Efficient Multi-Channel Simulation of Wireless Communications

Fabian Bronner and Christoph Sommer
Heinz Nixdorf Institute and Dept. of Computer Science, Paderborn University, Germany
{fabian.bronner,sommer}@ccs-labs.org

***Abstract*—Simulation is a key tool for studying new system designs, but its scalability is often limited by the complexity of underlying models. We investigate to what degree different channel models – in particular differently-complex signal representations and loss models – impact simulation performance. Measurements reveal that, if all effects relevant to typical vehicular network simulations are to be captured, use of a highly efficient signal representation that can exploit modern CPU features allows to cut its performance impact by an order of magnitude. Yet, measurements also reveal that in typical vehicular network simulations, runtime performance is dominated by that of loss modeling instead. To address this issue, we also present a universal approach that can speed up loss modeling. We show that this approach can improve the overall runtime performance of simulations by more than an order of magnitude with no loss in precision.**

I. Introduction & Motivation

Wireless network simulation is a prime method for performance evaluation of future system designs. This is particularly true when analytical evaluations are limited by the complexity of the system under study and field tests are limited by its scale – such as in the domain of vehicular networking [1], [2].

Of course, simulators should provide reliable data and results, making correctness one of their key attributes. Thus, a prime requirement is that the abstraction from real-world behavior of wireless communication is performed at an appropriate level for vehicular networking, particularly as far as modeling the wireless channel is concerned.

The many approaches to this abstraction that can be found in the literature differ widely in the level of fidelity afforded by them. With higher fidelity, the simulation results can capture a larger variety of effects that influence system performance in the real world, thus more closely approximating real-world behavior, and therefore allowing for more far reaching conclusions – often, however, at the cost of more complex models [3] and, hence, lower execution speed.

Thus, there is an apparent trade-off between scalability and fidelity [4]. Ideally, a simulator should produce accurate results within a short amount of time. Given a limited amount of time and resources to work on a specific problem, this increases the size of the available data set, which is particularly important as wireless simulations are often stochastic in nature (thus requiring a large number of independent simulation runs) [5]. As a side effect, faster execution speed also means that researchers get to spend more of their time on developing concepts, testing, and evaluating the simulation outcome.

In this paper, we first study the requirements of vehicular network simulation and the performance impact of different signal representations for multi-channel wireless network simulation (Section III), finding little difference (14 %) between their performance if accuracy is to be preserved. We then move on to describe a general way of speeding up multi-channel wireless network simulation (Section IV) and demonstrate speed-ups of more than an order of magnitude (4700 %) without loss in fidelity.

II. Related Work

In wireless communication systems, a sending node radiates a signal over its antenna. If receiving nodes detect this signal they can try to decode it in order to recover its information. Many simulators model this process by (i) checking whether the signal power is above a *sensitivity* or *detection* threshold and then (ii) checking the outcome of a probabilistic process based on the Signal to Interference plus Noise Ratio (SINR) at the receiver. This is based on the assumptions that a signal needs to be detected before decoding can be attempted and that the SINR is a prime indicator of the quality of the received signal. Assuming an Additive White Gaussian Noise (AWGN) channel, the SINR can be expressed as

$$\text{SINR} = \frac{\text{signal power}}{\text{noise power} + \sum \text{interference power}}. \quad (1)$$

In brief, the SINR captures not just raw signal power, but also other effects that might contribute to a signal being lost: First, noise, a phenomenon caused by parasitic effects within the radio channel or the receiving hardware itself. Second, interfering signals, mainly caused by multiple access of the transmission medium. For a higher value of the SINR, the receiver is more likely to successfully decode a received signal. Naturally, the interferer situation might not be constant during the reception process. Then, multiple SINRs have to be computed, a strategy sometimes referred to as piecewise SINR calculation [6].

One special kind of interference is Adjacent Channel Interference (ACI) in frequency-multiplexed channels [7]. It is rooted in the phenomenon that spectra of signals are not perfectly rectangular and limited to the intended channel, but (because of, e.g., physical limitations on filters or deficiencies in the transmitter) are present on other, neighboring channels as well. The power outside their main frequency range is smaller, but it can influence the success of signal receptions for the other affected channel.

978-1-5386-9428-2/18/$31.00 ©2018 IEEE

From Equation (1) it is clear that the radio channel between transmitter and receiver has a major impact on the signal. The spectral shape of a signal at the receiver side usually looks differently than the one originally radiated by the sender. This is caused by antenna gains and different path loss effects that attenuate a signal. This link budget is often expressed [8] as the sum (in dB) of the sending power P_t, the antenna gains G_t and G_r, and all loss effects L, as

$$P_r[\text{dBm}] = P_t[\text{dBm}] + G_t[\text{dB}] + G_r[\text{dB}] - \sum_x L_x[\text{dB}], \quad (2)$$

all of which can either be simple scalar quantities or multi-dimensional data structures.

