Bridging the Gap Between Physical Layer Emulation and Network Simulation

Stylianos Papanastasiou*, Jens Mittag†, Erik G. Ström* and Hannes Hartenstein†
*Signals and Systems, Chalmers University of Technology, Sweden, Email: {stypap, erik.strom}@chalmers.se
†Institute of Telematics, University of Karlsruhe, Germany, Email: {jens.mittag, hannes.hartenstein}@kit.edu

***Abstract*—Many of the simulations reported in wireless networking literature contain several abstractions at the physical layer and the corresponding channel models. In particular, the basic simulation unit assumed in such simulations is the frame (or packet), which omits considerations of the signal processing details at the physical layer, such as frame construction and reception. Due to this abstraction, available channel models for network simulators are applied to frames as a whole and cannot reflect properly the effects of fast fading or frequency-selective channels. Moreover, it is not possible to study the mechanisms of the physical layer and their impact on higher layers such as the MAC. Therefore, we propose to address the lack of accurate physical layer representation in modern network simulators by incorporating a physical layer emulator for OFDM-based IEEE 802.11 communications into the popular NS-3 simulator. In this paper, we outline the architecture of the physical layer emulator and present initial results which highlight the promise of the new architecture in providing more detailed simulations to the networking community. The additional memory and computational requirements of the new model are also discussed.**

I. Introduction

Simulation has proved to be an invaluable tool for the networking community, supporting the controlled assessment of the impacts of wireless environments on existing communication mechanisms and the evaluation of solutions to emerging issues. At present, most popular networking simulators include certain basic abstractions in their physical layer implementations as well as the accompanying channel models. Thereby, they make a particular selection in the trade-off between simulation time and accuracy. Specifically, modern simulators such as NS-3 [1] and QualNet [2] contain a fairly accurate representation of the layers above and including the MAC, but abstract significantly the physical layer and the channel.

Notably, the smallest simulation unit considered in such simulators is the packet standing for a collection of bits. The collection is treated as an inseparable unit and does not allow the characterisation of individual bits as erroneous, i.e. the frame is received in its entirety or not at all. However, in much of the physical layer simulation and channel modeling literature, individual bits are treated separately and the smallest unit employed is generally the time sample, i.e. a complex representation of the produced signal at the sender.

Due to the aforementioned collective consideration of frame bits, the corresponding channel models are statistical abstractions that apply to the frame as a whole. Similarly, the signal reception characteristics of a frame are expressed only in terms of the average received signal strengths and average signal-to-interference-noise ratios (SINR). Yet, the computation of the SINR is based on the additive white gaussian noise (AWGN) channel model and assumes that amplitudes of interfering signals are gaussian distributed — an assumption which is not inherently valid in real systems. Intuitively, a more detailed representation of a frame, in terms of bits and complex time samples, allows not only the application of more accurate channel models but also, since an implementation of a real receiver in software is used, the study of different low-level receiver techniques and their impact on higher layers of the stack — studies that are not possible with current network simulators.

Recently, several research efforts have focused on these issues, e.g. Judd et. al [3], Tan et. al [4] and others [5], [6]. While [3] suggests a wireless channel emulator which connects real IEEE 802.11-based systems in order to enable controlled and repeatable wireless experiments, the authors of [4], [5], [6] present several software defined radio platforms to support a flexible development of communication systems and to provide the ability to adjust and test lower layer techniques rapidly. However, their work emulates either the wireless channel or the physical layer — but not both at the same time — and are either expensive or difficult to use for network studies with more than a few nodes, especially when nodes shall be mobile. In addition, the work in [3] supports only the 2.4 GHz frequency band and is therefore not applicable for communication systems operating at 5 GHz. We therefore propose to 1) combine both approaches in order to provide a joint emulation of the wireless channel and the IEEE 802.11 physical layer chipsets for pure OFDM-based communication at any carrier frequency and 2) to validate the approach at a microscopic level with only a few nodes before scaling it up to a full network.

Admittedly, the use of the packet as a basic simulation unit leads to reduced computational requirements as compared to the time sample case, where a detailed (and thus computationally intensive) imitation of a receiver and sender need to be employed. Nonetheless, the resulting high level of simulation accuracy may be desirable and even necessary for certain applications and studies; for instance, consider the requirements of safety-related research on the topic of inter-vehicle communications, where it is mandatory to determine precisely the circumstances under which a vehicle can successfully receive a frame or not.

