Transport Protocol Performance over 4G Links: Emulation Methodology and Results

Stefan Alfredsson, Anna Brunstrom
Department of Computer Science
Karlstad University
SE-651 88 Karlstad, Sweden.
Email: {Stefan.Alfredsson, Anna.Brunstrom}@kau.se

Mikael Sternad
Signals and Systems
Uppsala University
SE-751 20 Uppsala, Sweden.
Email: Mikael.Sternad@signal.uu.se

Abstract— **This paper presents a wireless link and network emulator for the "Wireless IP" 4G system proposal from Uppsala University and partners. In wireless fading downlinks (base to terminals) link-level frames are scheduled and the transmission is adapted on a fast time scale. With fast link adaptation and fast link level retransmission, the fading properties of wireless links can to a large extent be counteracted at the physical and link layers. The emulator has been used to experimentally investigate the resulting interaction between the transport layer and the physical/link layer in such a downlink. The paper introduces the Wireless IP system, describes the emulator design and implementation, and presents experimental results with TCP in combination with various physical/link layer parameters. The impact of link layer ARQ persistency, adaptive modulation, prediction errors and simple scheduling are all considered.**

I. Introduction

The demand for higher capacity and wider coverage of wireless network access is increasing. As the third generation mobile systems is becoming commercialized, research focus has shifted towards 4G systems. One promising technology for 4G is orthogonal frequency division multiplexing (OFDM), which uses multiple carrier frequencies dedicated to a single data source.

One 4G system proposal based on OFDM has been developed within the "Wireless IP" project at Uppsala University, in cooperation with Chalmers University of Technology and Karlstad University [1], [2]. The main focus is to cover wide areas to service vehicular users, in excess of speeds of 100 km/h with a 30-fold bandwidth increase compared to UMTS/3G. To realize this goal, adaptive OFDM is used in combination with channel prediction. Transmissions can then be scheduled to maximize the total satisfaction of the users, depending on their current channel quality. This is combined with increased cross layer interaction, link level ARQ, and other mechanisms.

The Wireless IP system is conceived as an all-IP system intended for Internet traffic. It should be designed to provide a good service to the network and transport layers and it is thus important to consider the system level implications of lower layer design decisions. In order to allow performance measurements on the interaction between the physical/link layer design and upper layers the Wireless IP emulator (WIPEMU) was developed. In this paper we describe the design of WIPEMU along with experimental results that illustrate the impact of different lower layer parameter settings on transport protocol performance.

The experimental results presented in the paper focus on the impact of lower layer parameters on the performance of TCP, which is the dominant transport protocol on the Internet for reliable data transfer. The emulated scenario consists of a mobile wireless user downloading content from a server that is located in the fixed Internet. Parameters studied include the impact of adaptive modulation, the effect of channel prediction errors, the importance of link layer persistency and the impact of scheduling using a simple algorithm. The results presented in the paper also include an experimental validation of the emulator design against theoretically derived values.

The remainder of the paper is organized as follows. Section II motivates the use of emulation, describes the Wireless IP system proposal and details the WIPEMU design. Section III presents the experimental setup, explains the validation of the emulator and describes the experimental results. Section IV contains the conclusions.

II. WIPEMU

A. Emulation

Although there exist a number of previous emulators, such as NIST Net [3], End-to-end Network Delay Emulator (ENDE) [4], Ohio Network Emulator (ONE) [5], Delayline [6], Dummynet [7], Seawind [8], and various trace-based approaches [9], [10], [11], they typically model the network at a quite high abstraction level, where the underlying network is modeled by probability distributions for parameters like packet loss and delays. For our purposes, this abstraction is too coarse motivating the need for development of a new emulator. We want, for example, to evaluate the impact of fast link layer retransmissions, in combination with adaptive modulation (which gives a varying throughput on a short time scale), and user scheduling in both time and frequency. At the same time we want to see the implications on the system level, by running real network traffic over the emulated link. Although not considered in this paper, investigations of transport protocols that are designed to accept bit errors, like UDP-Lite [12] or TCP-L [13], require that a distinction is made between bit errors and packet loss.

