation. It does allow us to identify trends however as we shall show in **chapter** 6.

4.4.2 Validation of Assumptions

In order to validate the assumptions presented in the previous subsection, we conducted an interview [Mur03] with a protocol implementor from Siemens Communications South Africa. Our objective was to find out how easily accessible information such as transition rates between states in a protocol and amount of power consumed by the different states was during the implementation. The TC45 Siemens wireless engine [SIE03] was used to conduct this study[1].

4.4.3 Implementation Practices

During the protocol implementation process, the protocols are pre-compiled and tested on a standard compiler before they are loaded onto the TC45 module. Due to the large number of different applications that can be loaded onto the wireless module and varying communication conditions, there are no simulation tools that can profile the power consumption of the module. During the protocol implementation process, the protocol performance is application specific. What this implies is that each specific application implementation designed for the wireless module is entirely at the software implementor's discretion.

Performance analysis studies the software implementation on the TC45 are done by calculating the Java command execution throughput ("jPS"=Java statements per second). The scope of this measurement is only the statement execution time, not the execution delay (Java command on *Application Toolkit* (AT) interface $\rightarrow$ Java instruction execution $\rightarrow$ reaction on *General Purpose Input Output* (GPIO)). The following Java instruction is used for calculation of the typical jPS:

$$value = (2 \times number\ of\ calculation\ statements)/$$

[1]The information provided on the TC45 Siemens Cellular Engine has been used with permission of Kruger Murray Siemens Communications South Africa

$$((1/frequencyB) - (1/frequencyA));$$

In order to establish the hardware and protocol interaction, the frequency of the interaction can be probed by the the number of Java statements that call the AT hardware commands. This is a possible metric that can be used to track power consumption characteristics of the wireless module. Unfortunately, getting the actual power consumption measurements from the software protocols is not possible.

4.4.4 Profiling Power Efficient Implementations

At present, Siemens does not have equipment that determines the power consumption of individual protocol layer states. The main reason for this is that protocol implementors normally do not write application software which includes further software (instrumentation) which gathers the kind of information we need for the model. A example of this information is counting the number of transitions between states or the time spent in a state. This is no surprise as we mentioned in **section** 4.3.1 herein, of the difficulty of getting this type of empirical data.

Power efficient protocol implementation is a very important issue during the design of the TC45. However, more emphasis is put in saving power in the low processing modes i.e. idle mode, than in normal operation modes i.e. transmission. In addition, because of the unavailability of power consumption profiling tools, protocol implementors must rely on the efficiency of the protocols they implement. Furthermore, Siemens has placed a higher priority on improving bandwidth utilization, albeit minimal low power mechanisms are in place for normal communication modes. Indeed it would be beneficial if the protocol implementor was able to determine the power profile a particular implementation before loading it onto the wireless module. The only deterrent is that the number of wireless communication scenarios that have to be considered is quite large considering the different environmental conditions of the wireless neighborhood (i.e. signal strength, interference). Hence this provides a possible application of our methodology because it offers the flexibility of modelling specific communication scenarios that are of concern to the protocol implementor.

Chapter 5

Application

5.1 Introduction

In this chapter, we apply the methodology we proposed in **section** 1.2.1 of **chapter** 1 to analyze the power consumption of the wireless protocol architecture of the UMTS protocol stack. We derive the state description of the UMTS *Physical*, *Medium access control* and *Radio link control* protocol layers of the *User Equipment*. The *Markov* model is specified for each of these layers and the different power consumption experiments are conducted based on specific reward structures for the model. Subsequently, the Möbius experiment parameters are specified for each of the power consumption studies: *Power factor*, *Power level*, and *Power consumption comparison*.

5.2 Protocol States

In this section, we derive the state description of the UMTS *Physical* (PHY), *Medium Access Control* (MAC) and *Radio Link Control protocol*(RLC) layers of the *User Equipment* (UE). These protocols are specified by the *ETSI 3rd Generation Partnership Project*(3GPP) in the following documents: [vR02e] for the PHY, [vR02g] for the MAC, and [vR02h]. Additional information about the UMTS protocol architecture is provided by [vR02c],[vR02a], [vR02f] and [vR02d].

5.2.1 UE Protocol Architecture

The state diagrams in the subsequent subsections are based on the UE protocol architecture and the possible behaviour of the various protocol entities when radio frames are received or transmitted at the UE. Each protocol entity is described by the set of possible states in which it may be in and possible state transitions that may occur. In addition, a brief account of the various processes that take place once a protocol is in a certain state is given. This information is available in the protocol specification document of layer being modelled.

In **Figure** 16, we illustrate the protocol layer entities, from [vR02a], we are considering for analysis. Protocol entities considered for state model description include the PHY, MAC, and RLC protocols of the *user plane* for layer-to-layer communication. The user plane protocols are responsible for processing and transfer of the actual user application data, e.g. speech or video data, and file transfer data.

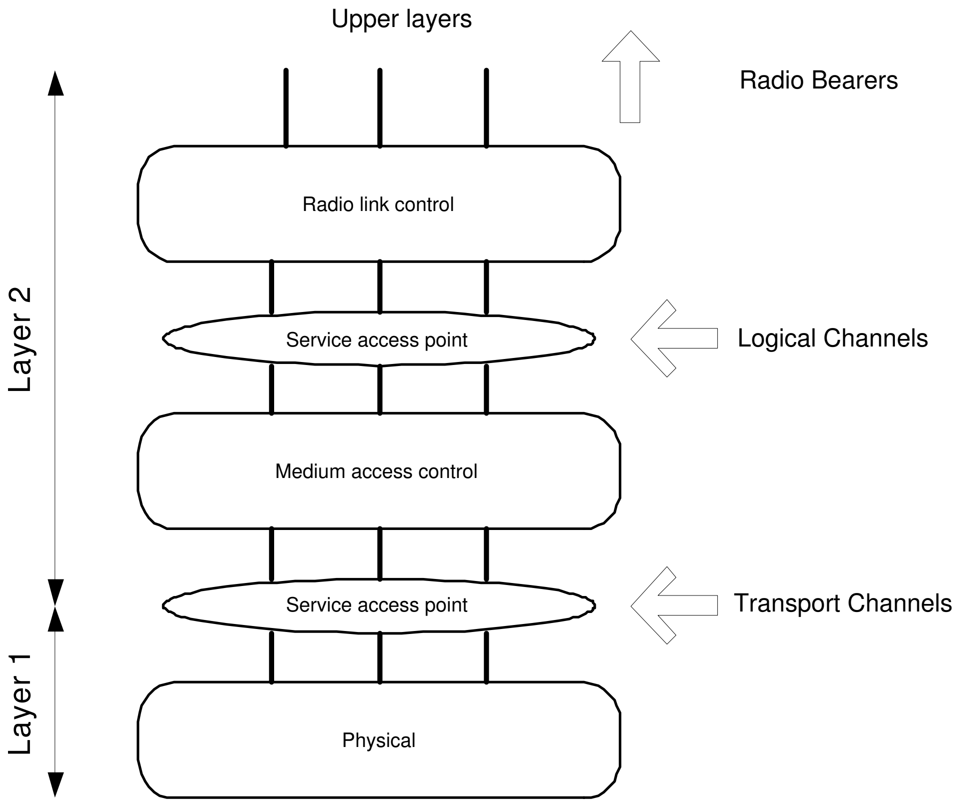


Figure 16: User Equipment User-Plane Protocol Architecture

In the next section, we discuss the various states of the Physical layer.

5.2.2 Physical Protocol States

Figure 17 derived from the [vR02e] specification document, illustrates the state model for the PHY protocol entity (both transmitting and receiving). The entity can be in one of the following states:

1. Null.
2. Channel Access Ready.
3. Data Transfer Ready.
4. Status Reporting.

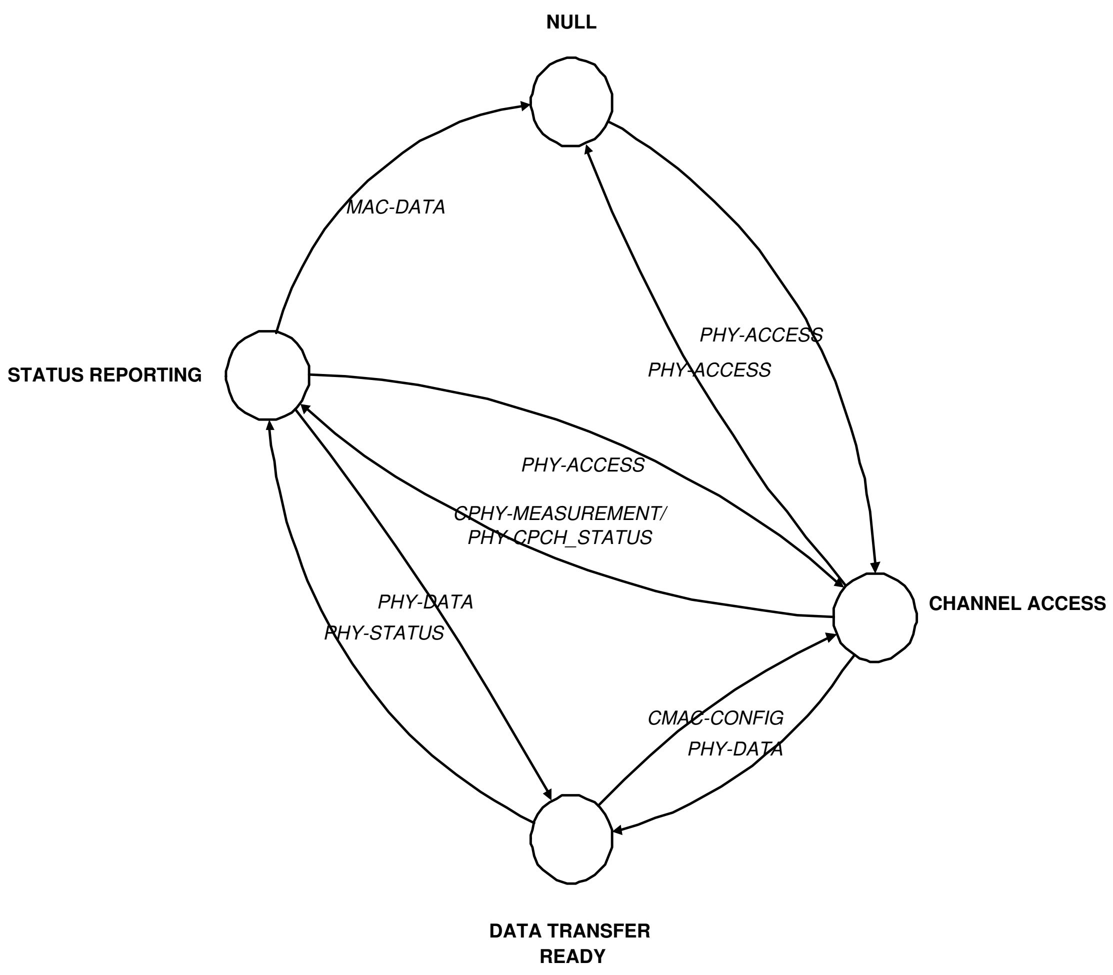


Figure 17: The State Model for the Physical Layer

Null State

In this state, there is no data flowing through the the PHY protocol entity. Upon reception of a request signal via the PHY-ACCESS, the PHY protocol entity is created and the protocol changes to Channel Access Ready state.