The rest of this paper is organised as follows. Section II outlines the evolution of IEEE 802.11 physical layer models in popular network simulators. Section III begins with an overview of the architecture of our physical layer emulator implementation, highlights the transition from packet to signal level and provides a detailed description of the frame construction, wireless channel and frame reception process. Section IV then contrasts the integrated physical layer emulator in NS-3 with the traditional simulation mechanisms through a particular simulation scenario. Concluding remarks and a plan for future work are included in Section V.

II. Related Work

The accuracy in models for the physical layer of wireless communication networks has considerably improved in recent years and evolved from simple frame reception and deterministic radio propagation models towards models which account for cumulative noise and interference, varying signal propagation conditions and advanced transceiver features such as frame capture. For instance, consider that the widely used NS-2 [7] simulator initially utilized a simple carrier sense and reception threshold concept. This basic approach assumes an incoming frame relevant for upper layers (e.g. for MAC) only if its received signal strength is above a configurable carrier sense threshold; otherwise, it is ignored. Similarly, a frame is considered to be received successfully as long as there is no interference present and the received signal strength is greater than a given reception threshold. In order to overcome those simplifications and increase accuracy, Chen et. al proposed an improved IEEE 802.11 simulation model to account for cumulative noise and interference at a receiver [8]. Their improvement keeps track of the incoming frames at each receiver and computes the minimum signal to interference-noise-ratio (SINR) observed for each frame. A comparison of the observed SINR values (of each frame) with modulation and coding specific reception thresholds then yields the decision whether a frame can be received or not. By using this method and adjusting the SINR thresholds, e.g. based on empirical results, it is possible to approximate the behavior of rather simple or sophisticated receivers.

A similar approach to [8] has been proposed and extended by [9] and [10] in order to address frame capture capabilities and to distinguish between frame preamble/header and body in both NS-2 or QualNet [2]. With frame capture it is possible to account for advanced receiver technologies which provide the ability to synchronize to a new incoming frame even if a different frame is already being decoded. These models have also been ported to other popular simulators, such as OMNeT++ [11] and NS-3 [1] as well as to impulse-radio ultra-wide band communication simulations [12].

Besides SINR-based reception models, the NS-3 [1], Jist/SWANS [13] and GloMoSim [14] simulators provide support for reception decisions based on bit-error rate (BER) computations. While SINR-based models neglect the length of a frame, BER-based reception models consider the SINR of the frame, derive a corresponding BER analytically and calculate the frame error rate by an upscale of the single BER to the whole frame length. An overview and in-depth explanation of analytical BER models and computations can be found in [15] or [16].

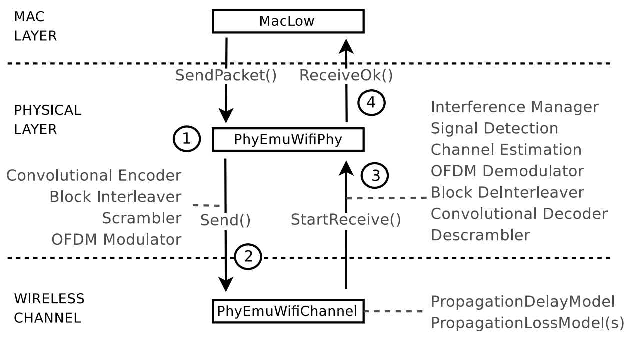
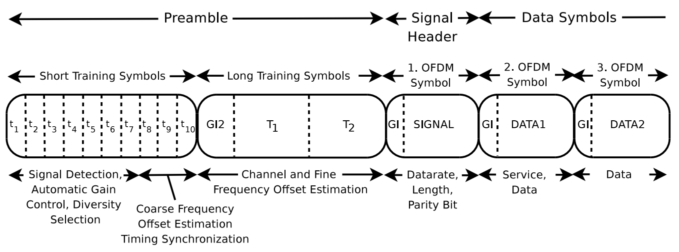


Fig. 1. Architecture of the physical layer emulation within NS-3. First the frame is transformed into a sequence of bits (1). After modulation (2) the bits are encoded into a sequence of complex time domain samples, which wireless channel models will operate on. After demodulation (3) we consider the bits derived from the time samples. When all processing is finished (4), a comparison of transmitted and received bits yields whether the frame was received successfully or not.

