Network Simulation with ns-3

Niklas Kuse, Benedikt Jaeger*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: niklas.kuse@tum.de, jaeger@net.in.tum.de

***Abstract*—As computer networks become more complex, agile and flexible methods for research are required, allowing for a faster development of new network technologies. Network simulation provides this flexibility. This paper gives an overview on the network simulator ns-3. In addition to that, ns-3 is compared to OMNeT++ another well known network simulator and other measurement methods using real or emulated hardware. This paper references current projects which were realized using ns-3 to give an overview of the various application areas of ns-3.**

***Index Terms*—network simulation, ns-3, network emulation, comparison, testbed**

1. Introduction

Today's global network infrastructure is growing faster and faster. In order to keep up with the growing data demand, new and improved techniques to send data through those networks are required. That can be achieved through more and optimized research in this field. A popular approach is network simulation, providing a universal applicable and cost effective way. Cost-effective in terms of manpower, material and working time as development using network simulation brings benefits for all these aspects. In addition, research results using network simulation are deterministic and can be validated by others, by sharing the simulation setup. Another way is to build a network topology using real world hardware. This is not as flexible and cost effective as network simulation. This paper gives an overview on the network simulator ns-3 including its main structure. In addition to this, other in the research well known approaches are named and compared to ns-3. To help to decide if ns-3 is the best research tool for a specific question, this study will provide an overview on current research using ns-3 in their evaluation.

The structure of this paper is as follows: Section 2 provides related literature. Section 3 explains the main structure and design goals of ns-3. To get an idea of the overall performance of ns-3, Section 4 compares ns-3 to OMNeT++, real and emulated hardware and gives insights into the implementation of ns-3. Section 5 names work of different research areas all using ns-3. Finally, Section 6 summarizes the main statements of this study and states further work that can be done to support the conclusions drawn.

2. Related Work

ns-3 is not the only network simulator available. The book [1] by Wehrle et al. provides a good overview on contemporary simulators. The book contains code snippets explaining many use cases. Additionally, the authors provide insights into the implementation of the simulators ns-3, OMNeT++ and others. This allows a better understanding of the use cases the simulators are aiming at.

To get more information on the performance of the simulators ns-3 and OMNeT++, the author of this paper recommends taking a look into the original papers by Weingartner et al. [2], Rehman Khana et al. [3] and Khan et al. [4]. These papers do not only compare ns-3 and OMNeT++, but other simulators as well, allowing to get a better overview of all simulators available. All three papers determine ns-3 as the most performant simulator of the ones tested.

ns-3 can also be used in cooperation with other software. Gawłowicz et al. picture a good example on such cooperation in [5]. With their paper they contribute ns3-gym. A framework composed of ns-3 and OpenAI supporting the research for better networking protocols using reinforcement learning.

3. Structure and Functionality of ns-3

ns-3 is a network simulator targeting research and educational usage [6]. The simulator is implemented in C++ and provides bindings for Python, backing much of the C++ API [7]. Therefore, simulations for ns-3 have to be written in one of these languages as well. Allowing executed tests to be more easy to debug [8, section 2.1] and more realistic than simulations realized in a higher abstraction level like the predecessor ns-2 does. Simulations are more realistic, because the C++ code will be executed and no extra level of complex interpretation has to take place. By avoiding this extra level of abstraction, the implemented code of the simulation is closer to the real world implementation used later. Simulation is realized by processing a discrete list of all events about to occur, sorted ascending by their occurrence time [8].

ns-3 provides several model types allowing the user to specify all the different components of a network. To run the simulation, the user has to create all network parts needed for the simulation with respect to the following categories:

- Nodes, used for physical network systems e.g. smartphones or switches.

doi: 10.2313/NET-2020-11-1_13

- Devices, representing the actual part of a network node that is responsible for network communication like the ethernet card.
- Channels, illustrating cabels and other mediums used to transfer data in the real world.
- Protocols, describing the actions carried out on each network packet.
- Headers, organizing the data stored in packet headers as required by network standards.
- Packets, used to depict the data passed over the network with header information and payload.

Besides these components ns-3 provides more objects helping with the simulation by providing random numbers or storing additional required information [8]. As described above, the simulator works by processing a sorted list of events. To achieve this, the network topology, that is about to be simulated, has to be created using the network parts named above. Initial events have to be declared that will start the simulation and may result in the creation of further events. To schedule events the event occurrence time, an event handler and all its needed parameters are handed over to the simulator. Once the simulation is started, the simulator iterates over the list of sorted upcoming events. For each event, the simulation time is adjusted with the event time and afterwards the corresponding handler is called to process it. Events that get triggered by this handler during simulation, will be added into the simulators list of upcoming events and processed accordingly. As the simulator hops from event to event, the processing time is not in correlation with the real time. The simulation stops after a specified simulation time, or if the list of upcoming events in the topology runs out [8]. Because ns-3 is aimed towards research, it is important that test results can be gathered in any way required. To fit this constraint, ns-3 includes a tracing subsystem, allowing to measure and log any data. Data collection is supported in common data formats like *pcap* that can be analysed with packet analyzer software like Wireshark. For a more customized data collection, the trace sinks, parts of the tracing subsystem for data consumption, can be edited, to support any format of data output to meet the users need. Besides these analysis tools, ns-3 ships with software supporting the visualised debugging. The AnimationInterface can be incorporated in the siumlatiom to collect test result data for debugging. After finalizing the simulation, this data can be viewed as a graphic, showing the packet flow in the simulations topology [8]. All these features enhance ns-3 to be a good simulator to fit many research and educational needs.

4. Comparison with other Methods

Besides ns-3, other network simulators like OMNeT++ and even other approaches using emulated or real hardware exist. This section will evaluate the major differences to these concepts.