Channel Access Ready State

In this state, the protocol requests access to either a Random Access Channel (RACH) or a Common Physical Channel (CPCH). After a channel request is sent, a confirmation via

PHY-ACCESS is received with information on whether or not a channel has been assigned or not.

If a channel is assigned, the protocol changes to Data Ready state so that data can be transmitted. If the channel is not assigned, the protocol can wait for the next request signal via PHY-ACCESS or change state to Null.

Data Transfer Ready State

In this state, the protocol can transmit and receive PDUs between peer PHY entities.

Status Reporting State

The PHY protocol enters this state whenever the protocol is notifying higher layers of events occurring in the protocol. Some of these events include:

- PHY layer hardware failure;
- Start of message indicator received;
- End of message indicator received.

Once the report has been sent to the higher layer, the protocol returns to its previous state. In the event that the report is on failure of PHY layer hardware, the protocol changes to Null state.

In the next section, we discuss the various states of the *Medium access control* layer.

5.2.3 Medium Access Control Protocol States

Figure 18 derived from the [vR02g] specification document, illustrate the state model for the MAC protocol entity (both transmitting and receiving). The states for this entity include:

1. Null.
2. Data transfer ready.
3. Measurement.
4. Channel config.

5. Status reporting.

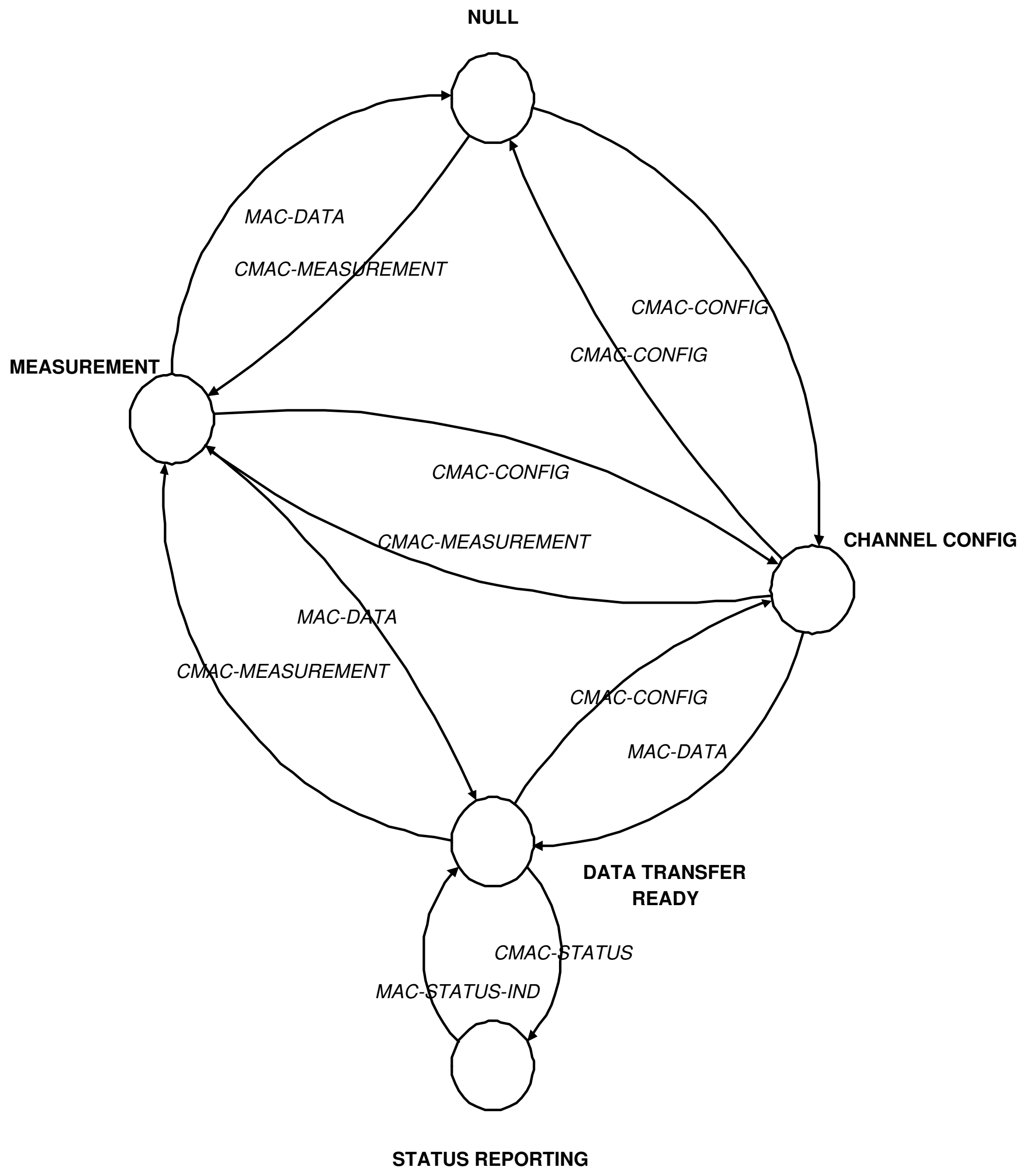


Figure 18: The State Model for the Medium Access Control Layer

Null State

In this state, the MAC entity inactive and therefore it is not possible to transfer any data. Once it receives request signal via the CMAC-CONFIG from the upper layer requesting a connection setup, it changes to Channel config state for transmission of data.

If this entity receives a request signal via the CMAC-MEASUREMENT with measurement information elements from the RRC, the entity changes to Measurement state.

Data Transfer Ready State

In this state, the entity can transfer MAC SDUs between peer MAC entities.

If the Radio Resource Control (RRC) requires configuration information from the MAC, a request signal via the CMAC-CONFIG is sent to the MAC entity with logical channel information. Upon receiving this request the MAC entity changes to Measure state.

If the entity receives a request signal via the CMAC-CONFIG with a connection release parameter, the entity changes to Null state. In this case the MAC entity no longer exists.

Measurement State

In this state, the entity receives a request signal via the CMAC-MEASURE from the RRC layer to perform measurements i.e. traffic volume measurements. The results from these measurements are reported to the RRC.

If the entity receives a request signal via the CMAC-CONFIG, it either remains in its present state or changes to Data ready state.

If the reported measurements to the RRC show that the traffic volume is not acceptable for transmission, the RRC sends a request signal via the CMAC-MEASURE to change state to Null. In this state the MAC entity does not exist and no data can be transferred.

Channel config

In this state, once the entity receives a request via the CMAC-CONFIG, a request to the higher layers is made in order to either setup, release or configure a logical channel.

Status reporting

In this state, the MAC notifies the RRC by sending an indication signal via the CMAC-STATUS primitive.

In the next section, we discuss the various states of the *Radio link control* layer.

5.2.4 Radio Link Control Protocol States

Figure 19 derived from the [vR02e] specification document, illustrates the state model for the acknowledged mode RLC protocol entity (both transmitting and receiving). The entity can be in one of the following states:

1. Null.
2. Data transfer ready.
3. Reset pending.
4. Local suspend.
5. Reset and Suspend.

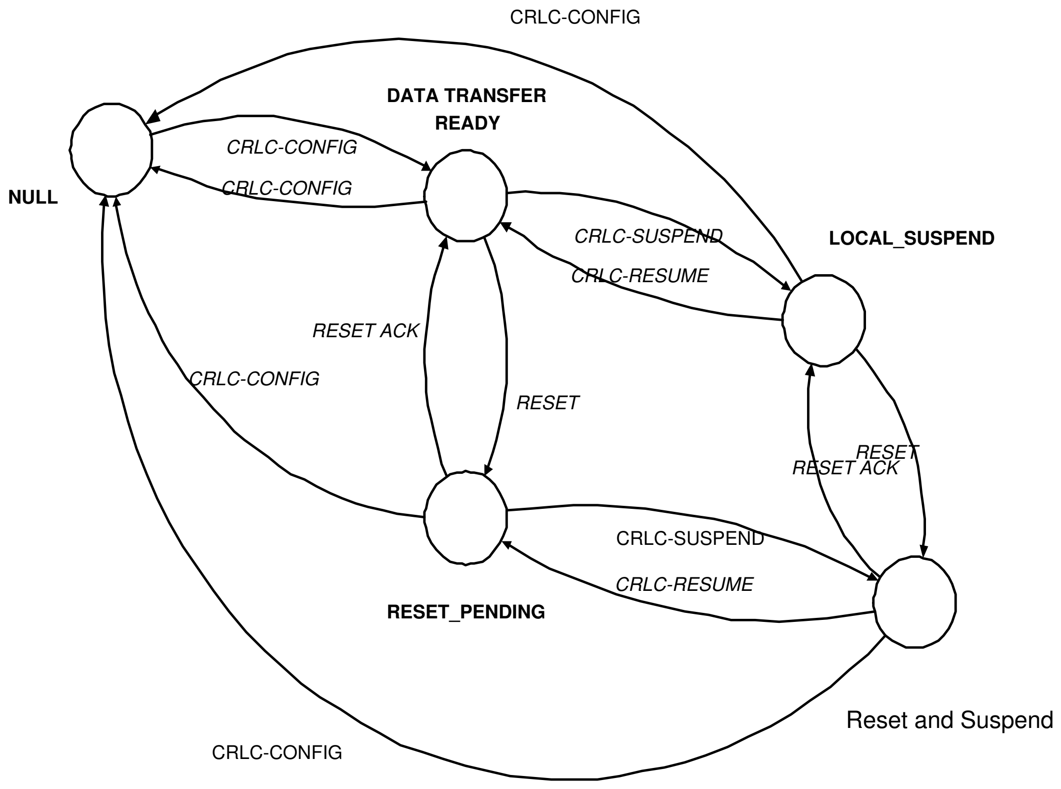


Figure 19: The State Model for the Radio Link Control Acknowledged Mode Entities

Null State

In this state, the RLC entity is inactive and therefore it is not possible to transfer any data. Upon receiving a request signal via the CRLC-CONFIG from the upper layer, the RLC entity is created and the protocol enters into Data transfer ready state.