In short, the packet (or frame) is the smallest bit-representing entity in modern network simulators, even when considering physical layer functions. Consequently, effects on lower layers, such as the wireless channel, or transceiver related phenomena will always impact the whole frame. For instance, when a frame is transmitted, oscillator and fine-grained transmit power control issues are neglected and one single and fixed transmit power is assumed during the whole transmission process. Similarly, radio propagation conditions are assumed to be non-varying during the reception of a frame, i.e. the received signal strength is static. As a result, frequency- and time-selective characteristics are difficult, if at all possible, to model, and, further, it is impossible to examine their impact on receiver synchronization and equalization techniques. From a physical layer research perspective, the existing approaches are therefore quite abstract and overly simplistic.

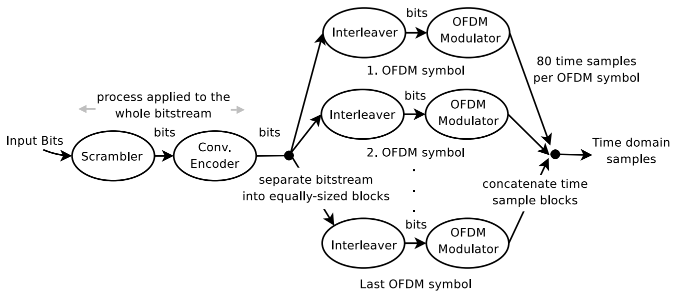
III. Implementation

In this paper, we propose the integration of a physical layer emulator into NS-3 in order to improve the accuracy of the IEEE 802.11 physical layer and the underlying channel models. The integration requires no changes in the upper layers, such as the MAC, and can be used as a “drop-in” alternative to existing PHY implementations. Figure 1 provides an overview of our modular implementation and outlines the various IEEE 802.11 physical layer mechanisms used in OFDM-based communication, as defined in the a, g amendments and the IEEE 802.11p draft for wireless access in vehicular networks. The legacy direct sequence spread spectrum (DSSS) modes and the infrared communication provisions of the standard are not considered in our work. The implementation makes extensive use of the open source IT++ mathematical and signal processing library [17], which provides several convenient data structures and functions including out-of-the-box provisions for signal processing and channel modeling techniques. The



Preamble
Signal Header
Data Symbols
Short Training Symbols
Long Training Symbols
1. OFDM Symbol
2. OFDM Symbol
3. OFDM Symbol
t1 t2 t3 t4 t5 t6 t7 t8 t9 t10
GI2
T1
T2
GI
SIGNAL
GI
DATA1
GI
DATA2
Signal Detection, Automatic Gain Control, Diversity Selection
Coarse Frequency Offset Estimation Timing Synchronization
Channel and Fine Frequency Offset Estimation
Datarate, Length, Parity Bit
Service, Data
Data

Fig. 2. PPDU frame format of an OFDM-based IEEE 802.11 PHY. Note that the service field in the second OFDM symbol is part of the header.



Input Bits
Scrambler
bits
Conv. Encoder
bits
process applied to the whole bitstream
separate bitstream into equally-sized blocks
Interleaver
bits
OFDM Modulator
1. OFDM symbol
2. OFDM symbol
Last OFDM symbol
80 time samples per OFDM symbol
concatenate time sample blocks
Time domain samples

Fig. 3. Transformation of a bit sequence into complex time domain samples during the construction of an IEEE 802.11 frame for OFDM-based communication.

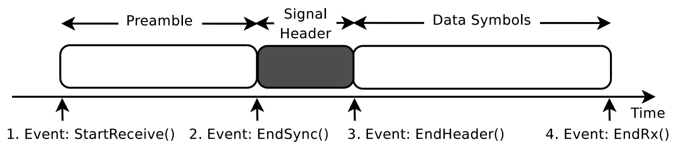


Fig. 4. The four events of the reception process: when the first sample of the frame arrives, the frame is added to the Interference Manager (1st event). After the duration of the preamble (2nd event), the emulator checks whether signal detection and synchronization was successful. If so, signal header decoding is performed (3rd event). If that is also successful, the detected frame length and data rate are used to decode the data symbols and to decide whether all symbols could be decoded successfully as well (4th event).

library itself has been widely used in physical layer research, is actively maintained and considered mature.