To give an example, one basic loss effect model is the free space path loss model. It captures that, as the distance between sending and receiving node increases, the signal power decreases. The impact of this effect is also frequency dependent. With increasing distance d and frequency f the path loss becomes higher. Also given the speed of light c in the transmission medium, the most basic free space path loss model computes loss as

$$L_{\text{FS}} = \left(\frac{4\pi df}{c}\right)^2, \quad (3)$$

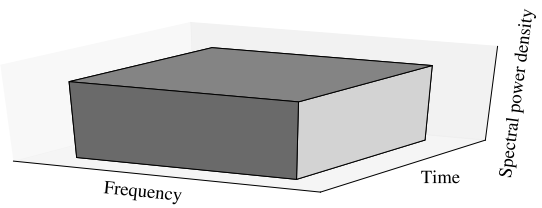
which can either be thought of affecting the received power uniformly across all frequencies (flat fading) or attenuating different frequency components differently.

There are various approaches to represent a wireless signal and the changes it undergoes according to Equation (2) within a simulator. They range from rather simple approaches that require less computational effort but suffer in accuracy to more complex representations that need more complex computations but offer a higher accuracy.

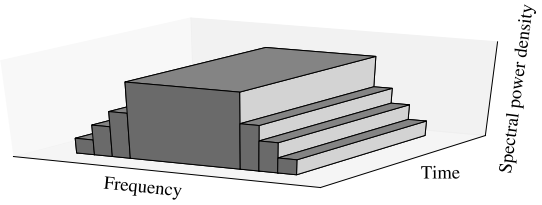
One simple approach is to describe a signal by just three major characteristics: a start time, a duration, and a sending power. So, it is a time invariant power level over a given time. Often, the signal holds further attributes such as the wavelength of the carrier frequency to support flat fading. This style of modeling a signal is, for example, used in GloMoSim to compute the impact of various path loss effects [9]. Yet, this way of modeling a signal makes it hard to efficiently reason about, for example, capacity gain of Frequency-Division Multiple Access (FDMA).

Describing the frequency dimension of a signal more fine grained can be done by explicitly modeling a signal as being present on a center frequency (and a given bandwidth around it). This representation allows to define rectangular shaped spectra for the signals as in Figure 1a. Again, the signal is time invariant. This style of modeling is, for example, one of the options offered by the OMNeT++ simulator's INET framework, which serves as the basis for a wide range of scientific work [10], [11]. While this allows for efficient modeling, it still hinders efficient reasoning about, e.g., ACI.

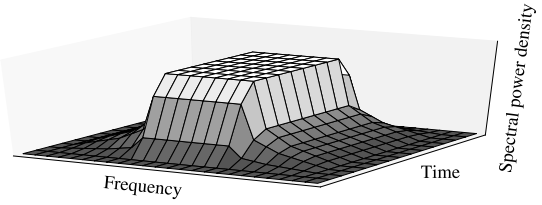
When the signal has a more complex spectral shape, one solution is to divide the frequency dimension into bands (frequency ranges) and describe the power within each band individually. These values then describe the spectral power density of the signal. As in the previous representations, the represented signals are time invariant. One example for such a signal description is given in Figure 1b. This style of modeling is one of those offered by, for example, NS-3 [12]. It allows free configuration of the number, center frequencies, and widths of the bands. NS-3 offers a set of different path loss effects like a two-ray model, shadowing, etc. and applies them on each band [13].



(a) time invariant static power with frequency and bandwidth relation



(b) frequency band model



(c) multi-dimensional model

Figure 1. Different approaches for signal representation in various simulators.

Even more complex signal shapes can be modeled by describing signals as a multi-dimensional function of spectral power density. Typically, this function is defined in the frequency and time domains, as shown in Figure 1c. This makes it possible to describe time-variant signals as well. This style of modeling is adopted, for example, in the OMNeT++ simulator's MiXiM framework [14], which was later incorporated into other simulation tools like Veins or the aforementioned INET Framework.

III. Performance Impact of Signal Representations

We study the performance impact of signal representations based on the fully-featured multi-dimensional model (shown in Figure 1c) as well as a best-case model, a minimalistic representation that still fulfills the aforementioned goals of vehicular network simulation. We argue that a signal should be able to capture the following aspects:

- **Timing**: A signal has a lifespan; it is only present for a given time period. This means that there are two major attributes that describe the timing behavior of a signal: start time and duration. Put another way, for a given timestamp t, a signal is present for $t_{\text{start}} \leq t < t_{\text{end}}$.
- **Signal strength**: Each signal has an individual power, depending on the transmit power chosen by the sender (e.g., according to ETSI ITS-G5 rules), antenna gains, and path loss effects. Note that this does not need to be a time-varying quantity: For all established wireless technologies for vehicular networking, the signal power is constant during transmission (though this might mean modeling time-stepped signals as an aggregate of multiple signals of one power level each).
- **Multi-channel**: A signal contains information on which frequencies it is present, allowing a simulation to explore efficiency gains of using more than one frequency channel, e.g., according to IEEE 802.11p.
- **Adjacent channel interference**: Signals might cause parasitic power on neighboring channels that interfere with transmissions in such channels, particularly important in multi-channel wireless networks that span large distances (near-far problem).
- **Frequency selective fading**: The attenuation of different path loss effects can affect different frequency components of the signal to different degrees, e.g., for modeling the compound impact of multiple hard-hitting, but selective loss effects such as shadowing by buildings and vehicles.