WIPEMU has been developed to meet these needs. It is intended to be plugged into a real network environment as a gateway. This enables a wide range of transport protocol implementations and applications to be tested, since the common interface is a regular Ethernet connection. Although WIPEMU has been specifically designed to emulate the downlink of the Wireless IP system proposal, many of the features emulated are part of other similar 4G systems, such as the WINNER system [14], [15]. With appropriate reparameterization WIPEMU could thus be modified to also emulate other 4G systems.

B. Wireless IP System Downlink Design

As mentioned in the introduction the Wireless IP system is based on adaptive OFDM. The downlink bandwidth that is available within a base station sector is slotted in time and each of the slots is partitioned into a number of time-frequency bins. These resources are shared among the active users. The resource partitioning of the bins to different users is controlled by a scheduler that is located in (or close to) the base station. To support the scheduling decision, each user predicts the signal to noise ratio (SNR) for all bins, with a prediction horizon which is larger than the time delay of the transmission control loop, and signals its predicted quality estimates to the base station on an uplink control channel. Once a time-frequency bin has been assigned, the predicted quality estimate is used to select an appropriate modulation format for the transmission. A high SNR enables higher modulation to be used, and the reverse for low values of SNR. A variable link layer frame size is used, where the payload data in a bin makes up one link layer frame.

The bin size has been selected so that the channel appears as relatively flat and time-invariant within each bin, allowing all payload symbols within a bin to use the same modulation format. Based on a design vehicle speed of 100 km/h and a carrier frequency of 1900 MHz, the bin size has been set to 0.667 ms times 200 kHz. Using a downlink with a 5 Mhz radio bandwidth provides 25 channels of 200 kHz and means that 1500 times 25 frames are transmitted every second.

Each time-frequency bin carries 120 symbols[1]. Of the 120 symbols, 12 are for training and downlink control, leaving 108 payload symbols useful for data transfer. In the baseline system design, an adaptive modulation system that uses 8 uncoded modulation formats, BPSK, 4-QAM, 8-QAM, 16-QAM, 32-QAM 64-QAM, 128-QAM, and 256-QAM, is used for the payload symbols.

C. The Emulator

To facilitate the system emulation in conjunction with real network traffic, the emulator is installed in a gateway. In brief,

[1]A subcarrier spacing of 10 kHz and a symbol period of 111 μs is used.

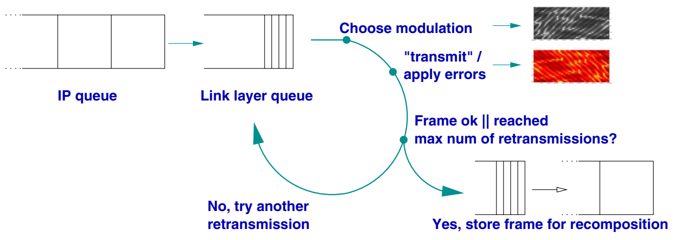
Choose modulation
"transmit" / apply errors
IP queue
Link layer queue
Frame ok || reached max num of retransmissions?
No, try another retransmission
Yes, store frame for recomposition

Fig. 1. The WIPEMU core functionality

the gateway collects IP packets destined for the “mobile node” and passes them to the emulator software, which emulates the transmission of the packets over the wireless link. The packets are then forwarded to the destination. At present the emulator only handles one user and is limited to the use of one time-frequency bin (or channel) in each timeslot, but multi-bin capability and real-time scheduling between users will be implemented in the future.