From a physical layer perspective, an IEEE 802.11 frame for OFDM-based communication is divided into three functionally different sections: a preamble, a signal header and the data unit section with the actual payload. Figure 2 shows the overall structure of such a frame as well as the objectives of each part. The preamble is precisely defined in the IEEE 802.11 standard and is identical for every frame regardless of transmission mode. It consists of ten repetitions of a short and two of a long training sequence, which are used for signal detection, automatic gain control, diversity selection, timing synchronization as well as channel and frequency offset estimation. After the preamble, there follows a signal header which contains information about the length of the data unit section, the modulation and coding scheme and includes a parity bit to support basic error detection. The last part contains the payload to be transmitted. In the following, we elaborate on the details of the frame construction, the wireless channel and the receiver modeling.

A. Frame Construction

The physical layer emulator mimics the behavior of a real IEEE 802.11 chipset. As illustrated in Figure 1, it accepts transmission requests from the MAC. Until this point, frames are treated as dummy objects which contain header information from different protocols but no actual payload. So, in order to perform the desired transition from packet to bit level, the physical layer emulator generates a random data bit sequence with a size equal to the length specified in the header of the frame object[1]. Afterwards, the emulator continues as specified in Section 17 of the IEEE 802.11 standard: the data bits are scrambled by the *Scrambler* which prevents long sequences of 0s or 1s, the *Convolutional Encoder* adds redundancy to enable error correction and the *Block Interleaver* ensures that long runs of low reliability bits are avoided. In addition, the *Block Interleaver* divides the bitstream into equally sized blocks, which later end up in the OFDM symbols to be transmitted. Then, the *OFDM Modulator* modulates the bits of each block using either phase-shift keying (BPSK or QPSK) or quadrature amplitude modulation (16-QAM or 64-QAM), inserts pilot symbols in four of the 52 sub-carriers to support channel tracking in the receiver and performs the final OFDM modulation per block — transformations that result in a sequence of complex time domain samples. Figure 3 depicts the aforementioned transformation from bit level to time domain samples. A similar process is also applied to the signal header of the frame with the special condition that bit scrambling is omitted and BPSK modulation is applied regardless of the transmission mode.

To ensure that frame construction adheres to the specification of the standard, we have carefully verified that the time samples generated by the emulator match the example provided in Annex G of the IEEE 802.11 standard [18].

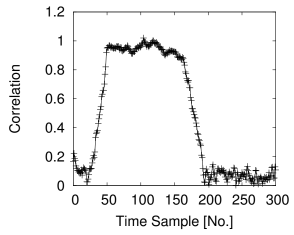
B. Channel Modeling

Once the frame has been generated, the sequence of complex time domain samples is passed on to the wireless channel module which allows chaining several propagation loss models such that the output of one model serves as input for the next model. Since IT++ provides a large collection of channel models, our implementation supports basic pathloss models such as Friis, Two-Ray Ground and LogDistance, large- and small-scale fading models as well as multi-tap channel models. In particular the latter ones allow the modeling of time and frequency-selective channels. Note that different receivers will experience a different channel response, i.e. for each receiver a copy of the originally transmitted sequence of complex samples is used to compute the response.

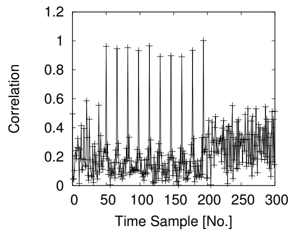
C. Receiver Modeling

When all channel effects have been applied, the sequence of complex time samples is passed up to the physical layer again (cf. Figure 1). There, the overall reception process can be distinguished into three stages (implemented through

[1]It is also possible for higher layers to specify the payload exactly. Support for his feature is included in NS-3.



(a) Auto Correlation method



(b) Known Values method

Fig. 5. Two different signal detection methods at work: auto-correlation and correlation with the known time samples. Both declare confidence (correlation approx. 1.0) that an incoming frame starts around the 50th time sample.

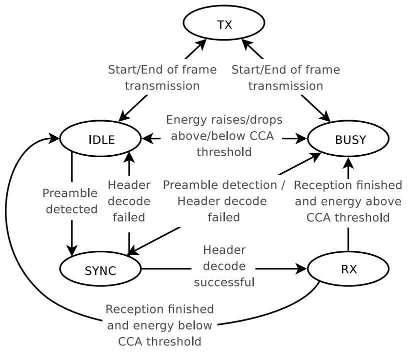

TX
Start/End of frame transmission
Start/End of frame transmission
IDLE
Energy raises/drops above/below CCA threshold
BUSY
Preamble detected
Header decode failed
Preamble detection / Header decode failed
Reception finished and energy above CCA threshold
SYNC
Header decode successful
RX
Reception finished and energy below CCA threshold