Data transfer ready State

Once in this state, the acknowledged mode data (AMD) is prepared for transmission between RLC peer entities. This involves segmentation of the RLC SDUs into AMD PDUs, setting the length indicator field for the SDU, starting the timer discard for transmitted SDUs from the upper layer, and scheduling the AMD PDUs for transmission.

Upon reception of a request signal via the CRLC-CONFIG from the upper layer indicating release, the RLC entity enters into Null state and is considered as being terminated.

If an initiating condition for the RLC reset procedure is detected, the RLC entity initiates the RLC reset procedure and enters into the Reset pending state.

For the recept of a RESET ACK PDU, the RLC entity does not take any action. If it receives a request signal via the CRLC-SUSPEND from the upper layer, the RLC entity is suspended and enters into Local suspend state.

Reset Pending State

In this state, the entity waits for a response from its peer entity and no data can be exchanged between the entities. Upon recept of a request signal via the CRLC-CONFIG from the upper layer indicating release, the RLC entity enters into Null state and is considered as terminated.

If it receives a RESET ACK PDU it can either enter into Data transfer ready state or remain in its current state depending on the Radio Network Subsystem value (RNS) set in the PDU.

If a RESET PDU is received, the RLC entity remains in its current state.

If a request signal via the CRLC-SUSPEND from the upper layer is received, the RLC entity enters into Reset and suspend state.

Local Suspend State

In this state the RLC entity is suspended, i.e. it does not send AMD PDUs with Sequence Number greater than a specified value. Upon receiving a request signal via the CRLC-RESUME from the upper layer, the RLC entity resumes the data transmission and enters into the Data transfer ready state.

If it receives a request signal via the CRLC-CONFIG from the upper layer indicating release, the RLC entity enters the Null state and is considered as being terminated.

Upon detection of an initiating condition for the RLC reset procedure, the RLC entity initiates the RLC reset procedure and enters into Reset and suspend state.

Reset and Suspend State

In this state, the entity waits for a response from its peer entity or primitive (request signal via the CRLC-RESUME) from its upper layer and no data can be exchanged between the entities. Upon recept of a request signal via the CRLC-CONFIG from the upper layer is received, the RLC entity is terminated and the protocol enters into Null state. When a RESET ACK PDU is received, the peer RLC entities are reset and protocol enters into Local suspend state.

If a request signal via the CRLC-RESUME is received from the upper layer, the RLC entity is resumed, i.e. releases the suspend constraint; and the protocol enters into the Reset pending state.

In the next section, we discuss how we derive the *Markov* model from the state diagrams.

5.3 Markov Model

In this section, we show how the state description, given in **Section** 5.2.1, we have derived from the *formal specification* documents of UMTS is used to derive the *Markov* chains that capture the stochastic behaviour of the protocol layers considered thus far.

5.3.1 Markov Chain Derivation

In **Sections** 5.2.2-5.2.4 of this dissertation, we derived state descriptions that illustrate the behaviour of the PHY, MAC and RLC protocol layers. The state diagrams are used as the *continuous-time Markov chains* that closely capture the stochastic behaviour of these protocol layers. In addition, these state diagrams allow us to derive the generator matrices for the PHY, MAC, and RLC *Markov chains.* The generator matrices store the transition rates associated with state transitions of the *Markov* chains.

PHY Generator Matrix and State Legend

Below is the generator matrix of the PHY layer and the states represented.

$$Q_{PHY} = \begin{pmatrix} -\alpha_{2,1} & \alpha_{2,1} & 0 & 0 \\ \alpha_{1,2} & -(\alpha_{1,2}+\alpha_{3,2}+\alpha_{4,2}) & \alpha_{3,2} & \alpha_{4,2} \\ \alpha_{1,3} & \alpha_{2,3} & -(\alpha_{1,3}+\alpha_{2,3}+\alpha_{4,3}) & \alpha_{4,3} \\ 0 & \alpha_{2,4} & \alpha_{3,4} & -(\alpha_{2,4}+\alpha_{3,4}) \end{pmatrix}$$

State number	State name
1	Null
2	Channel access
3	Status reporting
4	Data Transfer Ready

MAC Generator Matrix and State Legend

Below is the generator matrix of the MAC layer and the states represented.

$$Q_{MAC} = \begin{pmatrix} -(\lambda_{2,1}+\lambda_{3,1}) & \lambda_{2,1} & 0 & \lambda_{3,1} & 0 \\ \lambda_{1,2} & -(\lambda_{1,2}+\lambda_{3,2}+\lambda_{4,2}) & \lambda_{3,2} & \lambda_{4,2} & 0 \\ 0 & \lambda_{2,3} & -\lambda_{2,3}+\lambda_{4,3}+\lambda_{5,3} & \lambda_{4,3} & \lambda_{5,3} \\ \lambda_{1,4} & \lambda_{2,4} & \lambda_{3,4} & -(\lambda_{1,4}+\lambda_{2,4}+\lambda_{3,4}) & 0 \\ 0 & 0 & \lambda_{3,5} & 0 & -\lambda_{3,5} \end{pmatrix}$$

State number	State name
1	Null
2	Channel Config
3	Data Transfer Ready
4	Measurement
5	Status Reporting

RLC Generator Matrix and State Legend

Below is the generator matrix of the RLC layer and the states represented.

$$Q_{RLC} = \begin{pmatrix} -\mu_{2,1} & \mu_{2,1} & 0 & 0 & 0 \\ \mu_{1,2} & -(\mu_{1,2}+\mu_{4,2}+\mu_{5,2}) & \mu_{4,2} & 0 & \mu_{5,2} \\ \mu_{1,3} & \mu_{2,3} & -(\mu_{1,3}+\mu_{2,3}+\mu_{4,3}+\mu_{5,3}) & \mu_{4,3} & \mu_{5,3} \\ \mu_{1,4} & 0 & \mu_{3,4} & -(\mu_{1,4}+\mu_{3,4}+\mu_{5,4}) & \mu_{5,4} \\ \mu_{1,5} & \mu_{2,5} & 0 & \mu_{4,5} & -(\mu_{1,5}+\mu_{2,5}+\mu_{4,5}) \end{pmatrix}$$

State number	State name
1	Null
2	Data Transfer Ready
3	Reset Pending
4	Reset and Suspend
5	Local Suspend

In the next section, we discuss the reward structures used for the experiments.

5.3.2 Reward Structure

Using *Markov reward models*, we can specify reward structures for the different states of the derived *Markov* chains. In our experiments, we investigate the following possible cases in which *Markov reward models* can be applied to analyze power consumption based on the reward definitions provided in **Section** 4.3.2 of **Chapter** 4:

Power factor: A factor that captures the predicted power consumption behaviour of a protocol state and protocol layer in time,

Power level: The power levels of different protocol states, based on how much reward (in this case power), the different protocol layers contribute (i.e. consume) to the overall accumulated reward in time, and

Power consumption comparison: The comparison of power consumption characteristics of individual protocol layers.

The next section discusses how the analytical model is solved using the Möbius modelling tool.

5.4 Analytic Model Solution

In **Sections** 5.3.1 and 5.3.2 of this chapter, we derived the *continuous-time Markov chain* of three radio interface protocols: *Physical*, *Medium Access Control* and *Radio Link Control*, and stated the different reward structures that will be used to model power consumption of the various states of the protocol respectively. In this section, we discuss how the *Accumulated reward solver* of Möbius is used to solve the *Markov reward model.*

5.4.1 Accumulated Reward solver

The Möbius *Accumulated reward solver*(ARS) solves for transient interval of time variables, i.e., for intervals $[t_0, t_1]$ where both t_0 and t_1 are finite [PER94]. It gives the expected accumulated reward, as well as the expected time-averaged accumulated reward over the interval.

From **Definition** 4.2 on page 44, we specify E_{AR} as the *accumulated reward* of the number of energy units, c_i, associated with the sojourn time in state i. We then use the Möbius *accumulated reward solver* to predict the following results over an interval of time:

1. *Power consumption in a protocol state i , and*

2. *Power consumption of the protocol layer.*

We will now distinguish between these two results in the following discussion. The power consumption of a state provides measures which characterize specified functions of a protocol. These measures include: (1) Information on which states consume the most power, and (2) Information on which states the protocol frequently spends time in. The results for power consumption of the protocol layer on the other hand provide an indication of relative power consumption associated with the separate layers of a protocol stack.

Based on the aforementioned premise, we can use the Möbius ARS to analyze the characteristics of power consumption in the protocol architecture that would be of interest to the protocol implementor (i.e. *Power Factor*, *Power Level*, and *Power Consumption Comparison analysis*)

Chapter 6

Möbius Experiment and Results

6.1 Introduction

In this chapter we discuss the Möbius modelling experiment, and the different parameters required to conduct a power characteristic analysis of the *Markov reward models* of *Radio link control*, *Medium access control*, and *Physical layers* of the UMTS radio interface protocol architecture using the Möbius *Accumulated Reward Solver* (ARS). Hereafter, we present the results of the power consumption experiments. Subsequently, we show the trends in the analytic results for *Power factor*, *Power Level*, and *Power consumption comparison* generated by the ARS.

6.2 Möbius Experiment

In the next section we discuss the Möbius experiment implementation and the different power characteristics we analyzed.

6.2.1 Implementation

We implemented our model in Möbius using the *Buckets and Balls* formalism [Ste94]. **Figure** 20 and **Table** 2 show an example of the *Atomic editor* implementation of the RLC and its *initial marking* attributes respectively. The atomic editor files of the modelled protocol layers are illustrated in **Appendix** herein. The timed transitions from state to state were set to exponential time distribution and rates specified from 0 to 1.