These needs are closely aligned with the signal representation chosen by the NS-3 simulator discussed previously and illustrated in Figure 1b. When its frequency bands are perfectly aligned (which they are by default), there are no gaps. Values outside the lowest or highest frequency band have an undefined power density.

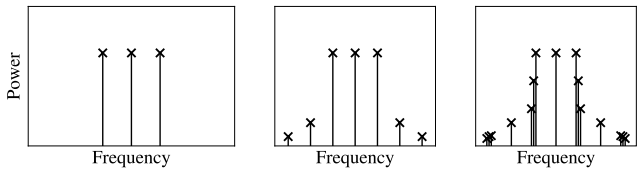
We go one step further in simplifying the signal representation and allow the user to only retrieve power densities at fixed frequencies (meaning that bandwidth is now only implied, not explicitly stored). This is possible as a model developer will know the exact points at which the spectral power density for signal processing must be known. Addressing an undefined frequency results in an error.

Assuming there are N defined points for querying spectral power densities at a fixed set of frequencies F, with

$$\mathbf{F} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_{N-1} \end{bmatrix}, \tag{4}$$

any signal S can be represented by only a start time, a stop time, and a vector of power values representing spectral power densities at the corresponding frequencies, with

$$\mathbf{S}_{t_{\text{start}},t_{\text{end}}} = \begin{bmatrix} S_0 \\ S_1 \\ \vdots \\ S_{N-1} \end{bmatrix}. \tag{5}$$



(a) no ACI (b) with ACI (c) higher resolution at transition to ACI part

Figure 2. Example spectral shapes of different signals.

This allows quick arithmetic with signals, incurs small memory overhead, and opens up the possibility of exploiting vector operations of modern CPUs.

We impose no limitations regarding number of frequencies, their values, and the space among them. So, the user has a lot of options to describe the frequency domain. Figure 2 illustrates different examples of how to represent a signal's spectrum. In Figure 2a the signal has the same power levels at all defined points, e.g., because it has a rectangular shape. The representation in Figure 2b contains additional frequencies, e.g., to describe the effect of ACI. For a higher resolution at the border frequencies of the main signal, in Figure 2c the frequencies are denser at those transitions.

A. Calculations on Signals

Moving from data stored for representing a signal to computations performed on them, the following operations are commonly implemented:

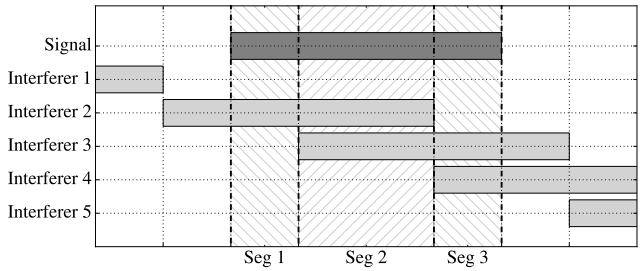
a) Modification of a signal: Different effects cause an increase or decrease of the signal power. These attenuations can either be the same for the whole frequency domain or frequency dependent. In order to apply attenuation factors on signals, the power density value for a specific frequency index is multiplied with an attenuation factor. This methodology allows to perform frequency dependent attenuations on signals. To model a uniform attenuation, the same factor can be applied on all values.

Note that there is no fundamental computational difference between applying antenna gain models or loss models to a signal – either model can both amplify and attenuate signals.

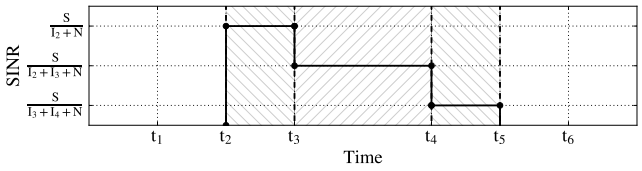
b) Querying signal or channel power level: For checking whether a signal exceeds the threshold for triggering a decoding attempt, it is necessary to be able to inquire about its power level. If multiple signals are present at the same time at a receiver, a common use-case is to compute their total power to decide whether a channel is to be considered idle or busy.

As it is only possible to define constant signals, multiple of them cannot be summed without loss of data. Their instantaneous power levels at a given time, however, can.

c) Querying SINR: In order to decide if a signal can be received or not, the receiver needs to compute the SINR for the corresponding signal. This involves the division among two signals and introduces the necessity to represent a constant, homogeneous noise floor.



(a) an exemplary situation of a signal and multiple interferers



(b) computed SINRs at a single frequency of the spectrum

Figure 3. Example of segment-wise SINR estimation.

During the time span of the signal, the interferer situation might not be constant. Coupled with enforcing non-time-varying power levels in a signal, this calculation is therefore a bit more involved.

For each interferer situation – called segment in the following – a new SINR calculation is required. The algorithm determines the segments by computing all *changes* during the reception of the signal. There are two types of such changes: whenever an interfering signal starts and whenever it ends. All changes are sorted by time with the earliest one first.

The algorithm initializes a variable representing the sum of all interference of a segment. Its initial value is the spectral sum of all interference that is present when the signal starts. With this information, it is possible to compute the SINR for the first segment. Next, there is an iteration over all changes left. If multiple changes have the same time stamp, all of them are applied at once. Each time the algorithm evaluates a change – or multiple in case of equal timestamps – there is a new segment which requires to calculate a new SINR.