Fig. 6. The state machine of the physical layer emulator distinguishes between *Transmit*, *Idle*, *Busy*, *Sync* and *Rx* states.

four events in NS-3), as illustrated in Figure 4: first, the *Interference Manager* adds the sequence of complex time samples to its internal list of incoming frames, no matter how strong or weak the frame is. The objective is to keep track of all incoming frames and possible overlaps in order to enable the computation of cumulative signals — including white Gaussian thermal noise. Second, after the duration of the preamble has passed, the receiver tries to achieve signal detection and synchronization, i.e. "lock-on" to the correct time samples in the incoming frame. Typically, signal detection approaches use the repeating pattern in the preamble to achieve this. Our implementation uses the correlation techniques described in [19] for this purpose. Figure 5 shows the behaviour of two different signal detection mechanisms described in that work. High correlation values indicate a high level of confidence that a sample is part of a preamble. In the illustrated example both detection methods (auto-correlation and correlation with the known sequence) identify an incoming frame successfully around the 50th time sample. In addition to signal detection, the receiver performs coarse and fine frequency offset and channel estimation according to [20].

If signal detection and synchronization succeeded, the third stage, signal header decoding, is entered and in the end completed with OFDM demodulation, de-interleaving and convolutional decoding (either with soft or hard decision decoding). If the parity bit of the decoded header is correct, the process moves forward to its final stage which lasts until the end of the overall frame reception (cf. Figure 4). Again, a decision on whether all data symbols can be decoded successfully is made at the end of the stage, however, this time a comparison of the transmitted bit sequence with the received and decoded bit sequence yields the outcome. It is worth pointing out that the emulator uses the cumulative signals of overlapping frames as well as thermal noise as input for the processing in every stage. Consequently, the decisions have to be made at the end of each stage, otherwise possible interfering signals would be ignored.

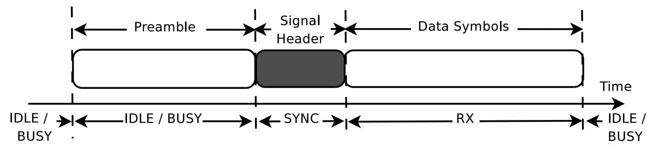


Fig. 7. The reception process with its corresponding stages and physical layer states: during preamble detection, the physical layer is still considered to be *Idle*. If signal detection is successful, the physical layer switches to *Sync* state and further to *Rx* if the signal header is decoded successful as well.

D. Physical Layer State Machine

The physical layer emulator and its behaviour is implemented as a state machine with five different states, as illustrated in Figure 6: *Idle*, *Busy*, *Transmit*, *Sync* and *Rx*. The *Idle* state is maintained if no signal header is successfully decoded and as long as the energy detected at the receiver stays below the CCA threshold (according to IEEE 802.11 [18] Section 17.3.10.5). As soon as the detected energy rises above the CCA threshold (while not having decoded a signal header successfully), the physical layer is marked *Busy* and the MAC layer is notified in order to block its own transmissions and to support the CSMA mechanism. The detection of a preamble leads to the *Sync* state and is eventually followed by the *Rx* state if the corresponding signal header is decoded successfully. When *Rx* state is active, the MAC is notified again to block its own transmission requests for the duration of the reception. Similarly, the transmission of a frame sets the physical layer in the *Transmit* state. Figure 7 depicts a typical case of the reception of a single frame and the corresponding states of the physical layer.

With some small additions to the above state machine, the physical layer emulator can also support the frame capture capabilities of modern transceiver chipsets. More precisely, this can be achieved by adding two state transitions from *Rx* to *Sync* and from *Sync* to *Sync*; these come into effect if the strength of a new incoming signal is sufficiently larger than the energy of the signal that is currently being processed.

IV. Results & Discussion

As a first approach in evaluating the impact of the physical layer emulator in NS-3 we have conducted a series of simulations using both, the plain vanilla and the emulator-augmented simulator variants. In addition, we have performed microscopic experiments on the CMU network emulator [3] to validate our implementation against real 802.11 OFDM-based communication chipsets, in particular against Atheros AR5212 based devices. The wireless testbed offered by CMU allows us to control the wireless channel in the same way as we do in our simulations and thereby a direct comparison of the emulated and a real physical layer.