After packets have been collected in the gateway, they are placed in a queue of IP packets, see Figure 1. As packets fill up the queue, WIPEMU dequeues one packet at a time and decomposes it into link layer frames. For every frame, predicted channel data is consulted to decide the current signal-to-noise ratio (SNR). Assuming adaptive modulation, this ratio controls which modulation level to use[2]. When the frame is then “transmitted”, the non-predicted (i.e. “real”) channel is used when calculating the probability that the frame is received with symbol errors.

If the frame was transmitted without errors, it is stored for later recomposition into its constituent IP packet, and the next frame in the queue is transmitted. In the case of a transmission error, a number of link layer retransmissions are performed[3]. Retransmission takes priority over transmission of new data. To achieve optimal use of the wireless channel, the transmissions are pipelined. This means that frame reordering can occur, which may lead to packet reordering on the transport layer. If the frame is still in error after the maximum number of allowed retransmissions, the symbol error rate is calculated from the channel data, and bit errors are applied. These errors will then be contained in the re-composed IP packet, and their presence is often detected because the network or transport layer checksum is invalid. The recomposed IP packet is then forwarded to its destination.

Allowing the release of erroneous IP packets (as opposed to discarding packets with erroneous frames) enables experiments with protocols for loss differentiation (for example Checksum-based Loss Differentiation [16] or TCP-HACK [17]), or semi-reliable protocols (for example TCP-L [13] or UDP-Lite [12]), or other protocols that are able to handle packets with bit errors.

[2]The emulator can also be configured to use fixed modulation.

[3]In the present implementation, the frame is retransmitted and received separately, without using soft recombining/coding with the previously received incorrect frame.

Regarding the consulted channel data, it is an array of SNR sample values, one per bin, indicating the received channel power. This power experiences variations in strength, or “fading”, for vehicular users. This channel can be obtained in a number of ways. One way is to use channel sounding to get the measurements from a real environment. Another way is to use ray-tracing models, for example [18]. A third way is to use common mathematical models to simulate the channel, such as Rayleigh or Jakes fading models. These models produce the fast (short-term) fading characteristics. There may also be shadow fading (also known as slow fading) involved, which can be modelled by an additive slowly varying contribution to the received power, on the dB-scale. Shadow fading is often modelled as an AR(1) process with prescribed variance. For the experiments in this paper we use a Jakes model with added shadow fading.

With the FreeBSD Dummynet [7] system that is used in the gateway, the fixed part of the network path can also be emulated. It is abstracted into a packet loss ratio with a possible delay component. With the use of dummynet pipes, different loss ratios and delays can be combined to form complex network scenarios.

III. Experimental Evaluation

A. Experimental Setup

The use case considered in the experiments consists of a mobile user downloading content from a server using TCP. The user has a wireless connection to a base station, which in turn is connected to the rest of the Internet, which also connects the server. This scenario is shown in Figure 2.

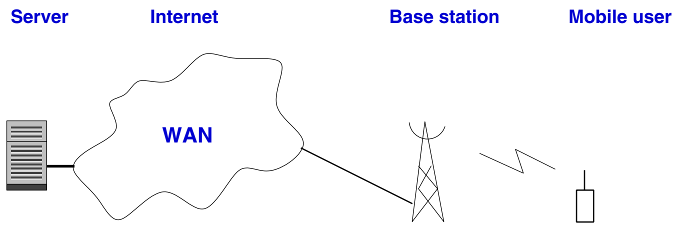

Fig. 2. Logical experiment setup.

The experiment setup to implement this scenario consists of three networked computers, as shown in Figure 3. The first is acting as a sender or content provider, the second is the gateway running the WIPEMU emulator, and the third is acting as a receiver and consumes data from the sender. All computers are also connected to an administrative network. This is used to control the experiments, so that packet capture in the emulated network is not affected by non experiment related packets.

As mentioned, the experiment consists of transmitting bulk data from the sender to the receiver. Meanwhile, all transmitted and received packets are collected at both end-points for later analysis.

