Towards Comparable Network Simulations

Pengfei Di, Yaser Houri, Kendy Kutzner
System Architecture Group
Universität Karlsruhe (TH)
Karlsruhe, Germany
{di|houri|kutzner}@ira.uka.de

Thomas Fuhrmann
Computer Science Department
Technical University of Munich
Munich, Germany
fuhrmann@net.in.tum.de

Technical Report 2008-9
ISSN 1432-7864
Dept. of Computer Science
Universität Karlsruhe (TH)

Abstract

Simulations have been a valuable and much used tool in networking research for decades. New protocols are evaluated by simulations. Often, competing designs are judged by their respective performance in simulations. Despite this great importance the state-of-the-art in network simulations is nevertheless still low. A recent survey [6] showed that most publications in a top conference did not even give enough details to repeat the simulations.

In this paper we go beyond repeatability and ask: Are different simulations comparable? *We study various implementations of the IEEE 802.11 media access layer in ns-2 and OMNeT++ and report some dramatic differences. These findings indicate that two protocols cannot be compared meaningfully unless they are compared in the very same simulation environment. We claim that this problem limits the value of the respective publications because readers are forced to re-implement the work that is described in the paper rather than building on its results. Facing the additional problem that not all authors will agree on one simulator, we address ways of making different simulators comparable.*

1 Introduction

Often, the simulation of a new network protocol is preferred over its evaluation in testbed experiments. The reasons are manifold, e.g., the increased speed of getting evaluation results, the reduced hardware demands and thus the reduced cost, or the flexibility in the scenario definition. As a result, many network simulators have been developed over the last decades.

Today, *ns-2* has become the de-facto standard for network simulation. It is mostly widely used in academia. Kurkowski et al. [6] found that 44% of the simulations in their MobiHoc survey used ns-2 as network simulator. Its development began in 1989 as a collaboration between a number of different researchers and institutions. Meanwhile, a vast number of models for all kinds of network protocols have been written for ns-2. At the time of writing this paper, a popular ns-2 web site [9] lists 59 models for media access, routing, and transport protocols, as well as various topology and traffic generators.

GloMoSim has become particularly popular for the simulation of wireless networks. It provides various mobility models, a commonly accepted radio model, and many wireless ad-hoc networking protocol implementations. GloMoSim is written in *Parsec*, a derivate of C, which provides parallel discrete-event simulation capabilities. Even though GloMoSim is number two in Kurkowski's survey, it has only 10% market share. Even this fact alone demonstrates the severe market fragmentation in the area of network simulations.

OMNeT++ is another simulation tool that is free for academic use [13]. It is especially popular in Germany, where several groups have contributed so-called frameworks that provide various protocol implementations. OMNeT++ features a simple, object oriented design, which leads to good scalability. Therefore, we found OMNeT++ particularly well suited for performance evaluations of large networks. Still, outside a small community of OMNeT++ enthusiasts few people seem to know this tool at all.

Besides these three, many other simulators have been developed by the networking research community. All of them have their own applications, frameworks, libraries and modules. Their respective popularity greatly depends on the particular field of interest. Thus, it is almost impossible to address more than one or two research groups with your choice of simulator, unless you stick to *ns-2*.

In many cases, however, the choice of ns-2 is prevented by various problems. For example, we found ns-2 unsuited for the simulation of large networks, i. e. network with several thousands or even hundreds of nodes. Other groups have other reasons; but in the end, ns-2 has *only* 44% market share in Kurkowski et al.'s survey. The resulting fragmentation of the simulation tool market leads to an important problem: The reduced comparability of simulation results!

Typically, researchers and developers implement their newly proposed protocol in one particular simulator. They evaluate it in one or more scenarios (in this simulator), draw their conclusions, and publish a paper about their new protocol. As Kurkowski et al. have pointed out, often, such a publication does not give sufficient information about the simulation so that it is impossible to repeat it. But even if the paper gave all the required scenario information and the source code of the simulation was publicly available, the published result would be of limited use, because the chosen simulator environment is limited.

If one wants to extend the work, for example, use the proposed protocol together with another protocol or in a different scenario, it is almost impossible to do so, even if the source code of the simulation was publicly available. The reason is that the structures of the simulators differ significantly and thus the modules cannot (easily) be applied in another simulator. As a result, it is typically impossible to compare a bunch of newly proposed protocols with one another.