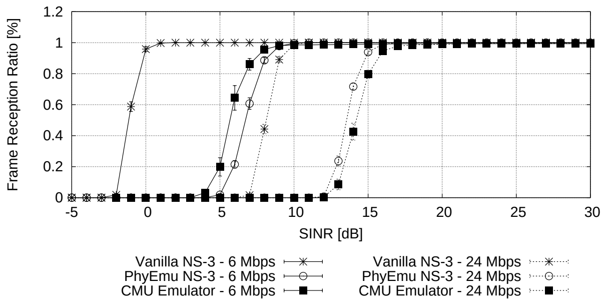


Fig. 8. Probability of successful frame reception w.r.t. average SINR when using either the abstract packet-level receiver models of NS-3 or the accurate physical layer emulation. The channel includes pathloss but no fading.

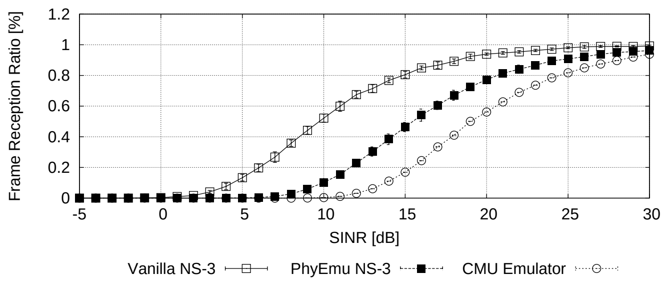
For the evaluation and validation, we measured and compared the probability of successful frame reception w.r.t. SINR when one single node is transmitting frames to a receiver in the absence of any interference. The simulations in NS-3 and the experiments on the CMU wireless testbed were configured according to the values specified in Table I[2]. We first studied the receiver performance in a scenario with non-fading radio propagation conditions, i.e. with a static pathloss only, followed by the consideration of a Rayleigh fading channel with different fading intensities due to different relative node speeds. Please note that due to the abstractions made in the plain NS-3 simulator, parameters such as relative node speed, channel bandwidth and transmission frequency are not used in the corresponding Rayleigh model. All our experiments were repeated 10 times, each run with 1000 frame transmissions.

Parameter	Value
Radio propagation model	Pathloss, Rayleigh fading
Relative node speeds	10, 30, 50 m/s
Channel bandwidth	20 MHz
Channel frequency	2.4 GHz
Symbol duration	4 μs
Frame size	500 bytes
Data rate	6, 24 Mbps
Channel estimator	Linear interpolation between pilot sub-carriers, according to [20]

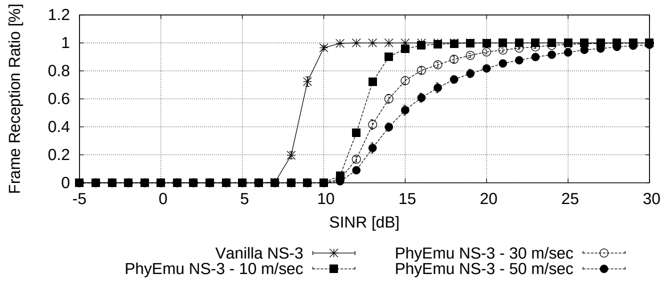
TABLE I
CONFIGURATION OF THE WIRELESS CHANNEL AND THE PHYSICAL LAYER FOR THE CONDUCTED SIMULATIONS AS WELL AS THE CMU NETWORK EMULATOR EXPERIMENTS.

Figure 8 shows the observed frame reception ratio w.r.t. SINR for two different data rates and the case when no signal fading is present. As we can see, all curves share a similar slope and the curves by the physical layer emulation in NS-3 and the curves by the CMU wireless testbed show a reasonable match, being only 1 dB apart from each other. Further, the results provided by plain NS-3 are significantly better, in the range of 7-8 dB, than the results provided by the other two approaches. To answer the question whether plain NS-3 provides overly optimistic results, we ran additional simulations of the physical layer emulation variant of NS-3 and discovered that the usage of a frequency offset estimator instead of the linear estimator leads to a performance improvement of 7 dB. So, we assume that the estimation method used by the Atheros AR5212 chip is less effective than our frequency offset estimation technique.

[2]Due to space restrictions we show results for 6 MBps and 24 Mpbs only



(a) SINR values derived from configured channel pathloss



(b) SINR calculation based on perfect simulator knowledge

Fig. 9. Probability of successful frame reception w.r.t. SINR when using a datarate of 24 Mbps and a Rayleigh fading channel: a) if the SINR is derived from the configured pathloss, both NS-3 variants as well as the CMU emulator produce similar reception curves (regarding the slope). b) if the instantaneous SINR of each individual frame is used (not provided/given by the CMU emulator), the vanilla NS-3 variant differs significantly from the physical layer emulator and can not distinguish different relative speeds.