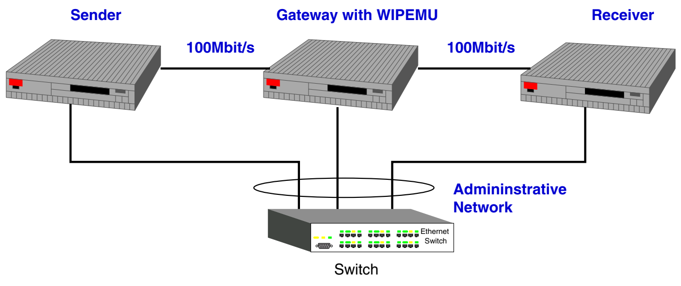

Fig. 3. Physical experiment setup.

Both the sender and the receiver run Linux 2.4.27 and we use the default available TCP version. The standard TCP settings are used except that the timestamp option is not used. The retentive TCP caching available in Linux is turned off to avoid creating any dependencies between the experimental runs. A standard MTU of 1500 bytes is used.

For the wireless channel calculations, the receiver and transmitter are assumed to have single antennas. Data for the 25 channels was obtained from a Jakes model with 12 taps set according to typical urban fading (-5.7, -7.6, $-inf$, -4.4428, -13.4, $-inf$, -13.5793, -14.2371, -14.4794, -16.9543, -20.0164, -24.3), at a speed of 75 km/h. To account for shadow fading, an AR(1) process provided shadow samples at an interval of 2 meter. The standard deviation, σ, was set to 4 dB and the pole, a, at 0.74. These channels were then processed to account for prediction errors, thus resulting in a “real” version and a “predicted” version of each channel. Two channels were then extracted out of the 25 channels. One that used a static allocation scheme (i.e. the same frequency band) and one that used a simple scheduling algorithm where the best channel was selected in each time slot.

As described in the previous section, the predicted version of the channel is used to choose the modulation scheme[4] for a frame, whereas the “real” version of the channel is used to calculate the error probability of the frame in the emulated transmission.

For the fixed part of the network, there is a set round-trip delay of 200 ms. Although not reported in this paper, lower delays have also been used, but the trends in the results remain. In addition, delays are introduced by packet queuing in the gateway (limited to 50 packets), and the transmission delay in the wireless link layer. These last two delays will vary, because of queue build up, and because the delay in the link layer depends on both retransmissions and varying modulation levels. Delays due to hand-over have not been introduced. The link level retransmission delay was set to 2 ms, which is due to the tight feedback loop in the Wireless IP system proposal.

Table I contains a compilation of the relevant parameters for the wireless network, the fixed network and the protocol settings.

[4]The switching levels for the adaptive modulation are optimized to provide maximum throughput in each link layer frame [19].

Fixed network	
Fixed network delay	200 ms (RTT)
Network queue size	50 packets
Wireless Downlink	
Frame transmission delay	0.667 ms
Channel model	12-tap Jakes typical urban fading model @ 75 km/h, 16 dB SNR + AR(1) shadow fading with variance 4 dB and pole at 0.74
Modulation	Adaptive BPSK,4-256 QAM with switching adjusted for perfect prediction or a prediction error of NMSE 0.1 [19], and fixed to BPSK, 4- and 8-QAM
Frame size	108 symbols
Coding	Uncoded M-QAM used
Scheduling	Static channel and best channel
Link ARQ	3 to 30 retransmissions
Wireless uplink	
Channel model	imposed bandwidth limit and delay
Packet loss	0% (lowest modulation level assumed)
Capacity	20 kbit/s
Delay	2 ms
Transport layer parameters	
Protocol	TCP (Linux 2.4.27)
Transferred data	3 Mb bulk data
TCP settings	Standard, except for disabled timestamp
MTU	Standard, 1500 bytes
Retentive TCP caching	Cleared before new connections

TABLE I

EXPERIMENT PARAMETERS

B. Emulator Validation

A validation of the emulator has been performed to see how the theoretical throughput and measured throughput match. A close match indicates that the emulator functions properly and does not drift, i.e. does not send data too slow or too fast as compared to what it should. The evaluation has been done by comparing the theoretically obtainable throughput to the one observed in experiments, while compensating for protocol overhead.