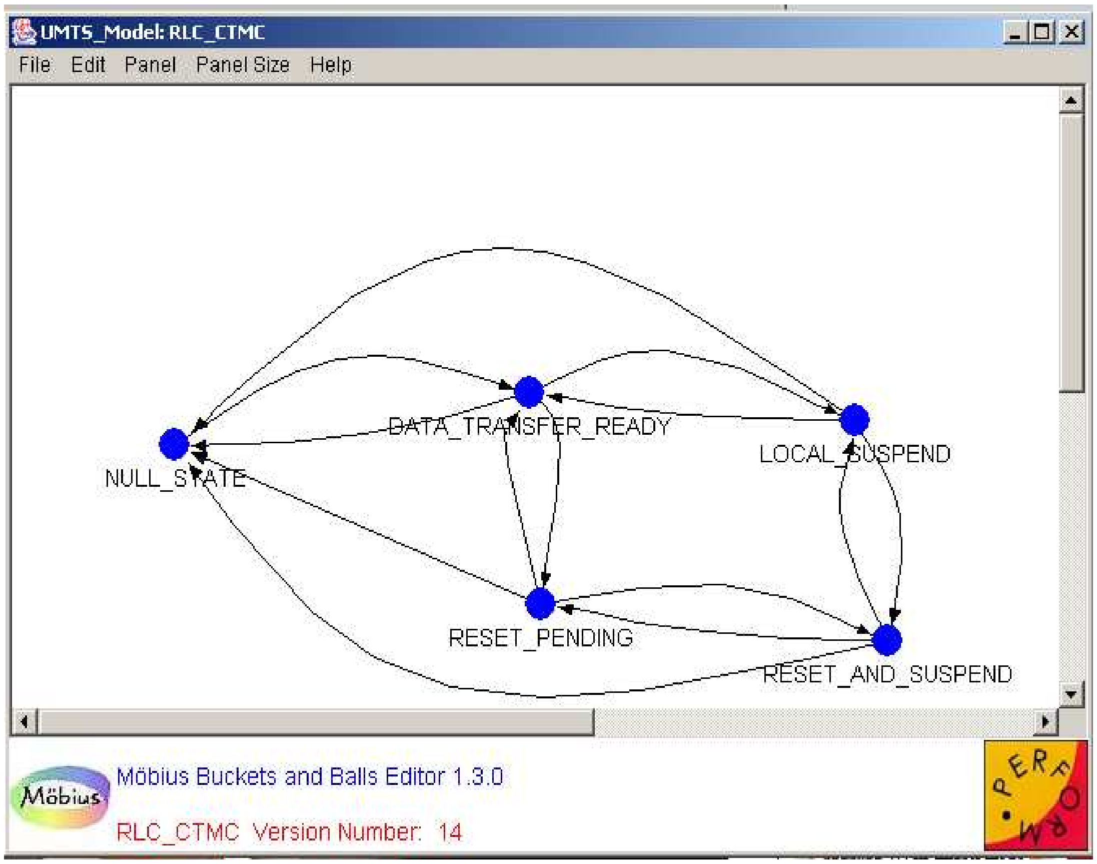

UMTS_Model: RLC_CTMC
File Edit Panel Panel Size Help
DATA_TRANSFER_READY
NULL_STATE
LOCAL_SUSPEND
RESET_PENDING
RESET_AND_SUSPEND
Möbius Buckets and Balls Editor 1.3.0
RLC_CTMC Version Number: 14

Figure 20: Screen shot of Möbius *Buckets and Balls* representation of the RLC layer

Name	Initial Marking
DATA_TRANSFER_READY	0
NULL_STATE	1
LOCAL_SUSPEND	0
RESET_PENDING	0
RESET_AND_SUSPEND	0

Table 2: Bucket attributes for the radio link control layer

Rewards were then specified in the *Reward editor*. For this experiment, each reward defined had a number of performance variables specified depending on how many states the layer has. The value of the reward for individual performance variables was determined by the type of study being conducted: *Power factor*, or *Power level*, or *Power consumption comparison*. Each performance variable had the following generic syntax for the reward function depending on the atomic model:

```
if( RLC_CTMC → NULL_STATE → Mark()){ return 1; }
```

This C++ statement means that if the "NULL_STATE" state of the RLC_CTMC atomic model is marked, then the function returns the value 1.

To solve for the specified *performability* variable, the state space and state transition matrix to be used for analytic evaluation are generated by the *Study* and *Solver* editors. Using the *Numerical* editor, we selected the *accumulated reward solver*(ARS) and specified the time interval for the experiment. In addition, the *Möbius simulator* was used to generate the distribution of the the accumulated reward in order to capture the fraction of time the model is in a state.

6.2.2 Power Factor Analysis

For the *Power factor* analysis experiment, we specify two cases in order to observe how the power accumulation can vary during different transmission condition. In *Case 1*, we model a transmission condition, where the protocol spends more time performing measurements on the wireless channel. What this implies is that the expected transition rates into all states associated with channel condition measurements have higher values compared to other states. On the other hand, *Case 2* models the a situation where the protocol spends more time in data transfer states during transmission. Ultimately, we would compare the power factor associated with each of the states under these two conditions.

Transition Rates

Tables 3, 4 and 5 show the source and destination states associated with the RLC, MAC and PHY layer respectively. In addition, they show the exponential distribution rates for the two case experiments. Due to the fact that actual state rates are difficult to get, as explained in section 4.4 of chapter 4, the rates used for the following studies are intuitive assumptions based on the type and number of protocol data units served to and by a protocol layer.

Source State	**Destination State**	**Rates**	
		Case 1	Case 2
NULL_STATE	DATA_TRANSFER_READY	0.3	1
LOCAL_SUSPEND	DATA_TRANSFER_READY	0.3	1
RESET_PENDING	DATA_TRANSFER_READY	0.3	1
DATA_TRANSFER_READY	NULL_STATE	0.2	0.2
RESET_AND_SUSPEND	NULL_STATE	0.01	0.01
LOCAL_SUSPEND	NULL_STATE	0.01	0.01
RESET_PENDING	NULL_STATE	0.01	0.01
DATA_TRANSFER_READY	LOCAL_SUSPEND	1	1
RESET_AND_SUSPEND	LOCAL_SUSPEND	1	0.4
DATA_TRANSFER_READY	RESET_PENDING	0.4	0.2
RESET_AND_SUSPEND	RESET_PENDING	0.4	0.2
RESET_PENDING	RESET_AND_SUSPEND	0.8	0.2
LOCAL_SUSPEND	RESET_AND_SUSPEND	0.8	0.2

Table 3: Activity case transition rates between radio link control states

Source State	**Destination State**	**Rates**	
		Case 1	Case 2
NULL_STATE	CHANNEL_CONFIG	0.5	0.2
MEASUREMENT	CHANNEL_CONFIG	0.5	0.2
DATA_TRANSFER_READY	CHANNEL_CONFIG	0.5	0.2
CHANNEL_CONFIG	NULL_STATE	0.01	0.01
MEASUREMENT	NULL_STATE	0.01	0.01
NULL_STATE	MEASUREMENT	1	0.2
CHANNEL_CONFIG	MEASUREMENT	1	0.2
DATA_TRANSFER_READY	MEASUREMENT	1	0.2
CHANNEL_CONFIG	DATA_TRANSFER_READY	0.2	1
MEASUREMENT	DATA_TRANSFER_READY	0.2	1
STATUS_REPORTING	DATA_TRANSFER_READY	0.2	1
DATA_TRANSFER_READY	STATUS_REPORTING	0.6	0.2

Table 4: Activity case transition rates between medium access control states

Source State	**Destination State**	**Rates**	
		Case 1	Case 2
CHANNEL_ACCESS	NULL_STATE	0.01	0.01
STATUS_REPORTING	NULL_STATE	0.01	0.01
NULL_STATE	CHANNEL_ACCESS	0.5	0.4
DATA_TRANSFER_READY	CHANNEL_ACCESS	0.2	0.2
STATUS_REPORTING	CHANNEL_ACCESS	0.6	0.4
CHANNEL_ACCESS	DATA_TRANSFER_READY	0.2	1
STATUS_REPORTING	DATA_TRANSFER_READY	0.2	1
CHANNEL_ACCESS	STATUS_REPORTING	1	0.2
DATA_TRANSFER_READY	STATUS_REPORTING	1	1

Table 5: Activity case transition rates between physical layer states

6.2.3 Power Level Analysis

For the *Power Level Analysis* we specify reward structures based on the *Markov reward model* property of *utilization.* This MRM generates the state-based measure that corresponds to the amount of reward the model contributes to the overall reward in time. This allows us to understand the *power utilization* of each protocol state of a particular layer.

In order to analyze the protocol state utilization, we specified the following as the reward structure for the model: for each state $i \in S$, we set the power reward $c_i = 1$ while $c_j = 0, \forall j \neq i$. A *Markov reward model* of this type is known *availability model* [STR88]. For our analysis though, this provides an aid to understanding the *power utilization* of each protocol state. In addition, it allows us to determine which states would consume the most power for a particular protocol implementation

6.2.4 Power Consumption Comparison Analysis

For the *Power consumption comparison* analysis we specify reward structures for each of the layers that allows us to compare the accumulated power consumption behaviour of each protocol layer. We compare the accumulated amount of reward in time based on the premise that each protocol layer is expected to accumulate a certain fraction of the overall power consumed. For a particular protocol implementation, this allows the protocol implementor to predict which protocol layers consume the most energy during normal operation. Table 6 shows the estimated contribution of reward that each protocol would make for the *Physical*, *Medium access control* and *Radio link control* layers. These parameters are based on the intuitive assumption of the expected consumption associated with being in a particular state. In practice, these parameters would be determined by the *protocol implementor* based on the desired layered architecture [Svo89].

Protocol State	**State Rewards**		
	Physical	*Medium access control*	*Radio link control*
1	0.01	0.01	0.01
2	0.9	0.9	1
3	1	1	0.5
4	0.5	0.5	0.8
5	$\star$	0.4	0.8

Table 6: State rewards for power consumption comparison analysis

6.3 Results

In this section, we present the results of the Möbius experiments. The Möbius accumulated reward solver generated both the expected accumulated reward and the time-averaged accumulated reward [PER94]. We will confine ourselves to the latter result though we will present one graph of the former as an example in the RLC layer.

6.3.1 Power Factor Analysis Results

Figures 21, 22, 23, and 24 illustrate the results of our power factor experiment for the RLC, MAC and PHY protocol layers. The trends in the results for each layer are discussed in turn in the discussions hereafter.

RLC Layer

Figures 21 and 22 below illustrate the *expected* and *time-averaged accumulated reward* results of the RLC layer respectively. They compare the power consumption characteristics when more time is spent performing measurements and data transfer, in *Case 1* and *Case 2* respectively. These graphs show the initial transient behaviour and the subsequent steady-state behavior. Results of this sort can be very useful to the protocol implementor in understanding and visualizing the expected power consumption of the protocols under different operational conditions.