In this paper, we want to raise the awareness for this problem; and we want to report on ongoing work that aims at circumventing this problem. This paper is structured as follows: In section 2 we illustrate the comparability problem by an example study that we made using ns-2 and OMNeT++. In section 3 we advocate our proposal to use modules and messages as a fine-granular abstraction that allows the re-use of models and protocol implementations across different simulators. Section 4 discusses related work, and finally, section 5 concludes with an outlook to future work.

2 Comparing Different Simulators

As stated above, the comparability of protocols and simulation results is severely affected by the fragmentation of the simulator market. Typically, when ns-2 has been found insufficient for a particular study, any substitute will have a negligible market share. Thus the simulation results cannot be compared with another, competing protocol. As a result, simulation studies of newly proposed protocols can only present some evidence for its principal fitness. They cannot differentiate between different competing proposals.

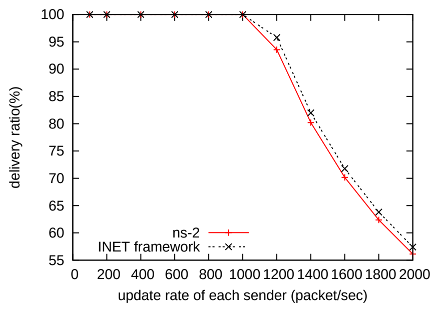
Let us illustrate this problem with an example: Researcher A presents a new routing protocol and publishes some evaluation results that she obtained with the OMNeT++ simulator. Researcher B publishes his new routing protocol with a performance measurement in ns-2. Both, A and B claim that their protocol outperforms the state-of-the-art protocols. Being aware of Kurkowski et al.'s critique, A and B have taken particular care of providing all the relevant information about their respective of the simulation settings, including the source code of their protocol implementation. As a result, other researchers can repeat their simulations and confirm their claim.

However, up to now, the relative performance of A's and B's proposal is still unclear. The results from the two papers cannot be compared directly because A and B chose different simulation environments. Assume, for example, that they use different MAC-layer modules which can influent the fairness of such a comparison.

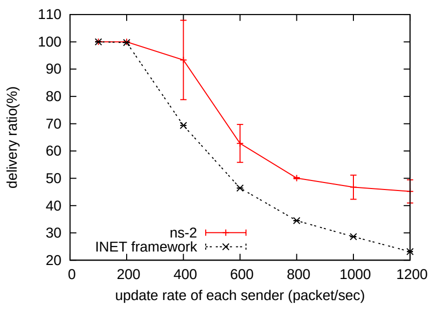
2.1 Differences Between ns-2 and the INET framework of OMNeT++

In order to illustrate this problem, we have compared of the IEEE 802.11 modules in the INET framework of OMNeT++ and ns-2.29. Obviously, the module implementations are based on the same standard. For the comparison, we set all the (available) parameters in the two modules to the same values. We modeled the application traffic with the help of a trace file that we adopted to both simulators: In the simulation, there is one stationary receiver and one or two stationary senders. Specifically the distance between sender and receiver is 50m, while the distance between two senders is 70.7m. The senders regularly send UDP packets with constant packet size to the receiver. In order to eliminate the influence of the routing layer, we studied scenarios that contain only 1 hop. Each run in this paper simulates 60 seconds. The figures show averaged values of 10 runs. Error bars (where applicable) show standard deviation. (For negligible small standard deviations, no error bars are plotted.)

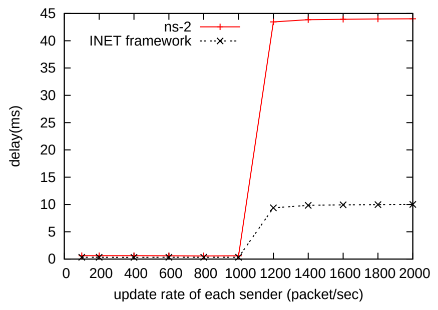
Despite the same node structure, the same protocol stack and the same traffic parameters (cf. table 1), the results from the different simulators deviate. The packet delivery ratio in the 2-node scenario can be considered identical in both simulators (cf. fig. 1(a)). But the delay in the 2-node scenario can only be considered identical when the update rate less than 1000 packets per second. Beyond this limit, the delay in ns-2 is significantly larger than that in OMNeT++ (cf. fig. 1(c)). In the 3-node scenario, the two senders are mutually hidden nodes. We see that the packet delivery ratio of OMNeT++ is much smaller than that in ns-2 (fig. 1(b)). The mean delay of OMNeT++ is similar to that of ns-2, but with very large deviation(fig. 1(d)). Most interestingly, the