From the theoretical viewpoint, there are i bins per second, each carrying a payload of j symbols, where every symbol carries k bits corresponding to the log_2 of the modulation level M for the adaptive modulation system.

The theoretical maximum throughput per channel is thus

$$TP_{max} = i * j * log_2 M \ bits/s$$

In this system, i corresponds to 1500 bins/second, j corresponds to 108 symbols/bin, and M to $2^k, k \in \{1, 2, ..., 8\}$.

The measured throughput is determined by performing experiments with fix modulation and without packet errors. The measured throughput is calculated with the tcptrace [20] tool, operating on tcpdump packet captures at the receiver. As the throughput is calculated on the application layer, the TCP/IP header overhead must be compensated for. The last part of the IP packet does typically not fill an entire link layer frame, which also needs to be compensated for.

(1) bits/ sym	(2) theoretical (bits/s)	(3) measured (bits/s)	(4) header (bits/s)	(5) trunc (bits/s)	(6) total (bits/s)
1	162000	156416	4270	1285	161971
2	324000	312832	8540	2571	323943
3	486000	460992	12585	12313	485890
4	648000	625656	17080	5142	647878
5	810000	761672	20794	27386	809852
6	972000	921984	25170	24626	971780
7	1134000	1094904	29891	8998	1133793
8	1296000	1251320	34161	10284	1295765

TABLE II

COMPARISON OF THEORETICAL AND MEASURED THROUGHPUT WHEN USING FIXED MODULATION FORMATS, WITHOUT PACKET ERRORS

The IP packets are 1500 bytes, with a 40 byte header. The throughput, including headers, is therefore $40/1460 = 2.73\%$ higher. The frame truncation depends on the link modulation level, and is *(bits in frame - bits of data in last frame) / total number of bits per packet*. For IP packets of 1500 bytes, we have an overhead in percent of:

$$\frac{108 * log_2\ M - 1500 * 8\ mod\ (108 * log_2\ M)}{1500 * 8}$$

Table II displays the results of the discussion above. It shows the number of bits carried in each symbol per modulation level (1), and the theoretical maximum bits per second (2). The table shows further the measured throughput on the application level (3), and the throughput added by headers (4) and compensation for truncation in the last frame (5). Comparing columns (6) and (2) it is seen that the measured throughput with compensation (6) and the theoretical throughput (2) match very closely, in the order of 99.98%. A few replications were performed for each modulation level to verify that the results were stable.

C. Results

For this paper, the impact of four physical/link layer parameters on TCP are considered: the number of link layer retransmissions, the use of adaptive modulation, the impact of prediction errors and the impact of channel scheduling. The experiments were performed in the context of the experimental setup described earlier. The experiments were repeated 30 times with channel data generated with different random seeds, and the graphs indicate the mean values within the 95% confidence intervals.

First we study the performance of fixed modulation to establish the baseline performance. The question is then how to decide which fixed modulation level to use, to get an efficient utilization of the available radio spectrum. The tradeoff is between performance and reliability. The higher the modulation level, the more sensitive the transmission becomes to interference and noise leading to transmission errors. Consider Figure 4 which shows a transmission using three fixed modulation schemes, for a channel with a mean SNR of 16 dB. The y-axis shows the throughput, and the x-axis the number of maximum allowed link layer retransmissions.