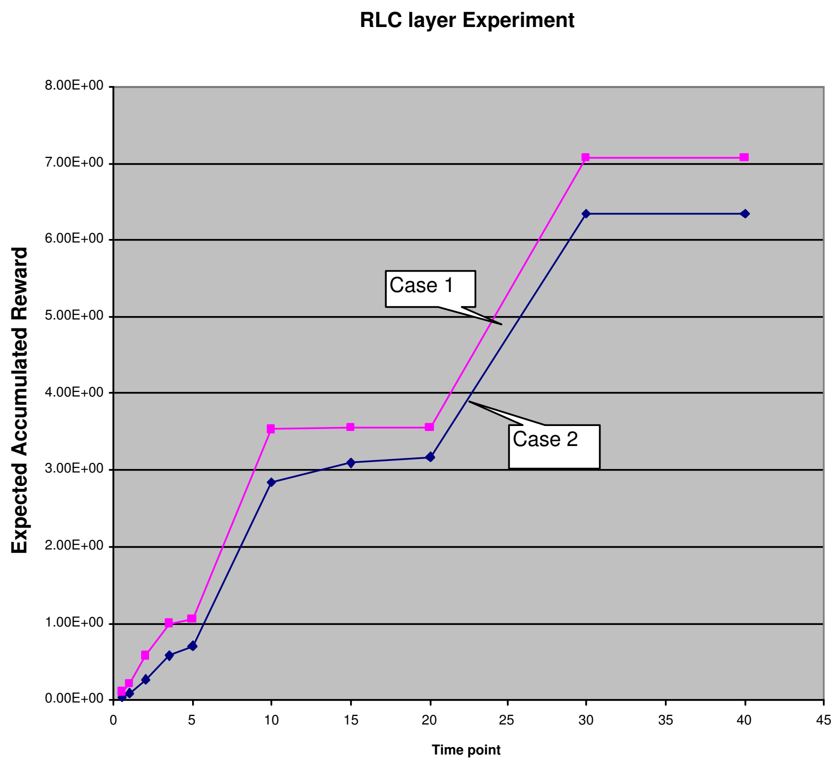


Figure 21: RLC Layer: *Expected Accumulation*

The expected accumulation results compare the expected accumulation of reward in time of the two experiment cases. It shows that for the parameter values assessed, the protocol layers consume more energy when they are performing measurements than when they are transmitting data.

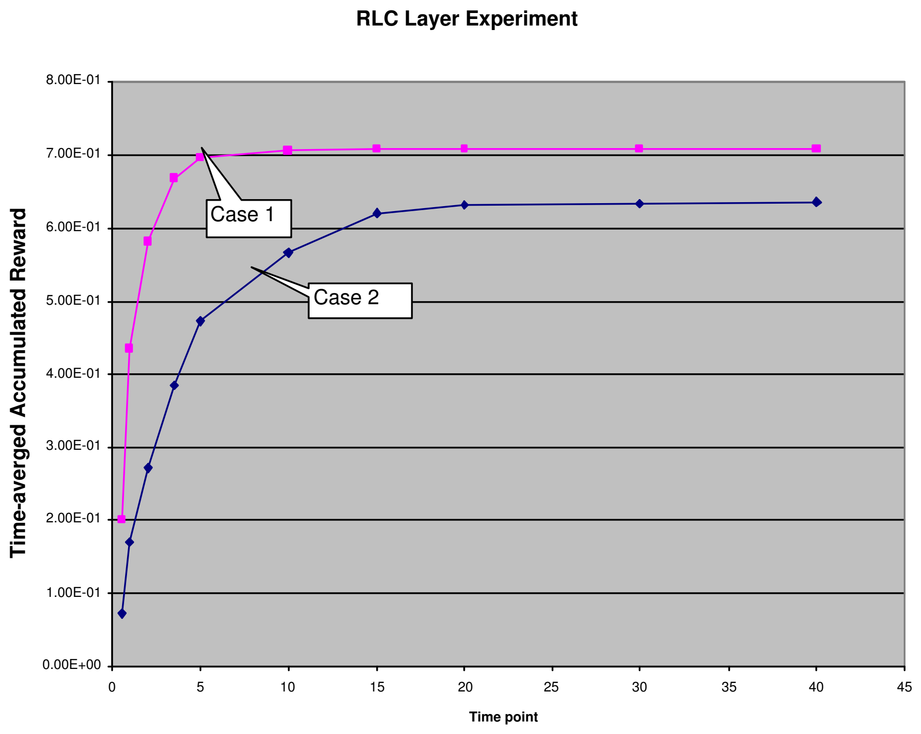


Figure 22: RLC Layer: *Time-Averaged Accumulation*

The time-averaged accumulation results show that after a certain amount of time, the consumption stabilized for both cases. This indicates that the protocol layer has entered into a steady state. Using this result, we are able to determine which case consumes the most and the least amount of power.

MAC Layer

Similar to the time-averaged accumulated reward results presented in the previous subsection, the MAC layer results are illustrated below. We are able to distinguish between the two cases where the protocol spends more time either performing measurements or performing data transfers. For this experiment, we observe that *Case 1* is more power intensive than *Case 2* in both transient and steady state.

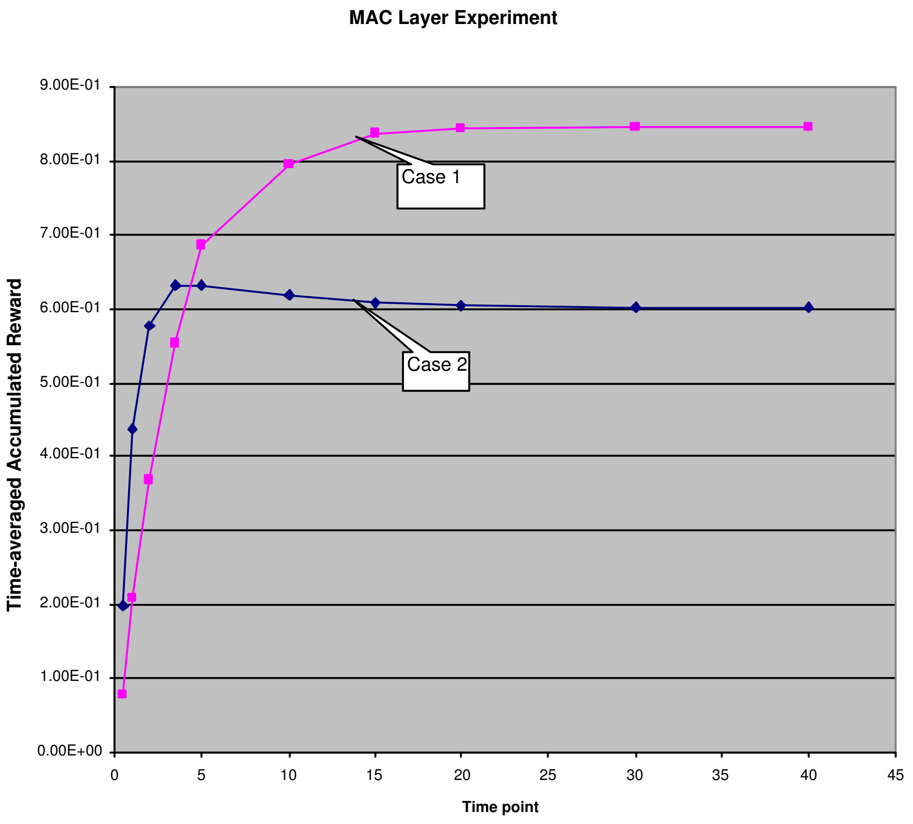


Figure 23: MAC Layer: *Time-Averaged Accumulation*

PHY Layer

Analogous to the results presented in the previous subsection, the time-averaged accumulated reward results of the PHY layer are illustrated below. It also shows the compared results of *Case 1* and *Case 2*, with the former case consuming more power than the latter. In addition, during the transient state, the rate of power consumption is similar for the two cases. However, there is a distinct difference once the protocol layer enters steady state.

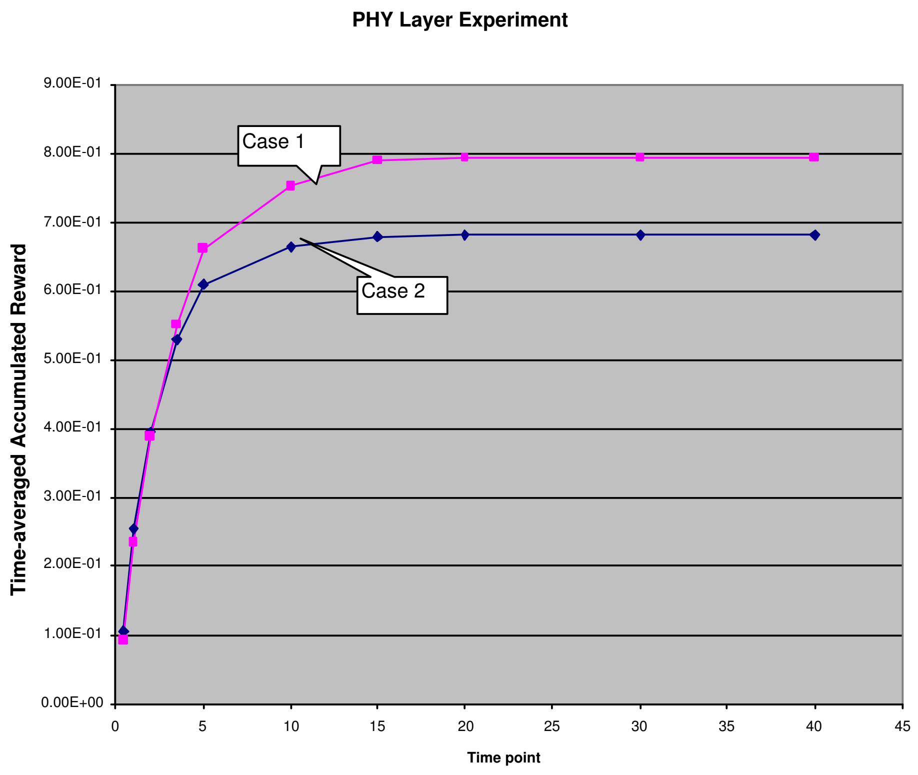


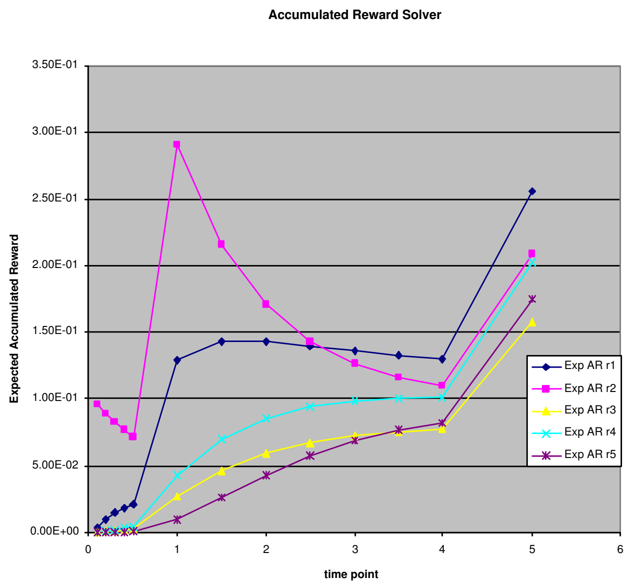
Figure 24: PHY Layer: *Time-Averaged Accumulation*

6.3.2 Power Level Analysis Results

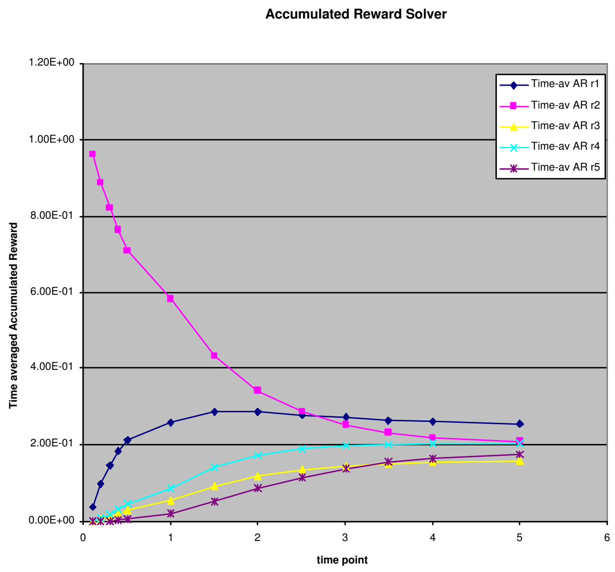
Figures 25, 26, and 27 illustrate the results of our power factor experiment for the RLC, MAC and PHY protocol layers. For this experiment, we specified 1 as the reward in each run of the accumulated reward solver. The trends in the results for each layer are discussed in turn in the discussions hereafter.