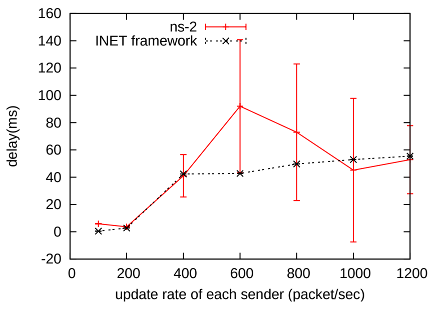
(a) 2 nodes



(b) 3 nodes (with 2 hidden senders)



(c) 2 nodes



(d) 3 nodes (with 2 hidden senders)

Figure 1. Performance comparison between 802.11 MAC modules in OMNeT and ns-2

results from ns-2 are erratic while those from OMNeT++ are not.

parameter	ns2/OMNeT++
application packet	20 bytes (UDP)
routing protocol	AODVUU
interface queue	50
data rate	11Mb
RTSThreshold	3000 (RTS/CTS off)
retry limit	7
cwMin	31
snirThreshold	10
signal attenuation threshold(OMNeT)	-71.843dBm(55m)
CSThresh_(ns2)	6.35631e-08W(55m)
radio sensitivity(OMNeT)	-71.843dBm(55m)
RXThresh_(ns2)	6.35631e-08W(55m)

Table 1. parameters in comparison between OMNeT and ns-2

In out comparison, we have paid attention that all configuration parameters were the same. But generally, such a difference can result from a different configuration. For example, both simulators use different default parameters: In ns-2, the bit-rate is set to 2Mb, and the RTS/CTS is set to ON; while in OMNeT, the default bit-rate is set to 11Mb, and RTS/CTS is set to OFF. More subtle is the fact that some parameters are in different units. For example, in ns-2, the receive threshold is set in Watt, while in OMNeT it is set in dBm.

In principle, these differences in configuration can be resolved with careful study of the simulator and module manual. But as we all know carefully reading the documentation is time-consuming and needs much attention. Especially, under pressure such subtleties can easily be overlooked.

In addition to these differences in the respective default configuration, there exist some differences in the implementations of both simulators that cannot (easily) be resolved:

1. Some parameters are not available in the configuration file but only in the source code. For example, in ns-2, the maximal contention window `CWMax` is set in the configuration file, with a default value of 1023. In OMNeT++ this parameter is defined as a constant of 255 in one of the source files.

2. Sometimes there is no corresponding parameter in the other simulator at all. For example, in ns-2, `longRetryLimit=4` is configured as the retry limit for data packets and `shortRetryLimit=7` is configured as the retry limit for a control packet. In OMNeT, there is only one `RetryLimit=7` for all the packets. In ns-2, `CWMin` is configurated as 31 for all the packets. In OMNeT, there are two distinct values, `CWMinBroadcast` and `CWMinData` for broadcast messages and data messages respectively.

3. The simulators use a different modeling approach for the parameters. For example, ns-2 and OMNeT++ have the

following different modeling approaches:

- **Propagation delay** — In ns-2, the delay is a constant defined in configuration file. In OMNeT++, the delay is a function of the nodes' distance.
- **Signal loss model** — $Pr \sim d^{-2}/K$ in ns-2 and $Pr \sim d^{-x}$ in OMNeT++.
- **Bit-error definition** — In ns-2 the frame is considered as having bit-errors if the distance between sender and receiver is between $RxRange$ and $CsRange$. In OMNeT the bit-error probability is a function of the distance, frame size and bit rate.
- **Collision model** — In ns-2, a collision between packets happens if $\frac{ReceivePower}{MaxNoise} > SNR$ while OMNeT++ defines that a collision occurred if $\frac{ReceivePower}{SumNoise} > SNR$.

Because of these implementation differences[1], it is impossible to make identical simulations without modifying the source code of either module. The latter would be time-consuming and error-prone; and it would further limit the comparability of the simulation results unless all relevant publications would use the same modifications.

2.2 Differences Within ns-2

As the development of a network simulator progresses, its code will change: bugs are fixed, code is refactored, and new features are introduced. Obviously, fixing a bug will modify the behavior of a simulation. But sometimes minor changes can have a significant impact on the result of a simulation.

In order to evaluate this potential problem, we set up simulations with the IEEE 802.11 MAC module in three different versions of ns-2: ns2.26 which was released in February 2003, ns2.29, and ns2.31. In our experiment, again, one or two senders regularly send 20 byte UDP packets to the receiver. We use the default MAC parameters where applicable. (See table 2 for details.)