Each segment thus represents a constant interference situation at the receiving node. For each defined frequency the signal power density is divided by the sum of interference plus noise.

Figure 3 illustrates this procedure by way of an example: A signal at t_2 is suffering from multiple sources of interference, starting at t_1, t_3, and t_4 respectively (Figure 3a). The algorithm splits the signal into three segments, each with its own SINR (Figure 3b).

It is now straightforward to calculate, e.g., bit error rate, segment error rate, or packet error rate for a given transmission. The most direct approach might be to search for the lowest SINR across all segments and frequency components and derive a packet error rate from that. So, for a signal that has its usable data spread over n frequencies and that requires m segments for SINR calculation, the lowest or *worst case* SINR is calculated as

$$\text{SINR}_{\text{worst case}} = \min_{m} \min_{n} \left(\frac{S_{\text{signal},n,m}}{S_{\text{noise}} + \sum S_{\text{interferer},n,m}} \right). \quad (6)$$

B. Speedup of Efficient Signal Representation

For evaluating the performance of different signal representations, we use Veins [15]. It is an Open Source[1] vehicular network simulation framework based on the OMNeT++ discrete event simulation kernel and relies on SUMO [16] for simulating road traffic.

We run the exact same simulation with two different signal representations. The first signal representation is the minimalistic one described above, which we implement on top of Veins 4.6. We compare its performance with that of the fully-featured multi-dimensional approach of MiXiM which is also included in Veins 4.6. As discussed (and shown in Figure 1c), this signal representation describes a signal by arbitrary points in the time, frequency, and spectral power density dimensions. Interpolation is used to obtain spectral power density values at arbitrary points.

To investigate the speed afforded by both signal representations, we track the amount of simulation time that could be executed within one real (wallclock) time second. We sample this value every wallclock second and average over all collected samples to derive the average execution speed. We call this metric *speed* in the following.

All measurements take place on computers equipped with an Intel Core i7-2600 CPU and 16 GByte of main memory, running Linux kernel version 4.4 of an Ubuntu 16.04 operating system. The software platform consists of OMNeT++ 5.1.1 and SUMO 0.30.0. All log data is either turned off or written to a RAM disk to mitigate I/O lag.

As our benchmark, we run a typical vehicular networking simulation study: We simulate vehicles driving in a city and periodically broadcasting 42 Byte messages (*beacons*) with a transmit power of 20 mW. Each simulated vehicle performs all required computations to decide whether it attempts decoding a beacon (sensitivity threshold −89 dBm) and whether it can successfully decode a received beacon. We perform this experiment with different beacon rates, that is, we choose different time intervals in-between a vehicle generating one beacon and the next.

The used scenario is that of traffic in the city of Paderborn [17], a medium-sized city with different types of environments like freeways, outlying housing areas, and the inner city. It thus includes a large number of radio obstructions.

For clarity, we configure only two loss models: a free space path loss model and an obstacle shadowing model [8] to account for loss by buildings. Therefore, there are two attenuation factors that impact signals. We configure the simulation to a length of 240 s, increasing the number of vehicles linearly from 210 to 1050 vehicles, each following realistic routes for a time of 8:00 AM. We run each simulation ten times with different seeds for the pseudorandom number generator.

[1] http://veins.car2x.org

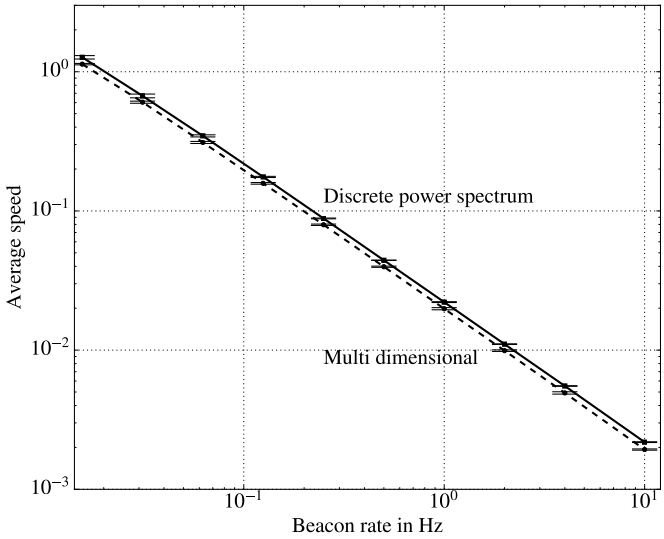
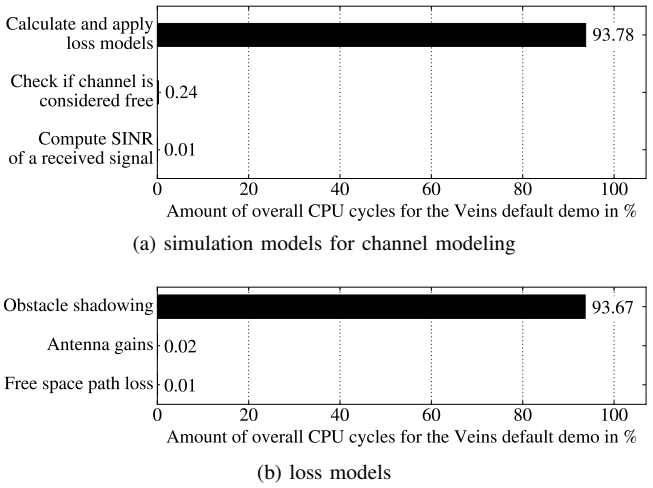


Figure 4. Average speed (computed simulation seconds per real time second) with confidence intervals (99 %) of the Paderborn scenario when using the multi-dimensional and discrete spectrum power model for signal representation in Veins for different beacon rates.