RLC Layer

The results presented here compare how much power reward each state of the RLC layer protocol would contribute over a period of time. Figure 25 shows the combined expected accumulated reward and time-averaged accumulated reward results.



(a) *Expected accumulated reward*



(b) *Time-averaged accumulated reward*

Figure 25: Accumulated reward solver results for the radio link control layer over an interval-of-time.

These results show us which states (denoted by *Exp AR*[1] r1→ *r*N) are expected to consume the most amount of power. In addition, the results illustrate the expected power consumption behaviour associated with each protocol state over a period of time. This is determined by the frequency with which are state is visited. Analogous to the *power factor* experiment, these results are largely dependant on the transition rates between the various states. For example, we can observe from **Figure** 25(b) that the state that returns reward $r1$ is more likely to be the most frequently visited state. Results of this sort can be very useful to the protocol implementor in determining which of the protocol states are power intensive based on a particular protocol layer implementation. At this stage, the implementor has an opportunity to address the implementation overhead caused by power intensive protocol states before the final protocol program code is uploaded on the wireless device.

MAC Layer

Below is an illustration of the results of power level analysis experiment of the MAC layer.

[1]Exp AR denotes the Expected Accumulated Reward.

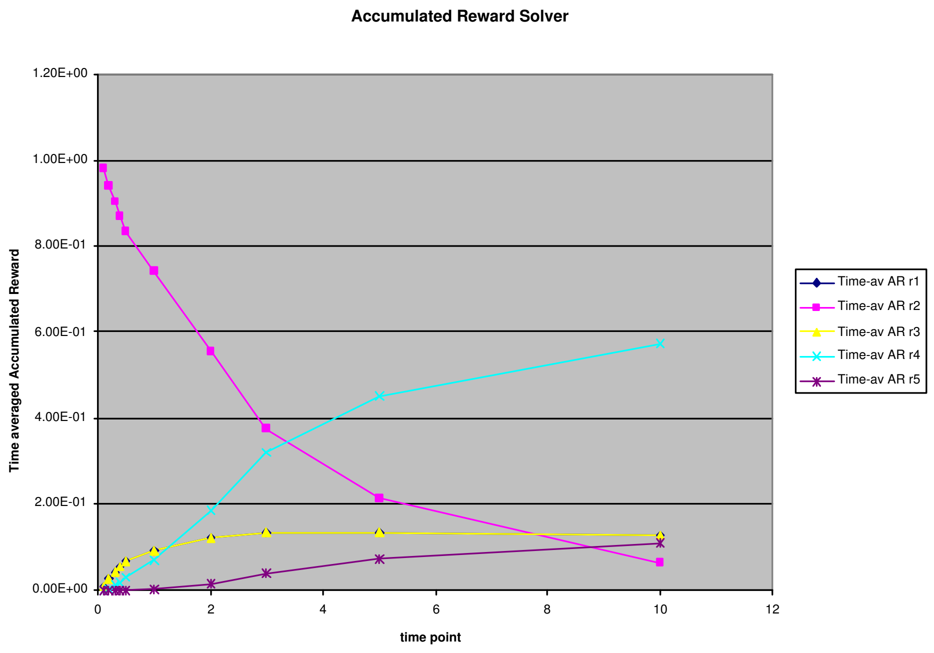


Figure 26: MAC Layer: *Time-averaged accumulated reward*

Analogous to the RLC layer experiment, we are able to determine the power consumption contribution each of the protocol layers states would make and which states have a higher frequency of visits. In **Figure** 26, we observe that the reward $r5$ signifies the state that has the most visits once the protocol layer has been in steady state for some time.

PHY Layer

Below is an illustration of the results of power level analysis experiment of the PHY layer.

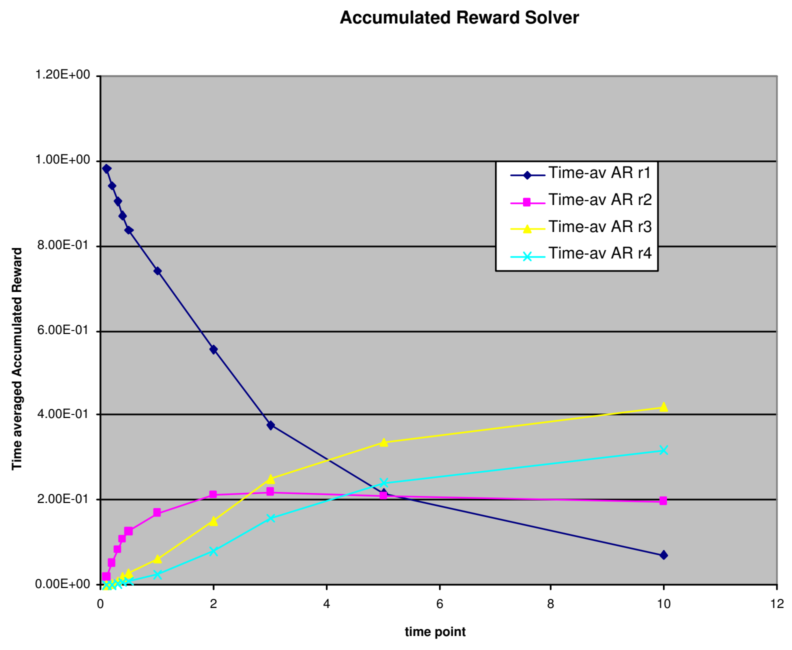


Figure 27: PHY Layer: *Time-averaged accumulated reward*

For this protocol layer, the state that returns reward $r3$ is the most visited state.

6.3.3 Power Consumption Comparison Analysis Results

Figure 28 shows the power consumption comparison of specific protocol implementations of the RLC, MAC, and PHY layers. The graphs illustrate how a protocol implementor can determine and compare expected power consumption characteristics of part or the entire protocol architecture implementation. It provides insight into which protocol layers consume the most and the least amount of energy when the mobile device is in a steady state. In this particular case, the MAC layer is expected to consume the most, and the RLC layer consumes the least amount of power for this particular protocol implementation.

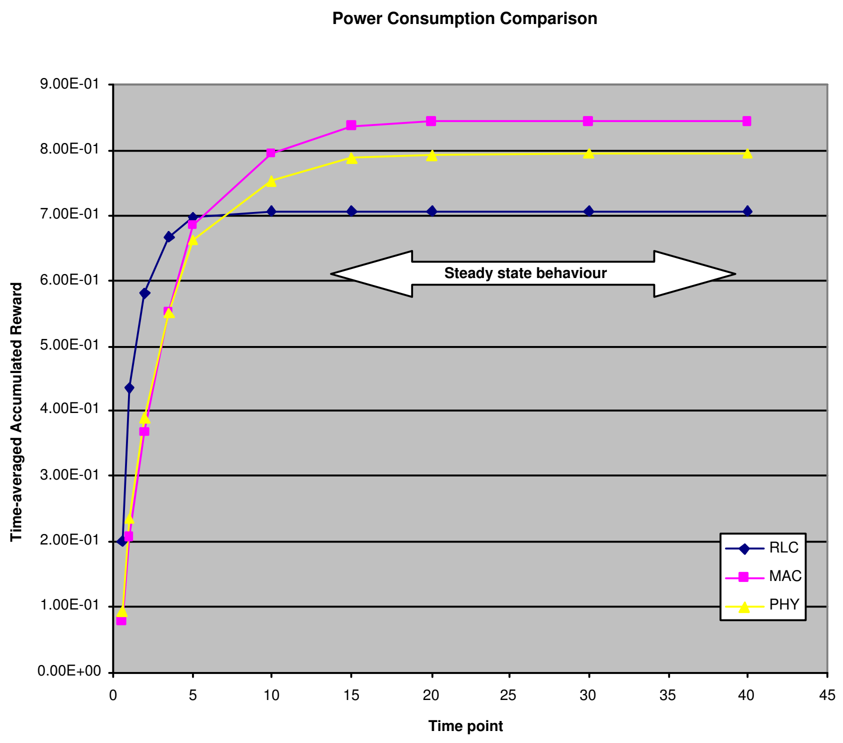


Figure 28: Compared power consumption of individual protocol layer implementations

Chapter 7

Conclusion

7.1 Summary

This dissertation has investigated a methodology that incorporates the *formal specification* of radio interface protocols for the purpose of analyzing the power consumption of wireless communication equipment. Similar to predicting protocol performance from meta-implementation, power consumption can be an additional concept to the whole process of implementing power-efficient wireless protocols. By deriving the state-transition graphs of selected radio interface protocols, we can use *Markov* models that allow us to track the evolution of the protocol during normal operation. From the *Markov* model, power consumption can be modelled as a reward by way of *Markov Reward Models.* For each protocol state, power can be represented as the reward obtained per unit time for being in that state.

We investigated techniques that took advantage of some of the properties of *Markov Reward Models* (i.e. steady state, accumulation, and availability), which allow us to assign weights that are analogous to the power consumed by the different states of the protocol. From these models, we were able to analyze the trends in power consumption. The analytic results were categorized as the **Power factor**, **Power Level**, and the **comparison** of the **Power consumption** of different layers. As an example of implementation, we applied our methodology to the *Universal Mobile Telecommunications System* (UMTS) radio interface protocols and presented some results.