If Rayleigh fading is considered, along with the pathloss, the slope of the reception curves are not as steep as without fading but still shares a similar gradient, cf. Figure 9(a) for a datarate of 24 Mbps and a relative node speed of 10 m/sec. It seems as if the vanilla, packet-level variant of NS-3 provides a sufficient amount of accuracy. However, it is important to focus on detail: for Figure 9(a) we computed the SINR after the channel pathloss had been applied and before Rayleigh fading was added. Thus the computed SINR does not reflect the instantaneous SINR experienced by individual frames. If we plot the reception ratios w.r.t. the instantaneous SINR instead, cf. Figure 9(b), we can see that there is a significant difference between the packet-level variant and the physical layer emulation. Further, the packet-level variant of NS-3 is not able to distinguish between different fading intensities.

The above simulations also provide some insight into the additional memory and computational requirements involved in the adoption of the emulator in NS-3. The memory overheads are mostly due to the fact that each frame is associated with an array of time samples as produced by the sender. The size of this array varies depending on the frame size and the data rate used, but it includes at a minimum (for a single OFDM symbol of payload) 480 complex values or approximately an additional 960 bytes to the minimum 36 bytes frame representation of NS-3 (no payload and only minimal 802.11 MAC headers).

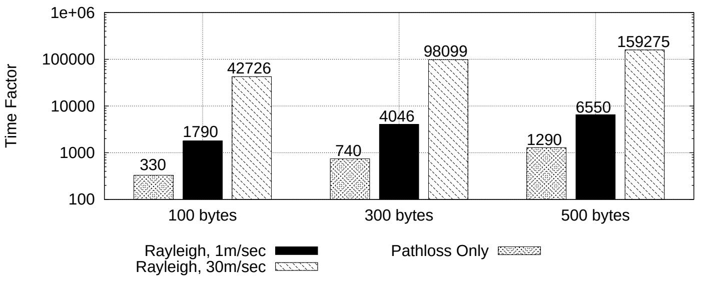


Fig. 10. Slow-down of a simulation when using the physical layer emulator variant of NS-3 with a data rate of 6 Mbps and with/without Rayleigh fading. The simulation time is compared to a baseline simulation with the vanilla variant of NS-3 in which only a static pathloss is considered.

Figure 10 further shows the additional computational effort required by the physical layer emulation (with one sender/receiver, a 6 Mbps worst case scenario w.r.t. performance, different frame sizes and channel conditions) — in comparison to a baseline simulation of a non-fading scenario using the vanilla NS-3 variant. As illustrated, the effort increases by a factor between 330 and 159 275 when using the physical layer emulation, depending on the frame size and whether a pathloss only, a slow- or a fast-fading channel is simulated. In particular the accurate implementation of a fast-fading channel with its huge number of required and to be generated random variates leads to a huge slow-down. Since our objective is an increased accuracy of the physical layer model by operating on complex time samples instead of working on frames as a whole, the performance optimization options are rather limited and can be categorized as follows: 1) optimization of a single frame construction and reception process by the usage of optimized signal processing algorithms, e.g. a parallel implementation of the (I)FFT or a lookup-table based convolutional encoder/decoder, and 2) optimization due to the networking aspect which would allow simultaneous receptions of different nodes to be processed in parallel. For both options, specialized math libraries such as the Intel Math Kernel Library and the AMD Core Math Library or emerging techniques such as General Purpose Computation on Graphics Processing Unit (GPGPU), e.g. based on NVIDIA CUDA or OpenCL, with up to hundreds of processing units on a single graphics chipset could provide a significant speedup.

V. Conclusions

In view of improving accuracy in widely used simulation tools, we have designed and implemented a physical layer emulator, which we integrated with the NS-3 simulator. The new hybrid simulator/emulator design adopts as the most fundamental simulation unit the signal time sample, which allows for the deployment of very detailed channel models as well as the creation of simulated transceivers that closely mirror the workings of their real hardware counterparts. We highlight the potential of the new simulator with a simple scenario and compare its computational and memory overhead to the traditional NS-3 model.

In the future we plan to implement additional state-of-the art channel estimation techniques and channel models as these appear in the literature. Since the adoption of the emulator implies no loss of generality in the simulator we expect it to be used for various other network research purposes.