As can be seen in fig.2, the different IEEE 802.11 MAC module versions perform identically with respect to the packet delivery ratio as well as the packet delay as long as there are no hidden nodes. If we reduce the carrier sense threshold to make the two senders mutually hidden nodes, the delay becomes erratic (fig.2(c) and fig.2(f))[3]. As a consequence, the results of the simulations differ, but only within the variance of the simulated value.

[1]These differences kept us from directly comparing the ns-2 model with the OMNeT++ Mobility framework. We could only compare ns-2 to the OMNET++ INET framework. However, we were able choose a scenario that allowed us to compare the INET and Mobility within OMNeT++, see section 2.3.

[3]Even after longer simulation time, the delay varies significantly. We think it is due to the incompatible implementation of backoff timer to the IEEE 802.11 specification [11].

parameter	ns2.xx
app. packet	20bytes (UDP)
routing protocol	AODV[2]
interface queue	50
RTSThreshold	0 (RTS/CTS on as default)
data rate	11Mb
CSThresh_	3.41828e-08W(75m)
RXThresh_	6.35631e-08W(55m), 3.41828e-08W(75m)

Table 2. parameters in comparison among diff. ns-2 versions

parameter	INET/MF framework
host	MobileHost/MFMobileHost
app. packet	20 bytes (UDP)
routing protocol	static table
mobility	no
RTS/CTS	off
bit rate	11 Mb
sat	-74.537 dBm(75m), -71.843 dBm(55m)
maxQueueSize	50 packets
sensitivity	-74.537 dBm(75m), -71.843 dBm(55m)
pathLossAlpha	2
snirThreshold	10 dBm

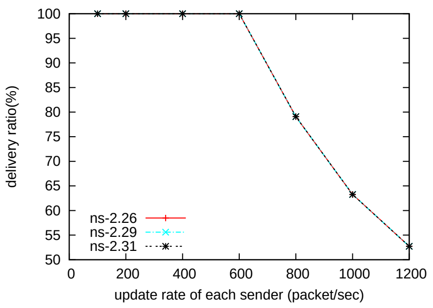
Table 3. parameters in comparison between INET framework and Mobility framework

Although these results cannot prove the absence of the presumed problem, they could not demonstrate it either. Thus, we can have some confidence that simulation results that have been obtained with these three module versions are indeed comparable.

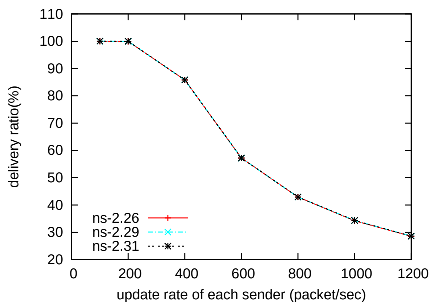
2.3 Different OMNeT++ Frameworks

Next, we compare different implementations of the same protocol in the same simulator. In OMNeT, several frameworks for the same protocol stacks have been developed independently. For example, both the INET framework and the Mobility framework contain an IEEE 802.11 model. They differ in their internal structure and the number and kind of their parameters.

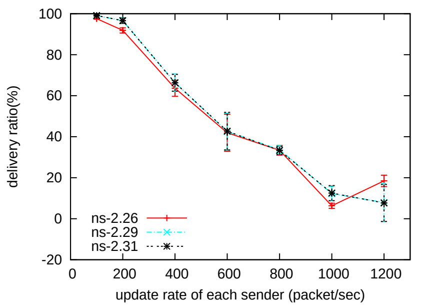
For example, upon encapsulation of a MAC packet into "‘airframe"’ packet, the Mobility framework adds an airframe header of 192 bit, while the INET framework does not add such an overhead. In our example, which uses 20 byte UDP packets, this is a significant difference. Most probably, many other such more or less subtle differences exist. But they are hard to tell without close inspection of the source code.



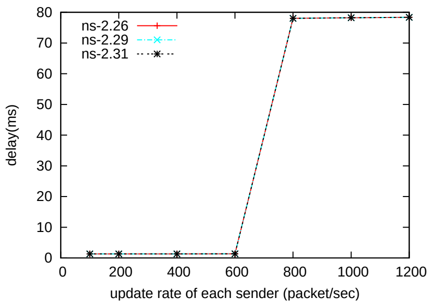
(a) 2nodes(1 sender CS:75m PR:55m)



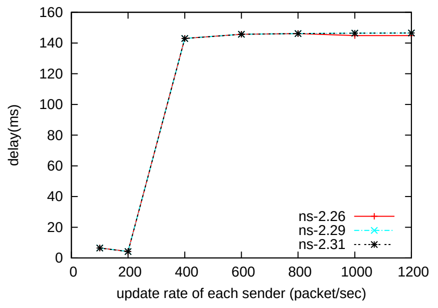
(b) 3nodes(2 senders CS:75m PR:55m)