Finally, we interviewed a *Protocol Implementor* from Siemens AG South Africa in order to share our ideas and gain insight into general practices involved in implementing power aware software protocols in wireless communication devices. Though power consumption is a concern, profiling protocol power consumption has not been addressed. This is because of the diverse nature of wireless communication environments that influence power consumption behavior. We believe this is an important motivation of using our proposed methodology because it allows the protocol implementor to decide what wireless communication scenarios he or she would like to model.

The implementation of the methodology suffers in that it is difficult to get actual data that represents the actual protocol power consumption. For this we can only make estimates based on the power consumption at the network interface. The methodology relies, in part, on the access to information at the lower levels of protocol software to determine, for example, the rates of transition between states. We know from the aforementioned interviews that such information can be generated with the help of special performance modelling tools that can measure the frequency with which certain commands are called. Using this information, we can assign rates to the *Markov* models, which allows us to closely capture what the real world power consumption characteristics would be. We discovered that getting the actual power consumption values of a protocol layer states was very difficult to acquire. The interviews confirmed the difficulty and attributed this to the unavailability of protocol layer state profiling tools. In addition, they are not documents that contain the power consumption values of individual states of a protocol layer. This concern can be addressed by considering the amount of power consumed at the radio interface and using it to make estimates of power consumed by the protocol architecture.

We envisage the application of this methodology not only in infrastructure type networks but also in ad hoc and other network topologies. In addition, predicting a particular protocol implementations power consumption would allow the implementor an opportunity to address power consumption issues at the design stage.

7.2 Future Work

The area of energy consumption in wireless communication devices at a protocol level is relatively new. According to Rao and Redi in [PRR01] there is a lot of research in this area because energy consumption is an important metric which has to be taken into account to increase the mobile systems usability and functionality.

A possible area of research our methodology can be applied is in verifying the work by Min and Chandrakasan in [MC03]. In their research with node designs for high-density wireless networks they reveal that the abstraction in software protocol layers gets in the way of energy saving. They further state that application designers have in the past been reluctant to manipulate values of the processor voltages or transmit power. They suggest introducing proper abstractions between communication software and hardware because it can encourage, and not hinder, energy savings. Because our methodology is based on a similar premise, their work offers an opportunity for applying our proposed methodology.

Appendix A

Möbius Editor Model Files

A.1 Atomic Editor Files

The following illustrations are the atomic model files for the UMTS protocol layers we modelled: *Physical*, *Medium Access Control*, and *Radio link Control* layers. The *Radio link Control* layer screen shot shows the additional editors that allow the modeler to to specify the type of distribution for the transitions and their respective rates.

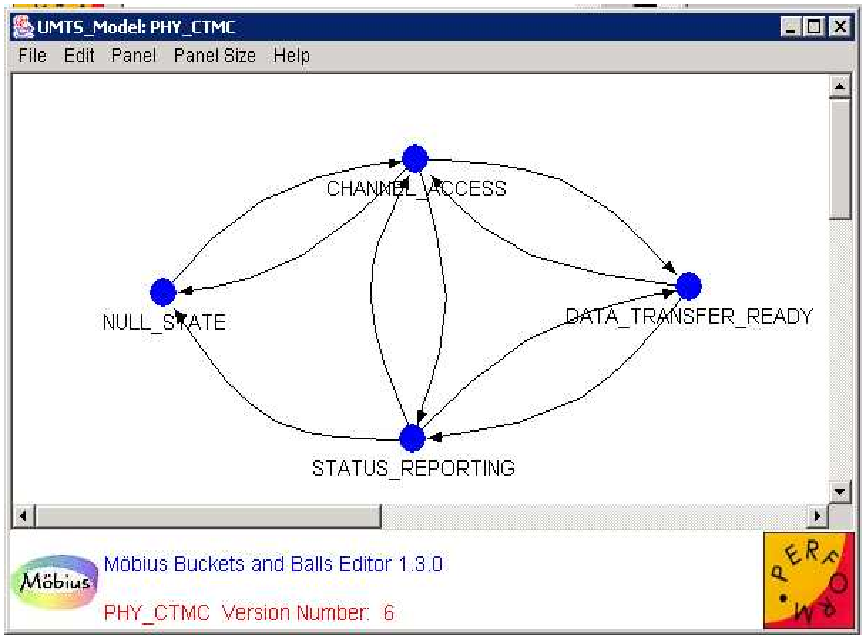


Figure 29: Buckets and Balls atomic editor of the Physical layer

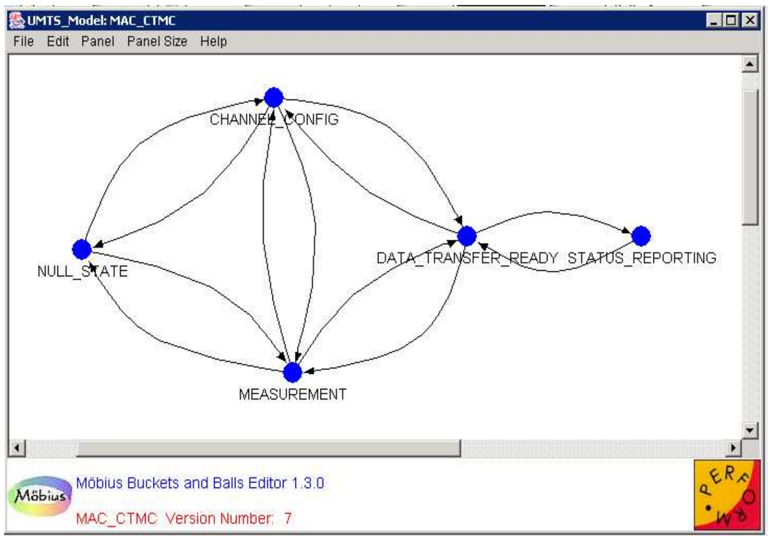
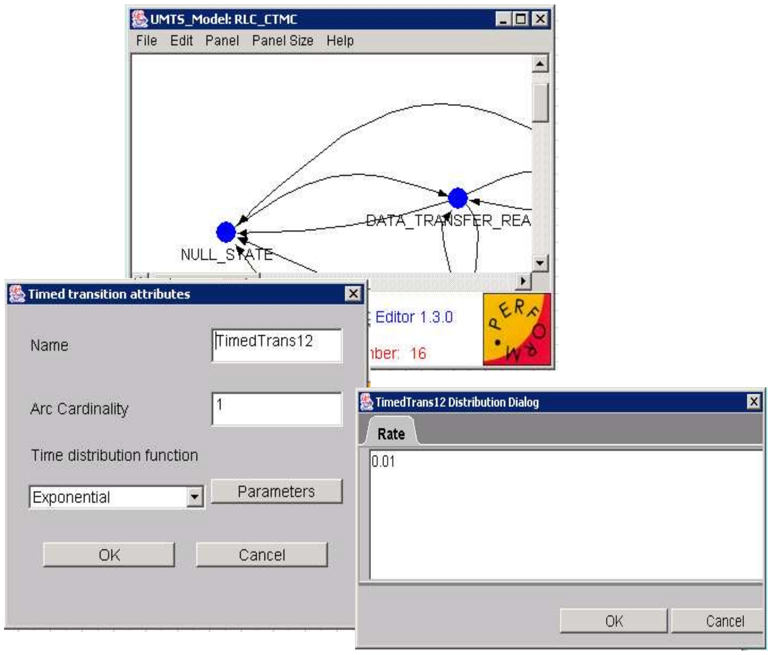

UMTS_Model: MAC_CTMC
File Edit Panel Panel Size Help
CHANNEL_CONFIG
NULL_STATE
DATA_TRANSFER_READY
STATUS_REPORTING
MEASUREMENT
Möbius Buckets and Balls Editor 1.3.0
MAC_CTMC Version Number: 7

Figure 30: Buckets and Balls atomic editor of the Medium Access Control layer



Name: TimedTrans12

Arc Cardinality: 1

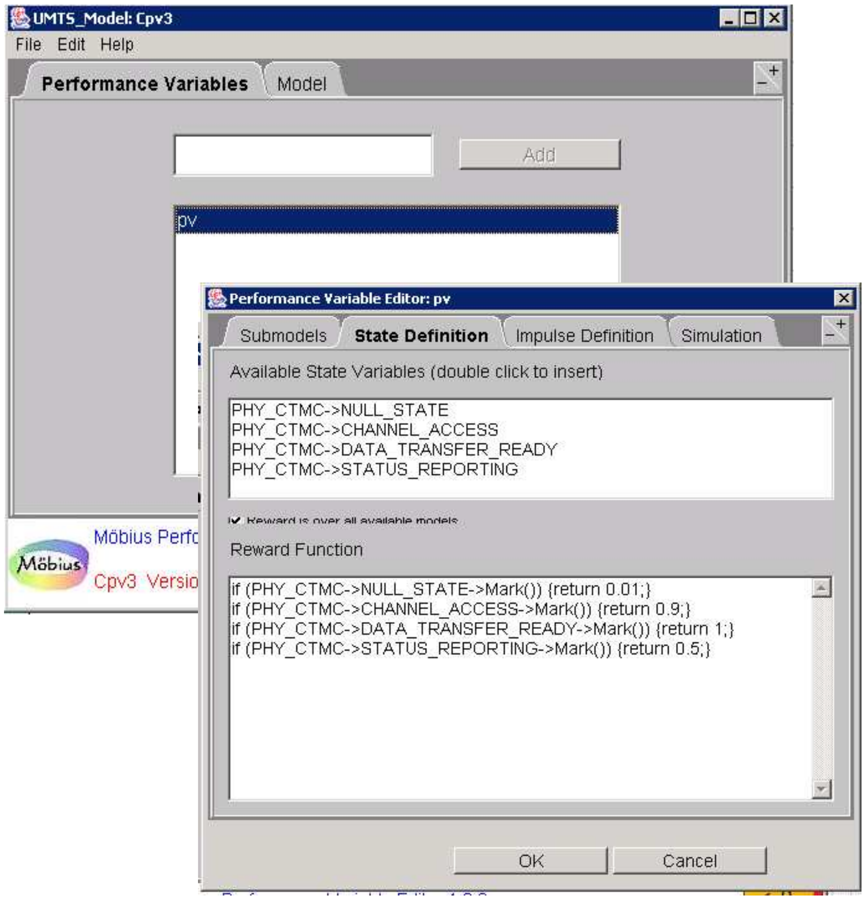
Time distribution function: Exponential

Rate: 0.01

Figure 31: Buckets and Balls atomic editor of the Radio Link Control layer with Transition Distribution and Rate editors

A.2 Reward Editor Files

The following illustration is an example of one of the reward editor file for the performance variable specified for the UMTS protocol layers. They include examples of how state reward variables are defined in the Möbius tool.



```
PHY_CTMC->NULL_STATE
PHY_CTMC->CHANNEL_ACCESS
PHY_CTMC->DATA_TRANSFER_READY
PHY_CTMC->STATUS_REPORTING
```

```
if (PHY_CTMC->NULL_STATE->Mark()) {return 0.01;}
if (PHY_CTMC->CHANNEL_ACCESS->Mark()) {return 0.9;}
if (PHY_CTMC->DATA_TRANSFER_READY->Mark()) {return 1;}
if (PHY_CTMC->STATUS_REPORTING->Mark()) {return 0.5;}
```

Figure 32: Reward editor for the Physical layer

Bibliography

[BD87] S. Budkowski and P. Dembinski. An introduction to Estelle: A specification language for distributed systems. *Computer Networks and ISDN Systems*, (14):3–23, 1987.