(a) simulation models for channel modeling

(b) loss models

Figure 5. Results of profiling individual simulation components (by fraction of total CPU cycles).

Figure 4 illustrates our results. Here, and in similar diagrams in this paper, we plot the mean speed and the confidence interval of the mean for a 99 % confidence level.

When comparing the full-featured (multi-dimensional) and the minimalistic (discrete power spectrum) signal representation, there is a small but reproducible offset between the speeds achieved with them: We compute an overall speed difference between approaches of approximately 14 %. For every beacon rate, the implementation using a minimalistic signal representation can run the scenario a little faster. Still, given that the two signal representations represent polar opposites in complexity, the difference is quite small.

For comparison, we also set up another scenario that had the obstacle shadowing model disabled (thus only modeling free space path loss). Here, the impact of the used signal representation on speed-up reaches one order of magnitude (1000 %, data not shown). This underlines that our findings are specific to simulations where loss model computations consume a substantial portion of time, in particular vehicular network simulation in cities.

IV. Improving the Performance of Multi-Channel Simulation Models via Thresholding

In order to further speed up simulations, we first want to precisely identify the components of a simulation that require the most computation time. We profile the execution of the aforementioned Paderborn simulation for a beacon rate of 1 Hz and the minimalistic (discrete power spectrum) signal representation. We employ the Linux *perf* toolkit which makes use of hardware counters to enable profiling. Its userspace tools enable us to periodically record which method of a binary is currently being executed. This way, the obtained results approximate the share a given method has in the total number of CPU cycles spent for a simulation run. After the execution of the simulation, perf can produce an output indicating this share of each method. It is possible to retrieve the share of each method itself and the share including all called methods. This makes it possible to determine which methods consume most of the CPU cycles for a given simulation.

Figure 5 plots the measured share of methods. It shows the ratio of the CPU cycles spent within a given model in relation to the total number of CPU cycles spent for executing the whole simulation. These methods do not call each other, so the shares do not overlap.

As can be seen, the method computing signal attenuation consumes by far most of the CPU cycles. Investigating this method in more detail as done in Figure 5b reveals that each loss model consumes a different amount of CPU cycles. The most expensive attenuation model by far is obstacle shadowing.

A straightforward next step would be to optimize this loss model (e.g., using quadtrees, pre-generating collision maps, caching, or similar methods) [18]. However, instead of improving the computation of this specific model, we aim for a general approach.

In more detail, we aim to skip the computation of models whenever possible. In order to understand in which way we can achieve the avoidance of attenuation model computations, we quickly repeat what a simulated receiver needs to know about a signal or channel:

(i) is a signal below a sensitivity threshold,
(ii) is a channel free (based on all present signals), and
(iii) what is the SINR of a transmission.

Commonly, answering these questions requires the position and orientation of the sender and receiver to compute attenuation factors, often involving independent streams of pseudorandom numbers. Therefore, each receiving node computes and applies the loss models individually.

Following the approach outlined in Equation (2), loss calculations in wireless network simulation are often performed by iteratively applying all configured loss models to each

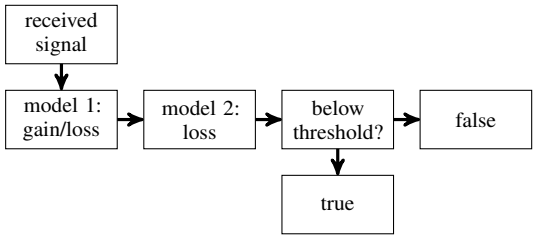


Figure 6. Common approach to checking whether the receive power level at a receiver exceeds a configured threshold in wireless network simulation: Application of all configured loss models to each received signal to derive its power level and checking the calculated power level against the configured threshold.

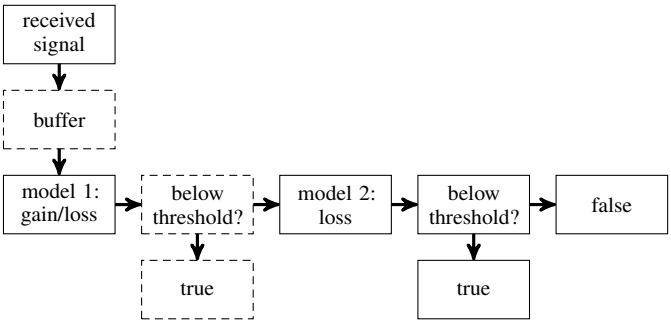


Figure 7. Steps performed in the proposed *thresholding* approach when receiving a signal: On-demand, iterative calculation of loss models to check reception power against a threshold – with short-circuit evaluation once a definite result can be determined. Steps added by the thresholding approach are highlighted.

received signal to derive its receive power level. These receive power levels can then be checked, e.g., against a configured threshold to see if a signal meets the receiver's sensitivity criteria. The diagram in Figure 6 outlines this procedure.