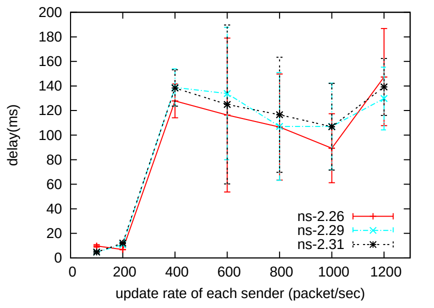
(c) 3nodes(2 senders CS:55m PR:55m)



(d) 2nodes(1 sender CS:75m PR:55m)

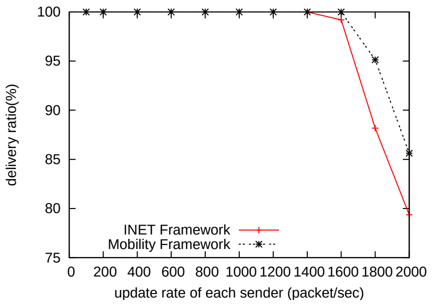


(e) 3nodes(2 senders CS:75m PR:55m)

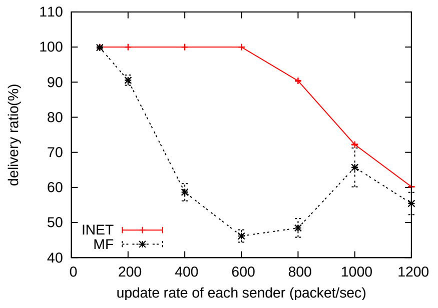


(f) 3nodes(2 senders CS:55m PR:55m)

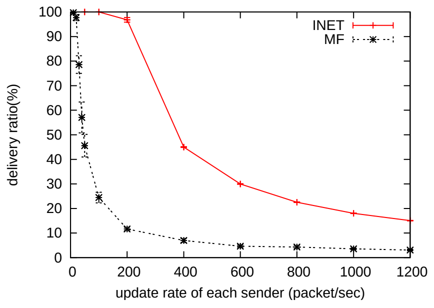
Figure 2. Comparing IEEE 802.11 MAC module in different versions of ns-2



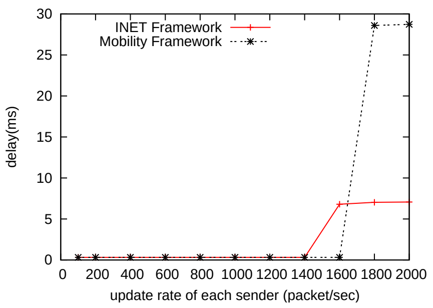
(a) 2nodes(1 sender sat:55m sensitivity:55m)



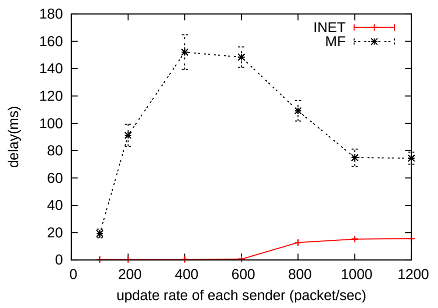
(b) 3nodes(2 senders sat:75m sensitivity:75m)



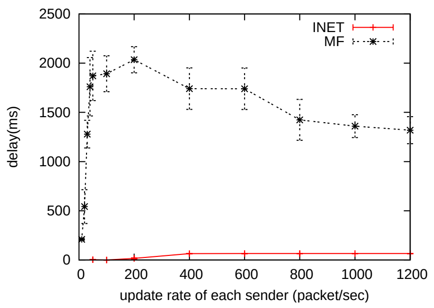
(c) 3nodes(2 senders sat:55m sensitivity:75m)



(d) 2nodes(1 sender sat:55m sensitivity:55m)



(e) 3nodes(2 senders sat:75m sensitivity:75m)



(f) 3nodes(2 senders sat:55m sensitivity:55m)

Figure 3. comparing IEEE 802.11 MAC module in different frameworks in OMNeT

The effect of these differences is dramatic: Fig. 3 shows the results of the simulations that we did with these two different 802.11 models. Again, we use the similar scenarios as used in section 2.2 with the parameters listed in table 3. Now, almost all the measured parameters deviate significantly. Only if there is no contention the result of the packet delivery ratio is similar for both modules (fig. 3(a)). In the two sender scenarios the INET framework based simulation seems to perform more than 10 times better than the Mobility framework based one. Such differences invalidate all protocol comparisons that would compare results from different frameworks.

