A Tool for Real Time Simulations of TCP/IP over Wireless Fading Channels

Anna Ewerlid and Nilo Casimiro Ericsson
Signals and Systems Group
Uppsala University, Sweden
{Anna.Ewerlid, Nilo.Casimiro.Ericsson}@signal.uu.se

Abstract

We describe an ongoing effort to create a platform for realistic real-time simulations of wireless communication systems. The purpose of the work is primarily to evaluate the performance of existing and new approaches to provide wide area mobile and wireless Internet. The new approaches comprise of enhancements to transport protocols, and spectrally efficient allocation of the wireless resources.

The simulation tool is, just like the communication protocol stack, divided into several separate layers, each providing its functionality. It is also distributed over several computers that communicate over an ethernet.

The main contribution, that makes the real-time simulation possible, is provided by very heavy calculations and extensive off-line simulations of the lower layers in the communication stack. Through these simulations we obtain large tables of probabilities that are accessed on-line by the simulator running the higher layers.

1 Introduction

We are developing a simulation tool for wireless, mobile, wide area networks. The purpose of the simulation tool is to enable complex simulations of communication systems to be carried out in real-time. Ultimately, we want to investigate and verify the theoretical work and results that we have carried out in the Wireless IP (WIP) project within the Swedish research program for Personal Computing and Communications (PCC) [2].

Our results this far show that a high gain in spectral efficiency can be obtained using channel prediction and scheduling algorithms [7, 14]. Through channel prediction [5], we obtain estimates of future transmission quality over the channels between a base-station and several mobile terminals. These estimates are fundamental in the following step, the media access scheduling. The scheduler's task is to match the offered Internet traffic to the different channels, so that we utilize the scarce bandwidth as efficiently as possible. The scheduler should also take higher layer requirements into account, e.g. priorities for different users, and constraints imposed by the applications.

We would also like to evaluate the implications of our approach for the higher layers in the communications protocol stack: How will different flavors of TCP (Transmission Control Protocol) react on the offered wireless services [3, 11–13, 15]? The media access scheduler and the TCP protocol work on different time-scales, TCP in the scale of Round-Trip Times (RTT) and the scheduler in the scale of channel quality fluctuations, but are not necessarily independent of eachother.

At first, simulating this seems like a straight-forward problem to solve: We could generate error-patterns due to erroneous reception at the mobile client, then simply apply these patterns to the transmitted data. However, our complex link layer ARQ system together with the media access scheduler influenced the higher layers, i.e. the transport layer and the TCP protocol. This interplay between layers that operate on different time scales is crucial to the understanding and optimization of the whole protocol stack. This means that we can't simply do the link and transport layer simulations separately and independently.

In Section 2 we briefly describe the communication system that we primarily want to evaluate. In the following section we describe and discuss in more detail the problem of using layered simulations for this particular type of communications system. Finally, in Section 4, our solution to the simulation problem is proposed, and the paper is concluded in the following section.

2 Simulated System Background

This section briefly describes the system we intend to simulate, in order to evaluate its influence on the applications that generate the data traffic handled by the system. A central term in these simulations and system proposals is the presence of *fast fading channels*. Fast fading occurs when a radio receiver travels through areas with reflecting walls and other obstacles generating multi-path propagation of the transmitted radio signal, which consequently interferes with itself. When the interference is destructive, the radio signal fades out. The fast fading can give rise to fluctuations in the received signal power of more than $10dB$ within time spans of $1ms$, depending on the carrier frequency of the signal and the speed of the mobile terminal.

There is also a *slow fading* present in the mobile wireless channels. This is due to the distance variations between the base-station and the mobile terminals, i.e. the path loss. In this slower time-scale we expect to see an interaction with the TCP flow control mechanisms.

2.1 Hybrid ARQ and Scheduling

To achieve a high system throughput also over fading channels, adaptive methods for the adjustment of the modulation alphabet and the coding complexity (the transmission rate) can be used.

One approach to achieve adaptation, is the Hybrid type-II ARQ (Automatic Repeat reQuest) scheme combined with an Adaptive Modulation System (AMS) [9], further described below. A rather different approach is based on prediction of the channel quality, thereby allowing for scheduling of the transmission among multiple users, taking the channel variations into account explicitly [6].

The idea behind HARQ-II/AMS is to reduce the transmission rate at each retransmission attempt. Therefore, more immunity against channel impairments can be provided in the following transmission attempts while still having the prospect of a high transmission rate in the first attempt. The drawback of this approach is the delay imposed by the method: The adaptation of coding rate/modulation is only controlled by the ACK/NAK (ACKnowledgement / Negative AcKnowledgement) feedback signal which increases the system delay in poor channel conditions.