[BS80] Gregor V. Bochmann and Carl A. Sunshine. Formal Methods in Communication Protocol Design. *IEEE Transactions on Communications*, COM-28(4):624–631, April 1980.

[CCD$^+$01] G. Clark, T. Courtney, D. Daly, D. Deavours, S. Derisavi, J. M. Doyle, W. H. Sanders, and P. Webster. The Möbius Modelling Tool. Proceedings of the 9th International Workshop on Petri Nets and Performance Models, September 11-14 2001.

[CMST88] Gianfranco Ciardo, Raymond Marie, Bruno Sericola, and Kishor Trivedi. Performability Analysis Using Semi-Markov Reward Processes. Technical Report DUKE–TR–1988–09, 1988.

[CSAK98] Jyh-Chen Chen, Krishna M. Sivalingam, Prathima Agrawal, and Shalinee Kishore. A Comparison of MAC Protocols for Wireless Local Networks Based on Battery Power Consumption. *IEEE: INFOCOM*, pages 150–157, 1998.

[CZ98] A. Chockalingam and Michele Zorzi. Energy Efficiency of Media Access Protocols for Mobile Data Networks. *IEEE Transactions on Communications*, 46(11):1418–1421, November 1998.

[DS01] D. D. Deavours and W. H. Sanders. Mobius: Framework and Atomic Models. Proceedings of the 9th International Workshop on Petri Nets and Performance Models, Aachen, Germany, September 11-14 2001.

[Eph02] Anthony Ephremides. Energy Concerns in Wireless Networks. *IEEE Wireless Communications*, 9(4):48–59, August 2002.

[Fee99] Laura Marie Feeney. Investigating the Energy Consumption of an IEEE 802.11 Network Interface. WWW page http://citeseer.nj.nec.com/325357.html, December 1999.

[FN01] Laura Marie Feeney and Martin Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. *IEEE INFOCOM 2001 - The Conference on Computer Communications*, 0(1):1548–1557, April 2001.

[G. 90] G. Ciardo and R. Marie and B. Sericola and K. S. Trivedi. Performability Analysis Using Semi-Markov Reward Processes. *IEEE Transactions on Computers*, C-39(10):1251–1264, October 1990.

[GPT00] Katerina Goseva-Popstojanova and Kishor S. Trivedi. Stochastic Modelling Formalisms for Dependability, Performance and Performability. In *Performance Evaluation*, pages 403–422, 2000.

[Gra02] Ing. Wolfgang Granzow. 3rd Generation Mobile Communications Systems. Ericsson Eurolab Deutschland Research Mobile Communications, Neumeyerstr 50, 90411 Nurnberg, 2002.

[HBS94] D. Hogrefe, F. Belina, and A. Sarma. *SDL with applications from protocol specification.* Prentice Hall International, 1994.

[Hil03] Jane Hillston. Constructing and Solving Markov Processes. WWW page http://www.dcs.ed.ac.uk/teaching/cs4/www/modsim/notes/note3.ps Accessed July, 2003.

[HS01] Paul Havinga and Gerard Smit. Energy-efficient Wireless Networking for Multimedia Application. *Wireless Communications and Mobile Computing*, 1:165–184, 2001.

[HT93] Boudewijn R. Haverkort and Kishor S. Trivedi. Specification Techniques for Markov Reward Models. *Discrete-Event Dynamic Systems: Theory and Applications*, 0(3):219–247, 1993.

[JSAC01] Christine E. Jones, Krishna M. Sivalingam, Prathima Agrawal, and Jyh-Cheng Chen. A Survey of Energy Efficient Network Protocols for Wireless Networks. *Wireless Networks*, 7(4):343–358, 2001.

[KB96] Pieter S. Kritzinger and Falko Bause. *Stochastic Petri Nets - An Introduction to the Theory.* Vieweg Verlagsgesellschaft, 1996.

[KK00] Robin Kravets and P. Krishnan. Application-driven power management for mobile communication. *Wireless Networks*, 6(4):263–277, 2000.

[Kri86] Pieter S. Kritzinger. A Performance Model of the OSI Communication Architecture. *IEEE Transactions on Communication*, COM-34(6):554–563, June 1986.

[Kri03] Pieter Kritzinger. Communication Systems and Computer Networks- An Advance Course. Course notes in the Department of Computer Science, University of Cape Town, 2003.

[Mar03] Siegle Markus. Computer Systems Modelling. Course notes available online WWW page http://www7.informatik.uni-erlangen.de/ siegle/lectures.html. Accessed May, 2003.

[MC03] Rex Min and Anantha Chandrakasan. MobiCom poster: top five myths about the energy consumption of wireless communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 7(1):65–67, 2003.

[Mur03] Kruger Murray. Wireless Module Engineer. Siemens AG South Africa, Personal interview 6 October 2003.

[PER94] PERFORM. *UltraSAN User's Manual.* The University of Illinois, Center for Reliable and High-Performance Computing Coordinated Science Laboratory University of Illinois at Urbana-Champaign, version 3.0 edition, 1994.

[PRR01] Chiara Petrioli, Ramesh R. Rao, and Jason Redi. Guest editorial: energy conserving protocols. *Mobile Networks and Applications*, 6(3):207–209, 2001.

[Qur96] Muhammad A. Qureshi. *Construction and Solution of Markov Reward Models.* Doctor of philosophy, University of Arizona, Department of Electrical and Computer Engineering, 1996.

[Rud83] H. Rudin. From formal protocol specification towards automated performance prediction. In *Workshop Protocol Specification, Testing, Verification, Rüschlikon, Switzerland*, pages 257–269, 1983.

[Rud85] Harry Rudin. An Informal Overview of Formal Protocol Specification. *IEEE Communications Magazine*, 23(3):46–51, March 1985.

[SIE03] SIEMENS. TC45-Siemens Cellular Engine: Java User's Guide. Used with permission from Siemens AG. Johannesburg, South Africa, July 2003.

[Ste94] William J. Stewart. *Introduction to the Numerical Solution of Markov chains.* Princeton University Press, 1994.

[STR88] R. M. Smith, Krishor S. Trivedi, and A. V. Ramesh. Performability Analysis: Measures, an Algorithm, and a Case Study. *IEEE Transactions on Computers*, 37(4):406–417, April 1988.

[Svo89] Liba Svobodova. Implementing OSI Systems. *IEEE Journal on Selected Areas in Communications*, 7(7):1115–1130, September 1989.

[SWW$^+$02] Wayne Stark, Hua Wang, Andrew Worthen, Stphane Lafortune, and Demosthenis Teneketzis. Low-energy wireless communication network design. *IEEE Wireless Communications*, 9(4):60–72, August 2002.

[Tan96] Andrew S. Tanenbaum. *Computer Networks*, volume 3. Prentice Hall, 1996.

[Tri82] Kishor S. Trivedi. *Probability and Statistics with Reliability, Queuing and Computer Science Applications.* Prentice Hall, 1982.

[vR02a] 3GPP TS 25.301 Universal Mobile Telecommunications System (UMTS) version 5.2.0 Release 5. Radio Interface Protocol Architecture. WWW page http://www.etsi.org, August 2002.

[vR02b] 3GPP TS 25.304 Universal Mobile Telecommunications System (UMTS) version 5.2.0 Release 5. UE Procedures in idle mode and cell reselection in Connected mode. WWW page http://www.etsi.org, December 2002.

[vR02c] 3GPP TS 25.323 Universal Mobile Telecommunications System (UMTS) version 5.2.0 Release 5. Packet Data Convergence Protocol-PDCP specification. WWW page http://www.etsi.org, August 2002.

[vR02d] 3GPP TS 25.324 Universal Mobile Telecommunications System (UMTS) version 5.2.0 Release 5. Broadcast/Multicast Control-BMC protocol specification. WWW page http://www.etsi.org, September 2002.

[vR02e] 3GPP TS 25.302 Universal Mobile Telecommunications System (UMTS) version 5.3.0 Release 5. Services provided by the physical layer. WWW page http://www.etsi.org, December 2002.

[vR02f] 3GPP TS 25.303 Universal Mobile Telecommunications System (UMTS) version 5.3.0 Release 5. Radio Resource Centre-RRC Protocol specification. WWW page http://www.etsi.org, December 2002.

[vR02g] 3GPP TS 25.321 Universal Mobile Telecommunications System (UMTS) version 5.3.0 Release 5. Medium Access Control-MAC Protocol specification. WWW page http://www.etsi.org, December 2002.

[vR02h] 3GPP TS 25.322 Universal Mobile Telecommunications System (UMTS) version 5.3.0 Release 5. Radio Link Control-RLC Protocol specification. WWW page http://www.etsi.org, December 2002.

[Wil99] Ingo Willimowski. UTRAN - UMTS Terrestrial Radio Access Network. WWW page http://www.imst.de/mobile/itg/itg_umts/willimowski.pdf, April 1999.

[WJPSW98] Hagen Woesner, Jean-Pierre, Morten Schläger, and Adam Wolisz. Power Saving Mechanisms in Emerging Standards for Wireless LANs: the MAC Level Perspective. *IEEE Personal Communication Systems*, June 1998.

[Zim80] Hubert Zimmermann. OSI Reference Model-The ISO Model of Architecture for Open Systems Interconnection. *IEEE Transactions of Communications*, COM-28(4):425–432, April 1980.

[ZR97] Michele Zorzi and Ramesh R. Rao. Energy-constrained error control for wireless channels. *IEEE Personal Communications*, 4(6):27–33, December 1997.

[ZR01] Michele Zorzi and Ramesh R. Rao. Energy Efficiency of TCP in a Local Wireless Environment. *Mobile Networks and Applications*, 6:265–278, 2001. Dipartimento di Ingegneria, Universit di Ferrara, 44100 Ferrara, Italy.