Based on the results illustrated in Figure 5 (loss models are costly, but to a varying degree), we advocate to instead employ a modified algorithm which we call *thresholding*. Its main idea is based on that of *short-circuit evaluation*, that is, skipping one or multiple of its computations if this does not affect the requested result. It thus avoids the calculation of as many loss models as possible. We describe its design below.

For the mechanism introduced in this section, it becomes necessary to group loss models into two categories:

(i) loss models that truly only decrease signal power (free space path loss, obstacle shadowing) and
(ii) loss models that might result in a gain of signal power (like those considering constructive interference due to multi-path effects).

We note that, if the maximum gain of a model of the second category is known, it is straightforward to split it into two models: One applies this maximum gain to the signal and is computationally very cheap; the other functions as before, but subtracts this maximum gain value (thus now falling into the first category).

Reflecting back on the three things a receiver needs to know about a signal or channel discussed earlier, the first question can be answered by a true/false-statement. It does not ask for a concrete power value, which enables thresholding. The same goes for the second question. The third question involves only a subset of signals. For some signals, e.g., for those below the sensitivity threshold of the receiver or those that are received while sending, it is not required (or even not possible) to compute an SINR. We detail how thresholding allows the three questions to be answered more efficiently below.

a) Determining whether a received signal is above the sensitivity threshold: Figure 7 illustrates the proposed *thresholding* approach by way of an example. A received signal only has loss models applied when their results are needed to determine the result of, in this case, a power threshold check. While this delayed execution already opens up the possibility of promise/future based programming (and, hence, exploiting multi-core CPUs), the true strength of this approach lies in the fact that not always all loss models need to be applied to arrive at a result. When a decision should be made on a signal, the first loss models that need to be applied are the ones that might result in a gain of received signal power. Already then, however, there are two possible results for a first threshold check. The signal power is either already below the threshold or still above or equal. For the first case it is not necessary to apply any further loss models as the remaining loss models can only cause the power to further decrease. For the second case (the power still exceeds the threshold or is equal), the next loss model is applied and the process is repeated.

This procedure might continue until all loss models are applied, allowing it to yield a definite result of *above threshold*. Sometimes, however, the algorithm can abort the calculation sooner – with a definite result of *below threshold* as soon as any loss model takes the received power below the threshold. This methodology illustrates that calculating the received power value is not necessarily required to answer whether a signal exceeds a given threshold or not.

Algorithm 1 Channel free decision when using thresholding.

Input: List of all received signals $\mathbb{S}$ at given time
Input: Ordered list of all loss models that might increase received power $\widehat{\mathbb{L}}$
Input: Ordered list of all other loss models $\mathbb{L}$

```
for all L in L̂ do
    Alter all S in 𝕊 according to L
end for
for all L in 𝕃 do
    Compute received power of sum of 𝕊
    if power < threshold then
        return true
    end if
    Alter all S in 𝕊 according to L
end for
Compute received power of sum of 𝕊
return power < threshold
```

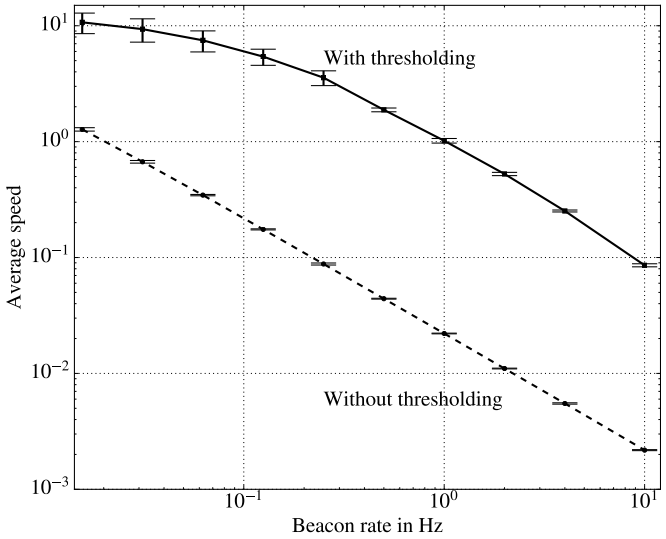


Figure 8. Average speed (computed simulation seconds per real time second) with confidence intervals (99 %) of the Paderborn scenario when using the multi-dimensional and no thresholding and the discrete spectrum power model with thresholding in Veins for different beacon rates.