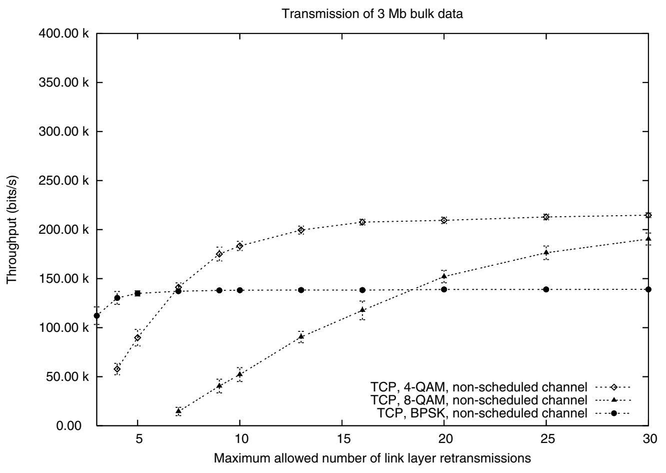

Fig. 4. Fix modulation, throughput

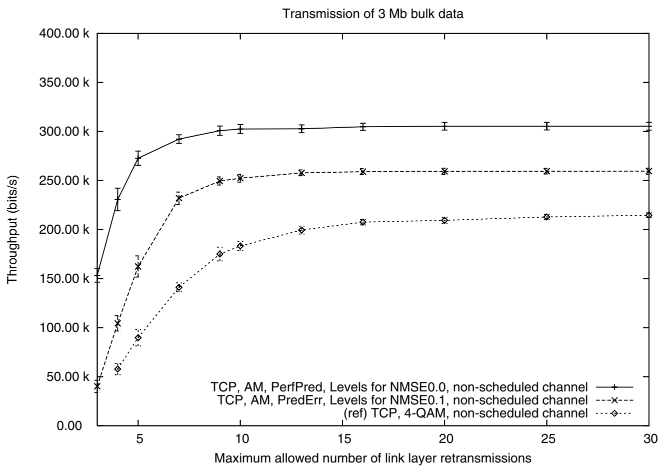

Fig. 5. Adaptive modulation, no channel scheduling, throughput

The lowest modulation level, BPSK, obtains the same throughput almost regardless of the amount of link layer retransmissions. Therefore, it indicates that it may be possible to use a higher modulation level. Increasing the modulation level one step to 4-QAM shows that it is possible to obtain a higher throughput, although this modulation level is more sensitive to the interference and noise as indicated by the need for link layer retransmissions. If the modulation level is increased even further to 8-QAM, we are now over-utilizing the radio resource. Even though more information is carried in every frame, the frames experience so many transmission errors that the retransmission mechanism cannot compensate.

Having established a baseline performance, we now proceed to investigate the use of adaptive modulation. The modulation level is now chosen for each link layer frame, depending on the predicted channel quality. In this context it is thus important to also consider the impact of prediction errors on performance. The performance results obtained with adaptive modulation are shown in Figure 5. We see that the best performance is obtained if we can perfectly predict the channel (top curve). However, in reality there is an error between the prediction and actual signal quality. This leads to a performance degradation, as evident by the middle curve. For this curve a prediction error of 0.1 NMSE (normalized mean square error) was used, representative for moderate vehicular speeds [19]. The bottom curve shows the best fixed modulation (4-QAM) as a reference, indicating that adaptive modulation is able to better utilize the variations in channel quality compared to a fixed modulation scheme[5].

The curves in Figure 5 (and Figure 4) also illustrate the importance of a persistent link layer for the performance of reliable data transport. The number of TCP retransmissions can be used to further illustrate the effect of varying link layer persistency and explain the slowdown in throughput. Figure 6 shows the corresponding percentage of TCP retransmissions