2.4 One Module in Different Simulators

Once a module is written, it should work identically in all discrete event simulators. Ideally, the simulator would just be a machine that processes the events in chronological order. The events – and especially their causal relation – should be entirely determined by the modules' properties (and the pseudo random number generator).

In order to check this behavior of the same module in different simulators, we have wrapped the IEEE 802.11 module from the ns-2 into OMNeT++. Then we compared the results obtained in OMNeT against the results obtained in original ns-2.

The scenarios used here are the same as in table 2. As expected, we obtained almost identical MAC layer throughput in both simulators (fig. 4). We attribute the remaining small differences to a different implementation of the MAC buffer module, which has most effect in the hidden node scenario. Furthermore, the results are not identical because the two simulators use different pseudo random number generators: OMNeT uses the Mersenne Twister RNG [8]; while ns-2 uses the combined multiple recursive generator *MRG32k3a* proposed by L'Ecuyer [4]. Finally, the use of floating point data types for the time stamps might introduce further deviations when the simulators are run in different computing environments.

3 Towards Comparable Simulations

As we have shown in the previous section, it is typically not meaningful to compare protocol evaluations that have been obtained in different network simulators: Even such a well-studied protocol like IEEE 802.11 is implemented so differently in popular simulators that the results can vary by an order of magnitude. It seems evident that this problem will become worse when more complex simulations are concerned, for example, simulations that involve radio models, mobility models, traffic models, or that consist of several interfering protocols such as network layer and overlay routing protocols.

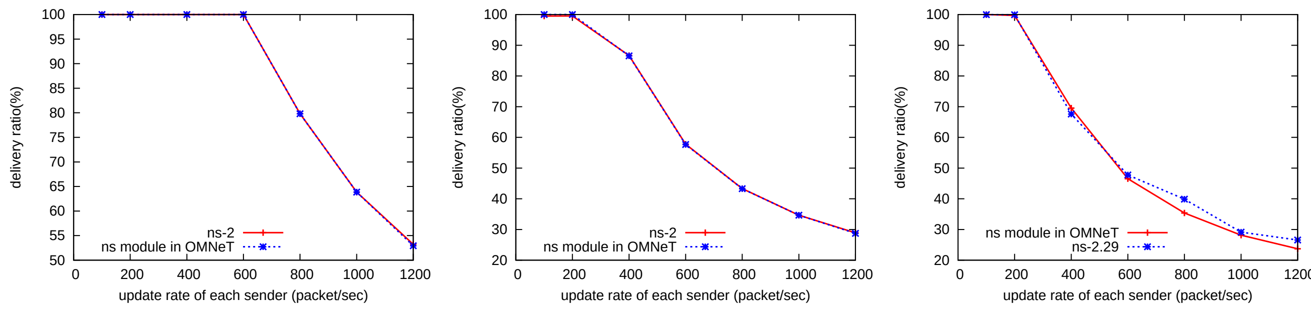
Luckily, our studies that we described in the previous section have also shown a potential way out of this comparability problem. As we have demonstrated, it is feasible to extract the implementation of a particular model or protocol such as IEEE 802.11 from one simulator and run it in another simulator. In this section we want to advocate this approach and recommend it as general way to obtain comparable network simulations. We claim that in fact the models and protocol implementations, not the simulators or frameworks are the right level of abstraction. A simulator should be used as a tool that executes the models and protocols. The simulator should not be used as reference of its own because it is not suitable to compare protocols.

In practice this means that the code that implements a model or protocol should be as independent of a particular simulation environment as possible. Here, the simulation environment includes both, the simulation engine and the other models and protocols such as the protocols in the layers below and above the respective protocol.

We found that the common denominator of all network simulators we studied are modules and messages. Here, a "'module"' means the implementation of a particular model or protocol. Modules interact with the simulation engine, for example, for enqueuing or processing events. Moreover, modules interact with one another, for example, along a packet's way through a protocol stack.

We claim that all these interactions can rather easily be expressed as the exchange of messages between the modules using the simulator as mediator. Therefore, we propose that modules use messages as the only way of communication. In particular, we propose that different modules should *not use* function calls into other modules or the simulator engine. The reason is that in our studies, we often found that modules that used function calls were significantly more difficult to port to other environments than modules that employed only messages. The latter could be used almost "'as is"'.Typically a simple wrapper was enough to translate the according messages into the respective other format. Moreover, we could also easily employ this approach with modules that were written in different programming languages. For example, *SWIG* and *Mono* provide OMNeT++ with special wrappers for modules written in other languages than C++ [2]. With this tool we could easily use Java and C# modules in OMNeT++, too.