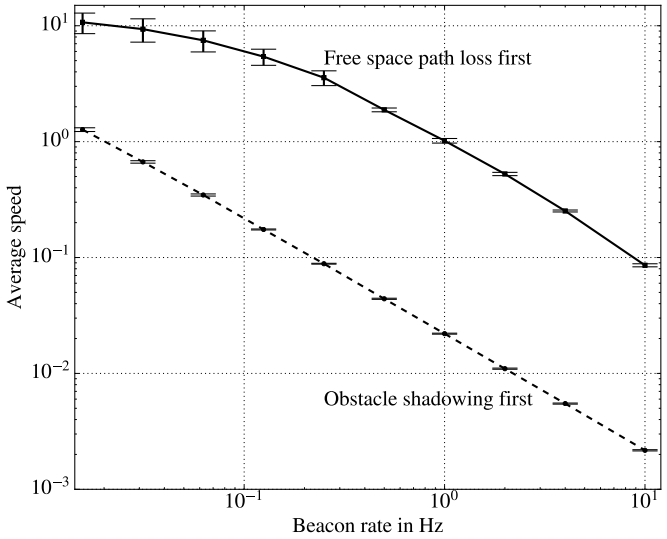


Figure 9. Average speed (computed simulation seconds per real time second) with confidence intervals (99 %) of the Paderborn scenario with enabled thresholding mechanism for different beacon rates, but two possible orders of loss models.

Note that the thresholding approach differs from the commonly-used shortcut of selecting a fixed maximum distance for a signal to travel in the simulation. While this approach can also speed up simulations, it only does so for signals that are too weak to even cause interference, not merely too weak to be decodable.

b) Determining whether the channel is free: The second question raised earlier – checking if a channel is free – requires the calculation of the sum of multiple received powers and checking it against a threshold. Thresholding can be helpful for this task again. The strategy is outlined in Algorithm 1. Starting with no loss model applied on all involved signals, the algorithm first applies all loss models that might increase the received power of the mix of signals currently on the channel. It then checks the received power against the threshold, short-circuiting the evaluation if the received power is below the threshold. In a next step, the first loss model is used to decrease the power of all signals currently on the channel and the process is repeated. This procedure continues until the received power has fallen below the given threshold or until all loss models have been applied.

Note that a simulator often first investigates whether any of the signals currently on the channel might trigger the decoding process to start before the simulator investigates whether the channel is free. Thus, a subset of signals will already have been altered to have a subset of loss models applied (see Figure 7). Naturally, these calculations need not be performed again, saving further computation time.

c) Computing the SINR value: For the final SINR computation, the signal and all interfering signals need to have all loss models applied. Afterwards, the SINR computation is done as described previously. Yet, as many of these computations have already been performed for any of the previous steps – and as many signals could already be ruled out as not triggering a decoding process – this step also benefits from the presented thresholding approach.

V. Results

Implementing the thresholding approach implies performing a number of additional operations during simulation, such as maintaining a list of which loss models have already been applied to which signal and performing a higher number of comparisons between signal power and threshold.

It is thus not immediately obvious that thresholding can be beneficial to the runtime of vehicular network simulations. We therefore investigate the impact of the described thresholding mechanism in detail.

For this, we extend the previously described implementation of a minimalistic signal representation for Veins 4.6 with a (purely single-threaded) implementation of the thresholding approach. We then run the previously described simulation study of a vehicular network operating in Paderborn again, now comparing the simulation performance with and without the implemented thresholding approach.

Figure 8 illustrates that (on top of any gain that could be delivered by choosing a minimalistic signal representation) applying the thresholding approach to channel modeling allows the simulation to run up to 4700 % faster. This speed-up is particularly noticeable for high beacon rates, beyond the point where the simulation time of regular simulations advances at approximately the same rate as wallclock time. Here, computation of loss models is the limiting factor of simulation speed – particularly for the investigated city scenario where densely-packed buildings obstruct radio communication and thresholding can avoid particularly expensive calculations.

We back up this claim by conducting a new set of simulations where we reversed the order of configured loss models, so that the most computationally expensive model (obstacle shadowing) is executed first.

Figure 9 illustrates the results of this experiment. As can be seen, this order of execution saves considerably less time. In fact, results are on par with those obtained in a simulation without the thresholding approach applied. The reason lies in the big difference in execution time between one loss model and the other.

Thus, for this simulation study of an environment dominated by radio obstacles, there is exactly one optimal order in which loss models should be applied when following a thresholding approach. For other simulation studies (particularly when using different loss models) the optimal order might not be as straightforward to determine and might differ among simulated nodes. Still, even with the exact opposite order of execution, the performance of the simulation is no worse than that of a variant without the thresholding approach.

VI. Conclusion

The channel models of wireless network simulators differ widely in how they represent wireless signals – from simple scalar models to complex multi-dimensional spectral power density distributions. This affords different levels of fidelity.

In this paper, we studied the requirements of vehicular networking to establish criteria that such representations ought to fulfill. We then moved to investigate the impact of different, but suitable representations on the runtime performance of simulations by comparing two extreme cases of such models: First, a highly flexible, but computationally expensive multi-dimensional spectral power density representation. Second, a minimalistic non-time-varying discrete power density model.

We found that, while in certain cases choosing a different signal representation can account for up to a magnitude difference in simulation run time, in typical vehicular network simulations the impact is very limited. Instead, simulation run time is dominated by the computation of loss models. Further, there are differences of orders of magnitude in terms of computational effort among different loss models.

We therefore propose the *thresholding* approach to loss modeling in vehicular network simulations.

It is a generic approach that is based on the idea of short-circuit evaluation of a chain of multiple loss models, avoiding the calculation of loss models when their result cannot influence the simulation outcome.