In the scheduling approach, prediction of different user channels provide a basis for detailed scheduling of the transmission, by combining time-slot allocation and adaptive modulation. This approach can also take into account the desired error-probability and the priority associated with different users, as well as the current traffic situation. Moreover, the frequency band can be used efficiently, since the different users are allocated time-slots when their transmission conditions are predicted to be favorable, allowing them to use a high modulation level. The resulting constant and low (user/application-specified) error-rate provides the error correcting codes with manageable data, avoiding bandwidth consuming re-transmissions. The main drawback is the sensitivity to channel prediction errors.

An even more promising solution, according to preliminary investigations, would be a combination of the two methods [8]. The combined method successfully reduces the drawbacks of the individual methods described above: Knowledge of the channel conditions indicates the initial coding rate/modulation to use, reducing the delay due to numerous NAKs. The presence of the ARQ-scheme provides robustness against incomplete information in the scheduler, due to channel prediction errors.

Ordinary Internet traffic should be able to run on top of this type of system, since we don't aim at making any large changes to the widely accepted TCP/IP protocol stack. However, we would like to evaluate different flavors of the flow control algorithms within the TCP transport protocol, and their performance over our enhanced wireless links.

In Figure 1 we present an overview of the system we would like to simulate, and also qualitatively evaluate by running it in real-time, with real applications running on the end-points that communicate over the simulated links, using real transport protocols.

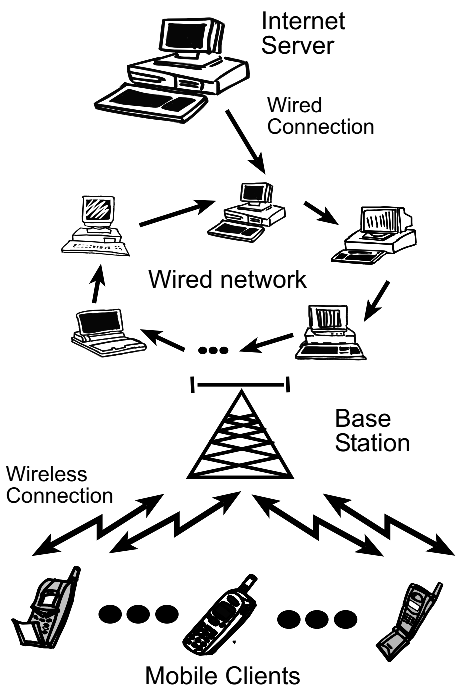

Figure 1. Overview of the communication system that we need to simulate.

3 The Simulation Problem

We need to run a realistic communication system *simulation* in real-time. At the lowest level, we need to simulate mobility of terminals in order to generate realistic fading radio channels. The next level contains the wireless link approach presented in the previous section. It comprises of a predictor for the fading channels, an advanced ARQ system using incremental redundancy and channel coding, and a media access scheduler. In the level above this, we have a number of transport protocols (TCP and UDP in different flavors), some of which have their own ARQ mechanisms that take action when things appear to go wrong. In this level we also have aggregation of traffic from different nodes in the network, destined for one of the mobile terminals under the control of that particular base station. Above that we have applications that utilize these underlying protocols for transmission of application and control information.

3.1 Physical Layer

In the physical layer we have to take into account the varying properties of the physical radio channel, properties that depend on a large amount of parameters, such as the speed at which the mobile terminal is moving, the environment in which it moves, and the interference from other neighboring transmitters. This type of simulation requires excessive calculations when a realistic outcome is desired. We use methods that involve ray-tracing of individual paths of the radio signals, in order to quantify the interference at specific points in space (the points where the receiving mobile terminals are situated) [5].

We have previously been using very short recordings of channel sounding measurments, that is, a kind of pinging from a base station that is recorded by a mobile receiver. These measurments only allowed us to run the simulations for about $140ms$, which is much too short to see the consequences of a time-varying channel quality at the higher layers. To overcome this problem, channel data are generated by a sophisticated model for the wireless fading channels, a model that allows us to extract arbitrarily long sounding measurements through simulations. The measured data has been used to validate the channel model.