As we proposed above, a module should only use the simulator's event queue. Besides the event queue most simulators also provide libraries with various convenience functions such as pseudo random number generators, statistical logging services, etc. Such libraries are extremely helpful in implementing a module. But as a drawback they are very simulator dependent, too. Therefore we propose to use these functions only with care. In particular, we propose to avoid functions that establish a communication



(a) 2nodes(1 sender CS:75m RX:55m) (b) 3nodes(2 senders CS:75m RX:55m) (c) 3nodes(2 senders CS:55m RX:55m)

Figure 4. evaluating the ns-MAC80211 Module in OMNeT(delivery ratio)

means between modules besides the event queue. For example, when the programmer of a framework uses a blackboard to convey additional information between a set of modules, we found it very hard to extract the individual modules from the framework. Moreover, in the cases we studied, we could always replace the implicit blackboard communication by explicit communication between the respective modules. Thus, we propose to avoid such special approaches and use messages as the only means of communication between the modules.

Note that we argue against frameworks because we believe that frameworks reduce the comparability of simulations. As we have shown above, the two popular OMNeT++ frameworks produce very different results. Hence, their existence further fragments the simulator market. We propose that authors of simulation studies should provide their models and protocol implementations in form of modules. These modules should be as self-contained as possible. They should communicate only explicitly via messages. Ideally, these messages should be as generic as possible. For example, when using addresses for a routing protocol, the programmer should abstract from peculiarities of certain address families such as IPv4 and assume only the properties that are actually need for their respective protocol implementation. This allows to re-use the code in other settings, too, for example in an IPv6 setting. Programmers should definitely refrain from mis-using IPv4 header fields to convey simulator-internal information, as we have for example, seen it in the GloMoSim implementation of AODV.

We are convinced that our proposed "‘modules-and-messages-only"’ approach will indeed allow others to reuse individual modules and combine them to obtain new results. Note that we haven't given out rigorous definitions of the implementation guidelines here, but rather some suggestions that the protocol developer might take care in order to make his model easy to be reused in more simulators later.

Currently, we explore the feasibility of this approach with a rather heterogeneous set of modules from different simulators. So far, this effort seems to be very promising.

4 Related Work

Various authors have already addressed the problem that different simulators can produce largely differing results for the same scenario.

Cavin et al. [3] present the incompatibilities among different simulators: OPNET, ns-2 and GloMoSim. Even for a simple flooding algorithm based on MAC 80211 models, there exists large quantitative and qualitative divergences. The authors have compared the different structures of these simulators, and think that the divergences can result from the abstractions and simplifications of the models from different simulators, which could contain errors or incompatibilities to the IEEE 80211 standard. However they didn't illustrate the differences.

Heidemann et al. [5] evaluate the effect of abstracted model with four case studies of wireless simulations. The abstraction can result in incorrect results, inapplicable results or correct results depended on the application's robustness to the error and sensitivities to the detail. They address that applications which are robust to error can tolerate abstract models of underlying layers, and additionally the visualization techniques can help pinpoint incorrect details.

Takai et al. [12] demonstrated the effect of different implementation of the physical layer on routing protocol (AODV and DSR) in ns-2 and GloMoSim. The physical layer modeling factors, which vary in different simulators, are: preamble length, interference model, fading model, path loss model and reception model. Eventually, they find the performance divergence of the same protocol between different simulators result from the different physical model. They give out that if the same modeling factors are applied, the divergence are negligible.

Reddy et al. [11] quantify the differences in the modeling of 802.11 MAC protocol in different simulators: GloMoSim, GTNetS and ns-2. They found that the simulators can produce identical results in the scenarios without medium contention when paying attention to the implementation discrepancies. In the scenario with medium contention, the ns-2 and GloMoSim have worse performance than GTNetS due to the incompatible implementation of NAV and backoff timer to the IEEE 802.11 specification.

Liu et al. [7] tried to validate ad-hoc routing simulations by executing the actual protocol implementation in a simulator and comparing it to real-world measurements. They found critical dependency on the physical model underlying the 802.11 MAC layer which resulted in large deviations between the various simulation runs.