We investigated the speed-up achievable with this universal method when applied to a typical vehicular network simulation and found it to be in the order of more than a magnitude (4700 %) with no loss in precision.

Our method can be combined with improvements of individual loss models to achieve even better results.

As future work, we plan to build on the insight that speed-up is now dependent on the order that loss models are calculated, making the automatic ordering of loss models at runtime a promising next step.

References

[1] C. Sommer and F. Dressler, *Vehicular Networking*. Cambridge University Press, Nov. 2014.

[2] T. Yashiro, T. Kondo, H. Yagome, M. Higuchi, and Y. Matsushita, "A Network Based on Inter-vehicle Communication," in *IEEE Intelligent Vehicles Symposium (IV 1993)*, Tokyo, Japan: IEEE, Jul. 1993, pp. 345–350.

[3] E. Egea-Lopez, J. Vales-Alonso, A. Martinez-Sala, P. Pavon-Mario, and J. Garcia-Haro, "Simulation Scalability Issues in Wireless Sensor Networks," *IEEE Communications Magazine*, vol. 44, no. 7, 64–73, Sep. 2006.

[4] E. Ben Hamida, G. Chelius, and J. M. Gorce, "Impact of the Physical Layer Modeling on the Accuracy and Scalability of Wireless Network Simulation," *SAGE Simulation*, vol. 85, no. 9, pp. 574–588, Jun. 2009.

[5] G. Ewing, K. Pawlikowski, and D. McNickle, "Akaroa2: Exploiting Network Computing by Distributed Stochastic Simulation," in *European Simulation Multiconference (ESM 1999)*, Warsaw, Poland, 1999, pp. 175–181.

[6] M. Lacage and T. R. Henderson, "Yet Another Network Simulator," in *2006 Workshop on Ns-2: The IP Network Simulator*, Pisa, Italy: ACM, Oct. 2006.

[7] C. Campolo, C. Sommer, F. Dressler, and A. Molinaro, "On the Impact of Adjacent Channel Interference in Multi-Channel VANETs," in *IEEE International Conference on Communications (ICC 2016)*, Kuala Lumpur, Malaysia: IEEE, May 2016, pp. 2626–2632.

[8] C. Sommer, D. Eckhoff, R. German, and F. Dressler, "A Computationally Inexpensive Empirical Model of IEEE 802.11p Radio Shadowing in Urban Environments," in *8th IEEE/IFIP Conference on Wireless On demand Network Systems and Services (WONS 2011)*, Bardonecchia, Italy: IEEE, Jan. 2011, pp. 84–90.

[9] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: a Library for Parallel Simulation of Large-scale Wireless Networks," in *12th Workshop on Parallel and Distributed Simulation (PADS 1998)*, Banff, Canada: IEEE, May 1998, pp. 154–161.

[10] R. Nagel and S. Eichler, "Efficient and Realistic Mobility and Channel Modeling for VANET Scenarios Using OMNeT++ and INET-Framework," in *1st ACM/ICST International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools 2008)*, Marseille, France: ICST, Mar. 2008, pp. 1–8.

[11] T. Steinbach, H. D. Kenfack, F. Korf, and T. C. Schmidt, "An Extension of the OMNeT++ INET Framework for Simulating Real-time Ethernet with High Accuracy," in *4th International Conference on Simulation Tools and Techniques (SIMUTools 2011)*, Barcelona, Spain: ICST, Mar. 2011, pp. 375–382.

[12] N. Baldo and M. Miozzo, "Spectrum-aware Channel and PHY layer modeling for ns3," in *4th International ICST Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS 2009)*, Pisa, Italy: ICST, Oct. 2009, 2:1–2:8.

[13] T. R. Henderson, S. Roy, S. Floyd, and G. F. Riley, "ns-3 project goals," in *1st Workshop on NS-2: The IP Network Simulator (WNS2 2006)*, Pisa, Italy: ACM, Oct. 2006.

[14] K. Wessel, M. Swigulski, A. Köpke, and D. Willkomm, "MiXiM – The Physical Layer: An Architecture Overview," in *2nd ACM/ICST International Workshop on OMNeT++ (OMNeT++ 2009)*, Rome, Italy: ACM, Mar. 2009.

[15] C. Sommer, R. German, and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," *IEEE Transactions on Mobile Computing*, vol. 10, no. 1, pp. 3–15, Jan. 2011.

[16] D. Krajzewicz, G. Hertkorn, C. Rössel, and P. Wagner, "SUMO (Simulation of Urban MObility); An Open-source Traffic Simulation," in *4th Middle East Symposium on Simulation and Modelling (MESM 2002)*, Sharjah, UAE, Sep. 2002, pp. 183–187.

[17] D. S. Buse, C. Sommer, and F. Dressler, "Integrating a Driving Simulator with City-Scale VANET Simulation for the Development of Next Generation ADAS Systems," in *37th IEEE Conference on Computer Communications (INFOCOM 2018), Demo Session*, Honolulu, HI: IEEE, Apr. 2018.

[18] A. Mantler and J. Snoeyink, "Intersecting Red and Blue Line Segments in Optimal Time and Precision," in *Japanese Conference Discrete and Computational Geometry (JCDCG 2000): Revised Papers*, Tokyo, Japan: Springer, Nov. 2000, pp. 244–251.