3.2 Link Layer

The link layer contains the functionality for error control and error protection of the data transmitted over the wireless link. It also takes care of the ARQ mechanism, which in our particular case is an incremental redundancy protocol that basically never retransmits the same data in case of a negative acknowledgement (NAK), but instead transmits a data chunk with additional redundant information to help the decoder at the receiver to correctly decode the message. The decoding of these types of codes (convolutional codes) is very processor and memory demanding, since they result in state machines with a very high number of possible states, out of which the most probable should be chosen.

One decoder is furthermore required for each mobile terminal that we wish to simulate. The computational complexity forbids us to run a complete set of decoders in real-time. Thus, the need to find a realistic approximation to the decoding and ARQ system.

3.3 Media Access Scheduler

On top of the link layer we need to run a media access scheduler that not only takes into account the demands from the queues in the receiving buffers on the wired network side, but also is aware of the channel conditions for each active link and makes efficient time-slot allocations based on this information. The scheduler runs a number of different algorithms for evaluation [7].

The decision of the scheduler is highly dependent of the instantaneous channel quality in a specific link between the base station and a mobile terminal. The scheduler may decide that access should be given to a user in a specific time-slot, based on both the predicted quality of the channel, and the amount of data that is queued up in the buffer for that user.

3.4 Our Layered Simulation

We found it appropriate for our current and future requirements, to divide the simulation process into three steps. Step one and two are carried out once and for all for a given system, whereas the third step, the system simulation, can be done over and over again:

1. Run channel simulator
   - Set up specifications (environment, mobile speed, carrier frequency, carrier bandwidth) for N channels
   - Generate sounding measurements and store in N files
2. Run link layer simulator
   - Set up link layer specifications (ARQ, adaptive modulation, error control coding)
   - Generate statistics for all possible and interesting outcomes from the link layer protocol and store in fast accessible data structures
3. Run overall real-time simulation
   - Set up applications (clients, services, TCP flavors) and scheduling algorithm
   - Use sounding measurements for scheduling
   - Use link layer statistics for each transmitted link layer frame
   - Run the clients over the simulated system

3.4.1 Channel Sounding Simulation

Based on advanced ray-tracing simulations of individual wireless links, where a mobile terminal travels through a “landscape” of electromagnetic wave diffracting objects, we are able to generate arbitrarily long channel sounding measurements. The transmitted signal is subject to self-interference, path loss, and measurement noise, making the “measurement” as realistic as possible. The resulting channel sounding measurements have been validated with respect to all the important statistics, using the real measurements mentioned above.

The sounding data will be used in three ways:

- In the scheduler, where it is used as the predicted channel quality. The predictions can not be assumed to be accurate, so a prediction error is added to the “real” channel quality. The prediction error distribution has recently been found, and will be introduced to the simulation model.

- In the decoder at the receiver, where it is used as an estimation of the instantaneous channel quality. This estimation is also assumed to be inaccurate, so an estimation error is added to the “real” value. This error is smaller than the prediction error.

- In the transmission simulation, where it is used as the “actual” channel quality, that gives rise to a certain distortion of the received data.

3.4.2 Hybrid ARQ Simulation

In Section 2.1 we described the Hybrid ARQ and Scheduling used for the link layer transmissions. Here we describe how we simulate this layer off-line, in order to speed up the overall system simulation.

The procedure is best described by outlining an example from our system:

A link layer transmission frame consists of 48 data-carrying time-slots, each of which can be allocated to a different user (or mobile terminal). A time-slot can contain two or more link layer packets, depending on the modulation level (numbers of bits/waveform). Each link layer packet is encoded with an error correcting (outer) code and an error detecting (inner) code. The transmission rate (a combination of the modulation level, and the coding rate) is decided by the scheduler, depending on the predicted channel quality. The actual coding rate is decided by the ARQ-system, depending on which of up to three transmission attempts that is performed, since the coding rates need to be taken in a specific order.

Up to three transmission attempts are performed for a link layer packet before giving up, and each transmission attempt will depend on the previous one, since the Hybrid ARQ system uses all the previously received versions of the (erroneous) packet in the decoding process. This will generate a tree of depth, $d = 3$, and width according to all the possible combinations of the parameters *channel quality, coding rate, and modulation level*. The coding rate and modulation level are bounded discrete values, but the channel quality is a bounded continuous value. Therefore we need to discretizise the channel quality into a limited number of intervals.

The statistics from the link layer simulations is generated by simulating a large number of link layer packet transmissions for each of the combinations of the parameters presented above. Most of the combinations will result in a successful transmission for all the packets, thus terminating that branch of the tree at the first level. Some combinations will however cause a number of erroneous transmissions, issuing a second attempt by the ARQ system. For each combination of the transmission parameters, the probability of erroneous reception is approximated by the ratio between the number of erroneous packets and the total number of packets transmitted.