Acknowledgment

The authors would like to thank Prof. Peter Steenkiste for the possibility to access the CMU Wireless Network Emulator testbed, Patrick Armbruster for his validation work on the Network Emulator testbed as well as Arash Tahmasebi Toyserkani for a lot of helpful discussions. Jens Mittag also acknowledges the support of the Ministry of Science, Research and the Arts of Baden-Württemberg (Az: Zu 33-827.377/19,20), the Klaus Tschira Stiftung, the INIT GmbH and the PTV AG for the research group on Traffic Telematics.

References

[1] "The NS-3 Network Simulator," http://www.nsnam.org/.
[2] "QualNet Network Simulator," http://www.scalable-networks.com/.
[3] G. Judd and P. Steenkiste, "Repeatable and Realistic Wireless Experimentation Through Physical Emulation," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 1, pp. 63–68, 2004.
[4] K. Tan, J. Zhang, J. Fang, H. Liu, Y. Ye, S. Wang, Y. Zhang, H. Wu, W. Wang, and G. M. Voelker, "Sora: High Performance Software Radio Using General Purpose Multi-core Processors," in *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2009, pp. 75–90.
[5] M. Cummings and S. Haruyama, "FPGA in the Software Radio," *IEEE Communications Magazine*, vol. 37, no. 2, pp. 108–112, 1999.
[6] "WARP: Wireless Open Access Research Platform," http://warp.rice.edu/trac/.
[7] "Network Simulator ns-2," http://www.isi.edu/nsnam/ns/.
[8] Q. Chen, D. Jiang, V. Taliwal, and L. Delgrossi, "IEEE 802.11 based Vehicular Communication Simulation Design for NS-2," in *Proceedings of the 3rd International Workshop on Vehicular Ad Hoc Networks*. Los Angeles, CA, USA: ACM, 2006, pp. 50–56.
[9] Q. Chen, F. Schmidt-Eisenlohr, D. Jiang, M. Torrent-Moreno, L. Delgrossi, and H. Hartenstein, "Overhaul of IEEE 802.11 Modeling and Simulation in ns-2," in *Proceedings of the 10th ACM Symposium on Modeling, Analysis, and Simulation of Wireless and Mobile Systems*. Chania, Crete Island, Greece: ACM, 2007, pp. 159–168.
[10] J. Ryu, J. Lee, S. Lee, and T. Kwon, "Revamping the IEEE 802.11a PHY simulation models," in *Proceedings of the 11th International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. Vancouver, British Columbia, Canada: ACM, 2008, pp. 28–36.
[11] A. Kuntz, F. Schmidt-Eisenlohr, O. Graute, H. Hartenstein, and M. Zitterbart, "Introducing Probabilistic Radio Propagation Models in OMNeT++ Mobility Framework and Cross Validation Check with NS-2," in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*. ICST, 2008, pp. 1–7.
[12] R. Merz, J.-Y. Le Boudec, and J. Widmer, "An Architecture for Wireless Simulation in NS-2 Applied to Impulse-Radio Ultra-Wide Band Networks," in *10th Communications and Networking Simulation Symposium*, 2007.
[13] "SWANS — Scalable Wireless Ad hoc Network Simulator," http://jist.ece.cornell.edu/.
[14] "GloMoSim — Global Mobile Information Systems Simulation Library," http://pcl.cs.ucla.edu/projects/glomosim/.
[15] M. Lacage and T. R. Henderson, "Yet Another Network Simulator," in *Proceeding from the 2006 Workshop on NS-2: the IP Network Simulator*. Pisa, Italy: ACM, 2006, p. 12.
[16] J. G. Proakis, *Digital Communications*, 4th ed., ser. in Electrical and Computer Engineering. McGraw-Hill, 2001.
[17] "The IT++ library," http://sourceforge.net/apps/wordpress/itpp/.
[18] IEEE Standards Association, "IEEE 802.11 LAN/MAN Wireless LANS," http://standards.ieee.org/getieee802/802.11.html, 2007.
[19] C.-H. Liu, "Design and Evaluation of Energy Detection Algorithms for IEEE 802.11a Systems," in *Radio and Wireless Conference*, August 2003, pp. 63–66.
[20] E. Sourour, H. El-Ghoroury, and D. McNeill, "Frequency Offset Estimation and Correction in the IEEE 802.11a WLAN," in *Vehicular Technology Conference*, vol. 7, September 2004, pp. 4923–4927.