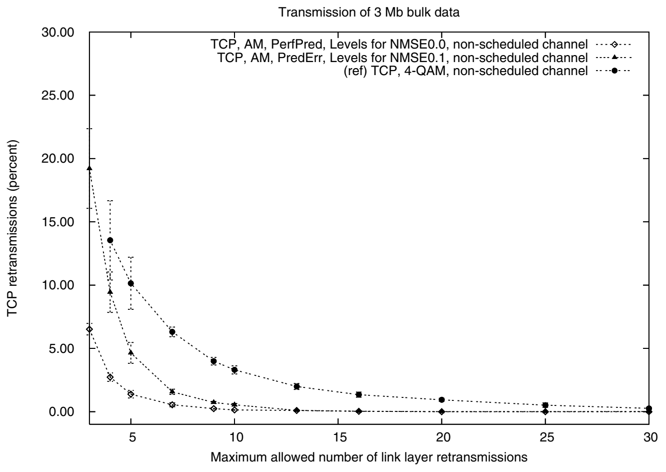

Fig. 6. Adaptive modulation, no channel scheduling, retransmissions

for the experiment discussed above (Figure 5). When there are few allowed link layer retransmissions, this leads to TCP packet loss, and the lost packets are then retransmitted. It is also seen that the packet loss is not linearly proportional to the degradation in throughput. For example, when a maximum of 3 link layer retransmissions are used, the upper curve in Figure 5 experiences about a 50% performance drop. However, looking at Figure 6, this happens when there are only about 6% TCP retransmissions. The explanation is that the losses are interpreted as congestion by TCP. As a result, TCP invokes its congestion avoidance mechanisms which slows down the sending rate.

The previous experiment used a static channel allocation for transmission. We next investigate how the use of a simple channel scheduling algorithm can further improve performance. In this case the channel (out of 25) with the best quality is chosen for transmission in each time slot[6]. The

[5]This does not consider the use of power control, which may be used to increase the signal quality, at the cost of increased inter-cell interference.

[6]Note that for a multi-user scenario, more advanced scheduling algorithms should be used to avoid starvation of users farther away from the base station.

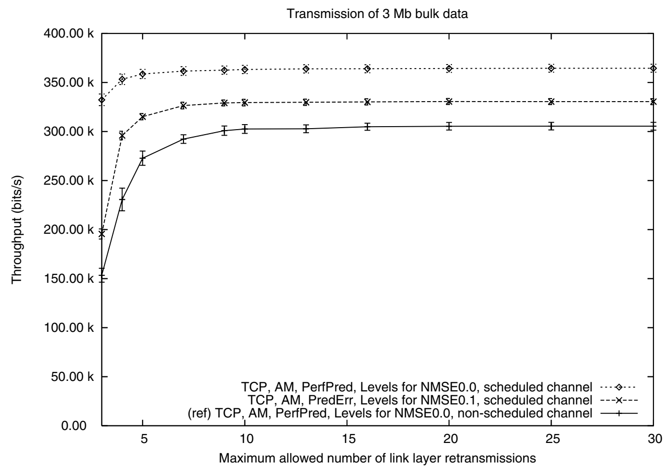

Fig. 7. Adaptive modulation, scheduled channel, throughput

results for this scenario are shown in Figure 7. As before, the upper curve shows the result for perfect prediction. The middle curve shows the impact of introducing prediction errors. These curves can then be compared to the bottom curve that shows the best result from the non-scheduled channel experiment described earlier. The results illustrate the potential benefits of using channel scheduling compared to using a static channel allocation scheme.

IV. Conclusions

As research on the next generation mobile and wireless systems continues to pick up pace, it is important to have a holistic (or system) view, while developing the underlying technologies. This paper contributes in this field by presenting an emulator of a 4G system downlink. This emulator enables performance evaluations on the application and transport layer, using real applications and transport protocols, where different parameters on the link and physical layers can be tuned. This paper describes the design and implementation, along with a validation, of the WIPEMU software. Experiments were performed to illustrate the impact of link layer retransmissions, adaptive modulation, channel prediction errors and channel scheduling on TCP performance. The results from these experiments indicate that the performance gains promised by lower layer techniques such as adaptive modulation and channel scheduling can also be realized at the transport and application layer.

V. Acknowledgements

The Wireless IP project is supported by SSF, the Swedish Foundation for Strategic Research. The first author was partly sponsored by the Swedish graduate school PCC++, funded by SSF. Part of the work in the paper has been carried out within the scope of the EU FP6 Network of Excellence Newcom. Anna Widenius has contributed to the design and implementation of the WIPEMU system.