This procedure is repeated until all three transmission attempts have been performed, collecting the probabilities for erroneous reception in each node in the tree. In the case that very few packets are subject to a second or third attempt for a combination of the transmission parameters, the second and third transmission attempts are repeated multiple times with the same incrementally redundant packet that, after transmission, will be combined with the previously erroneously received packets.

This simulation process will result in a large combinatorial tree, requiring a fast lookup procedure for each combination of the transmission parameters and each transmission attempt.

4 Overview of the Simulator

The simulator is written in C++, under the Linux operating system. Large parts of the code consists of a library of routines for mathematics, communications, and signal processing, developed at the Department of Signals and Systems, Chalmers University of Technology [1].

We are aiming at making as realistic simulations as possible of the communication system that we want to investigate, including all the layers in the protocol stack. The simulation setup consists of four different machines, connected via Ethernet, that run different parts of the simulation, see Figure 2. Two of the machines are designated for running multiple client and server applications. They represent the communicating end-points in the simulated network. One machine, called the *Bridge*, takes care of the delaying and dropping of packets, according to the outcome from the fourth computer, the *Simulator*.

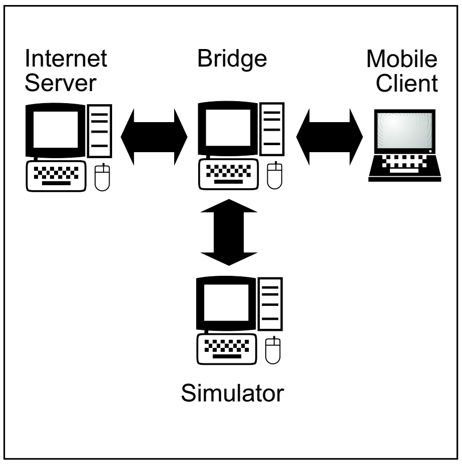

Figure 2. The simulation setup consists of four computers that communicate over an Ethernet.

It is only the higher layers, from the network and upwards, that are simulated in real time. The lower layers, that is the link and physical layers, have been simulated in advance, generating large databases that are used by the simulation of the higher layers. The reason for this partitioning became evident in the light of the computation requirements that we described in the previous section.

The data traffic that arrives at the simulated base station from the wired network is brought up to the network layer (IP) in order to separate the different source-destination pairs. We also to peek into the IP header to extract the type-of-service octet, that tells about different requirements on delay, throughput, and reliability.

The functions described above, are all simulated in the *Simulator* computer in Figure 2. The simulator thus simulates everything that happens from the network layer and down, in the base station, over the wireless link, and in each receiving mobile terminal.

Next, we describe the higher layers in the protocol stack. They run on the end-points of the simulated system, which in this case run on separate machines, and are referred to as *Mobile Client* and *Internet Server* in Figure 2.

4.1 Description of the Simulation Environment

An overview of the simulation environment is presented in Figure 2. The system consists of 4 dual-processor computers that are running Linux RedHat 7.2 with kernel 2.4.13.

One of the important parameters for the scheduling is the type of service. The system is designed to handle packets of different kinds of connections in different ways. To evaluate how they are handled by our system, we should simulate various kinds of services (telnet, FTP, HTTP, etc.). The connections are initiated by the Mobile Client. It establishes a UDP or TCP connection with the Internet Server. We place different versions of TCP (Westwood, Reno, Vegas, etc.) on them for comparison. Moreover, we can enable various end-to-end improvements, such as SACK (Selective ACKnowledgement), ECN (Explicit Congestion Notification), and ELN (Explicit Loss Notification), etc.

Packets are trapped in the machine referred to as the Bridge, the book-keeping machine, which places them in a holding-queue. The arrival of a packet results in the transmission of a UDP request to the Simulation machine, with information about the arrived packet. After the simulated transmission is finished (succesfully or not), a response will come, telling the Bridge to either release the packet for delivery to the destination, or to drop it as a consequence of an unsuccessful transmission over the wireless link.

The Bridge provides a connection between the Client and the Server in such a way that the two machines are unaware of any special arrangements made between them. TCP and UDP packets are lifted to the application layer and placed in a queue until a response comes from the Simulation machine, and all other kinds of packets (e.g. ARP, ICMP packets) are forwarded directly to the destination machine.