Despite the fact that these problems have long been known, our studies show that the problem is still present not only in the different simulators but also in the current simulation frameworks. Moreover, our studies demonstrate that the simulation models have actually identical behaviors in different simulators, i.e., simulation results are depended on models but not simulators

Recently, several authors have again addressed the question of repeatability of simulations. Kurkowski et al. [6] showed that most publications in the MobiHoc conference did not give enough details to repeat the simulations. Andel et al. [1] given more examples for the lack of rigor in MANET simulation studies. They found 80%-90% of the publications to give insufficient information about the simulations that were used to obtain the presented results. Naicken et al. [10] discuss the same problem with respect to peer-to-peer research.

5 Conclusions

In this paper, we have addressed the issue of comparability of protocol evaluations that are done with different simulators. To this end, we have compared simple simulations that involved the IEEE 802.11 MAC modules in different versions of ns-2 and two different OMNeT++ frameworks. Analyzing the simulators including their source code, we have found that differences in the implementation of the simulators and frameworks do not allow to reproduce the simulation scenarios from another simulator. Furthermore, we have shown that even for scenarios where two simulators allow the choice of identical parameters, the different simulators lead to vastly different results. As a consequence, we conclude that protocol evaluations from different simulators are not comparable even when the authors use the very same simulation scenario.

Based on the experience from our study, we propose that modules, i. e. the implementation of a particular model or protocol, should be the level of abstraction on which different model and protocol implementations should be compared. We have demonstrated our idea by porting the ns-2 IEEE 802.11 MAC module to OMNeT++. There, we found it to produce almost identical results as compared to being run in ns-2. Wrappers in OMNeT++ even allowed us to integrate modules that were written in other programming languages such as Java or C#.

In order to simplify the re-usability of such modules we recommend that modules should only communicate by exchanging messages. We found that messages could be easily wrapped, thereby allowing the straightforward combination of modules from different simulators. Modules that used function calls or communication side-channels such as a blackboard were rather hard to isolate from their original environment. Thus we believe that our proposed *modules-and-messages* approach is one important step to simplify the combination of different modules. Furthermore, we believe that once modules can be re-used and combined freely, it will become easy to make fair and reliable comparisons of new protocol proposals.

References

[1] T. R. Andel and A. Yasinac. On the credibility of manet simulations. *Computer*, 39(7):48–54, 2006.

[2] D. M. Beazley. Simplified Wrapper and Interface Generator (SWIG).

[3] D. Cavin, Y. Sasson, and A. Schiper. On the accuracy of manet simulators. In *POMC '02: Proceedings of the second ACM international workshop on Principles of mobile computing*, pages 38–43, New York, NY, USA, 2002. ACM Press.

[4] G. W. Fischer, Z. Carmon, D. Ariely, G. Zauberman, and P. L'Ecuyer. Good parameters and implementations for combined multiple recursive random number generators. *Operations Research*, 47:159–164, 1999.

[5] J. Heidemann, N. Bulusu, J. Elson, C. Intanagonwiwat, K.-C. Lan, Y. Xu, W. Ye, D. Estrin, and R. Govindan. Effects of detail in wireless network simulation. In *Proceedings of Society for Computer Simulation (SCS) Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS'01)*, 2001.

[6] S. Kurkowski, T. Camp, and M. Colagrosso. MANET Simulation Studies: The Incredibles. *Mobile Computing and Communications Review*, 9(4):50–61, 2005.

[7] J. Liu, Y. Yuan, D. M. Nicol, R. S. Gray, C. C. Newport, D. Kotz, and L. F. Perrone. Simulation validation using direct execution of wireless ad-hoc routing protocols. In *Proceedings of the Workshop on Parallel and Distributed Simulation (PADS)*, pages 7–16, May 2004. Nominated for Best Paper award.

[8] M. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8(1):3–30, 1998.

[9] S. McCanne and S. Floyd. ns Network Simulator. `http://www.isi.edu/nsnam/ns`.

[10] S. Naicken, B. Livingston, A. Basu, S. Rodhetbhai, I. Wakeman, and D. Chalmers. The state of peer-to-peer simulators and simulations. *SIGCOMM Comput. Commun. Rev.*, 37(2):95–98, 2007.

[11] D. Reddy, G. F. Riley, B. Larish, and Y. Chen. Measuring and explaining differences in wireless simulation models. In *14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, 2006.

[12] M. Takai, J. Martin, and R. Bagrodia. Effects of Wireless Physical Layer Modeling in Mobile Ad Hoc Networks. In *Proceedings of MobiHoc'01*, Oct. 2001.

[13] A. Varga. The OMNeT++ discrete event simulation system. In *Proceedings of the European Simulation Multiconference (ESM'2001). June 6-9, 2001. Prague, Czech Republic*, 2001.