References

[1] M. Sternad, T. Ottoson, A. Ahlén, and A. Svensson, "Attaining both coverage and high spectral efficiency with adaptive OFDM downlinks," in *Proceedings of IEEE Vehicular Technology Conference VTC2003-Fall*, Orlando, FLA, Oct 2003.

[2] S. Y. Wang. et al., "The design and implementation of the NCTUns 1.0 network simulator," *Computer Networks*, vol. 42, no. 2, pp. 175–197, June 2003.

[3] NIST Internetworking Technology Group, "NIST Net network emulation package," *http://snad.ncsl.nist.gov/itg/nistnet/*, November 2004.

[4] I. Yeom, "Ende: An end-to-end network delay emulator," Master's thesis, Texas A&M University, College Station, TX, 1998.

[5] M. Allman and S. Ostermann, "ONE: The ohio network emulator," Computer Science, Technical Report TR-19972, 1997.

[6] D. B. Ingham and G. D. Parrington, "Delayline: A wide-area network emulation tool," *Computing Systems*, vol. 7, no. 3, pp. 313–332, 1994.

[7] L. Rizzo, "Dummynet: A simple approach to the evaluation of network protocols," *ACM Computer Communication Review*, vol. 27, no. 1, pp. 31–41, January 1997.

[8] M. Kojo, A. Gurtov, J. Manner, P. Sarolahti, T. Alanko, and K. Raatikainen, "Seawind: A wireless network emulator," in *Proceedings of 11th GI/ITG Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems*, September 2001.

[9] B. Noble, M. Satyanarayanan, G. Nguyen, and R. Katz, "Trace-based mobile network emulation," in *Proceedings of ACM SIGCOMM '97*, September 1997.

[10] B. Noble, G. Nguyen, M. Satyanarayanan, and R. Katz, "RFC 2041: Mobile network tracing," October 1996.

[11] B. Melander and M. Björkman, "Trace-driven network path emulation," Department of Information Technology, Uppsala University, Sweden, Technical report 2002-037, 2002.

[12] L.-A. Larzon, M. Degermark, S. Pink, L.-E. Jonsson, and G. Fairhurst, "RFC 3828: The lightweight user datagram protocol (udp-lite)," July 2004.

[13] S. Alfredsson and A. Brunstrom, "TCP-L: Allowing bit errors in wireless TCP," *Proceedings of IST Mobile and Wireless Communications Summit 2003*, June 2003.

[14] M. Sternad, T. Svensson, and G. Klang, "The WINNER B3G system MAC concept," in *To be published in proceedings of VTC 2006-Fall*. IEEE, September 2006.

[15] S. Falahati. ed., "Assessment of adaptive transmission technologies," IST-2003-507581 WINNER, WINNER Deliverable D2.4, 2005, can be downloaded from http://www.ist-winner.org.

[16] J. Garcia and A. Brunstrom, "Checksum-based loss differentiation," *Proceedings 4th IEEE Conference on Mobile and Wireless Communications Networks (MWCN 2002), Stockholm, Sweden*, September 2002.

[17] R. K. Balan, B. P. Lee, K. R. R. Kumar, L. Jacob, W. K. G. Seah, and A. L. Ananda, "TCP HACK: A mechanism to improve performance over lossy links," *Computer Networks*, vol. 39, no. 4, pp. 347–361, July 2002.

[18] M. Berlin, "Channel modeling and simulation for the mobile internet," Master's thesis, Uppsala University, 2002.

[19] M. Sternad and S. Falahati, "Maximizing throughput with adaptive M-QAM based on imperfect channel predictions," in *Proceedings of IEEE PIMRC*, Barcelona, Sep 2004.

[20] S. Ostermann, "Tcptrace," *http://www.tcptrace.org/*.