The fourth machine, referred to as the Simulator, simulates the wireless links, according to the description in the previous section. The Simulation software is organized in two processes, one for the actual simulation, and one for maintaining the connection with the Bridge machine. This structure allows us to perform the tasks simultaneously on the separate CPUs.

The Communication process is responsible for the contact with the Bridge machine. Whenever the Bridge machine gets a TCP or UDP packet, it places it into the holding-queue and sends a request to the Simulator, containing information about the packet (packet size, type of service, etc.) The Communication process of the Simulator receives the request and places a dummy packet into the appropriate queue in the network layer buffer.

Based on the contents of the queues and the predicted channel quality, the scheduler in the Computation process, will generate decisions that will be executed by the link layer. The link layer will follow the scheduler's decision, but give precedence to link layer packets that were NAKed in the previous transmission attempt. In the case that all the transmission attempts failed, the Simulator will tell the Bridge to drop the specific packet. In the case that an IP packet is successfully transmitted, the Simulator will tell the Bridge to release the packet.

5 Conclusions

Simulation of wireless Internet connections in real-time is an extremly complex and challenging task. We are creating a system that will allow us to simulate the end-to-end connection between the Mobile Client and the Internet Server. The assumed wireless link uses advanced algorithms for channel prediction, scheduling, and error correction. The real-time simulation system described in the article is going to be used for study of the implications of all these algorithms on the TCP/IP traffic. Moreover, we are going to evaluate various proposals, that have come over the years, for the improvment of TCP/IP performance over wireless links [4, 10, 12].

Appropriate performance measures are throughput, latency statistics and average data rates over wireless links. More complete results of the simulations will be presented at the conference.

References

[1] IT++: A Communication Simulation Library for C++. http://www.s2.chalmers.se/software/index.html.

[2] The Wireless IP Project within the SSF PCC (Personal Computing and Commnication) Program. http://www.signal.uu.se/Research/PCCwirelessIP.html.

[3] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. RFC2581, Apr. 1999.

[4] E. Amir, H. Balakrishnan, S. Seshan, and R. H. Katz. Effi cient TCP over networks with wireless links. In *HoTOS*, pages 35–40, 1995.

[5] T. Ekman. *Prediction of Mobile Radio Channels*. Licentiate Thesis, Uppsala University, Dec. 2000.

[6] N. C. Ericsson. Adaptive Modulation and Scheduling of IP traffi c over Fading Channels. In *Proc. IEEE Vehicular Technology Conference*, volume 2, pages 849–853, Amsterdam, the Netherlands, Sept. 1999.

[7] N. C. Ericsson. On Scheduling and Adaptive Modulation in Wireless Communications. Licentiate Thesis, June 2001.

[8] S. Falahati and N. C. Ericsson. Hybrid type-II ARQ/AMS and Scheduling using Channel Prediction for Downlink Packet Transmission on Fading Channels. In *Proc. PCC Workshop*, Nynäshamn, Sweden, Apr. 2001.

[9] S. Falahati and A. Svensson. Hybrid type-II ARQ Schemes with Adaptive Modulation System for Wireless Channels. In *Proc. IEEE Vehicular Technology Conference*, volume 5, pages 2691–2695, Amsterdam, the Netherlands, Sept. 1999.

[10] K. Fall and S. Floyd. Simulation-based Comparisons of Tahoe, Reno, and SACK TCP. Technical report, Lawrence Berkeley National Laboratory, Dec. 1995. ftp://ftp.ee.lbl.gov/papers/sacks.ps.Z.

[11] S. Mascolo. Smith's principle for congestion control in high-speed data networks. *IEEE Transactions on Automatic Control*, 45(2):358–364, Feb. 2000.

[12] S. Mascolo, C. Casetti, M. Gerla, S. S. Lee, and M. Sanadidi. TCP Westwood: congestion control with faster recovery. Technical Report 200017, Computer Science Department, UCLA, Los Angeles, CA, 2000.

[13] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanov. TCP Selective Acknowledgement Options. RFC2018, Oct. 1996.

[14] T. Ottosson, M. Sternad, A. Ahlén, A. Svensson, and A. Brunström. Toward a 4G IP-based Wireless System Protocol. In *RVK 2002*, Kista, Sweden, June 2002.

[15] G. Wu, Y. Bai, J. Lai, and A. Ogielski. Interactions between TCP and RLP in Wireless Internet. In *GLOBECOM*, pages 661–666, Rio de Janeiro, Brazil, December 1999.