4.1. OMNeT++

OMNeT++ is a discrete event based simulator like ns-3. Just as ns-3, OMNeT++ is written in C++, but simulations are modelled using NED, a description language translated into C++ during simulation. OMNeT++ does not focus on network simulation only, allowing other event based simulations to be modelled and executed as well [2]. This is because OMNeT++ was originally not designed as a network simulator. Nevertheless, simulations are modelled using different approaches. Both tools can run similar simulations as shown by Weingartner et al. in [2]. Due to this, comparing the performance of both is crucial to decide which one to use for research. As shown in Figure 1, both tools perform simulations in linear computation time regarding the network size with ns-3 being slightly faster than OMNeT++. The second performance test executed by the authors of [2] investigated the memory usage. Their results, as shown in Figure 2, indicate a better memory usage of ns-3 compared to OMNeT++.

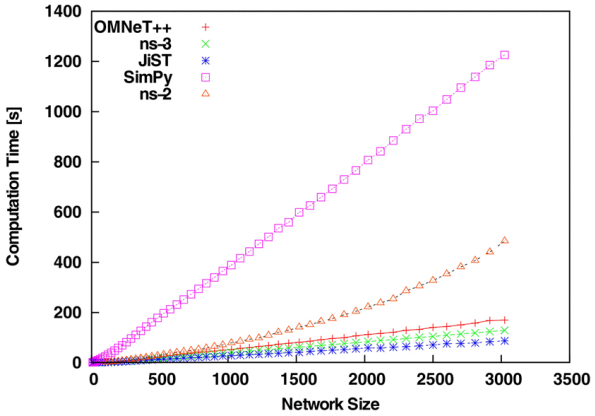

Figure 1: Simulation runtime vs. network size [2]

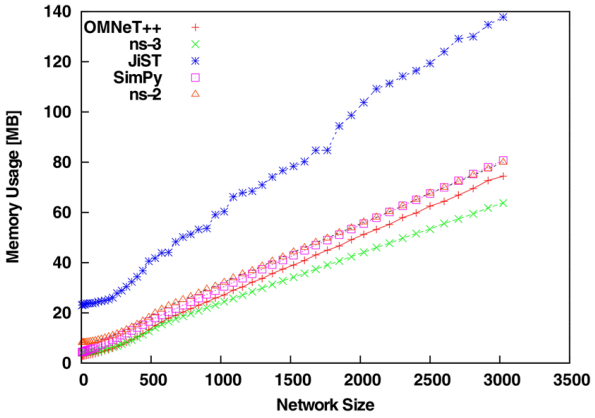

Figure 2: Memory usage vs. network size [2]

Comparing these results to the ones measured by Rehman Khana et al. in [3] and Khan et al. in [4] for ns-3 and OMNeT++ shows that ns-3 is able to maintain its advantage over OMNeT++ in different simulations. When evaluating these performance measurements today, it should be kept in mind that the three tests have been performed between 2009 and 2014. Today's versions of both simulators might behave different due to OS and software optimizations.

Another advantage of ns-3 is the implementation of packets. As packets in a network are carrying payload, this has to be possible in the simulation as well. For many

doi: 10.2313/NET-2020-11-1_13

simulation use cases the actual payload itself does not matter, but only its size. Therefore, dummy payload is only saved as a number representing the size used. As packets are subject to changes in a real world network, packets in ns-3 have to be as well. E.g. protocol headers are added or removed in different processing stages. Because ns-3 only passes pointers of packets around, changing the data for this pointer would as well result in changes at earlier stages of the packet, which are not intended. To avoid this, ns-3 enables a copy-on-write policy. This means, packets are duplicated, if a function writes to the memory at a pointer it has received from others [8]. Copy-on-write allows the programm to copy as little memory as possible. This again results in an optimized memory usage. In OMNeT++ packets are represented by C++ classes containing all carried information. In addition, each packet contains a pointer to its payload (the wrapped packet). Contrary to ns-3, OMNeT++ includes a copy-on-access policy to reduce the copy of unused packets [9, section 3.3.5].

4.2. Real Hardware

In addition to running network simulators, measurements can be performed using real hardware composed in a testbed. Using real world hardware allows the researcher to execute protocols and other software to be tested in their native environment. This results in more accurate test results, containing information a simulator could not determine [10] like random interference on wireless networks. Most simulators make assumptions (on the environment or based on random numbers) while simulating the network topology [11]. As network simulators are mainly constructed for research, they allow tests to be deterministic [11]. This constraint is highly required in scientific research allowing other researchers to verify the claimed statements. In contrast to this, testbeds using real hardware can not fit this constraint. As described by the author in [11] testbeds using wireless communication have to be shielded against external radiation. As this might be hard in larger scale networks, the author describes that their testbed runs tests during the night when less devices are emitting. In addition to that, their testbed allows to be reset to any point of the experiment, if unexplainable results are gathered afterwards. Adding such features allows the test results to be more reproducible, but not deterministic. Another drawback using real hardware is that test enviroments can not be scaled as easy as in network simulators. In order to scale networks, additional hardware has to be bought and added, resulting in network simulation to be more affordable [3]. These expenses require experiments to be planned more accurately in advance to minimalise experiment costs. Additionally, these expenses make it harder for other researchers to execute and verify the tests. Observing experiments in testbeds can be challenging as well, because test results can not be transferred using the same communication channel as the test data. Using the same channel, the test results would interfer with the test experiment resulting in non-determinism. This requires the data to be gathered and evaluated after the completion of the test or adding another communication channel [11]. Besides testing in testbeds, a researcher can perform active or passive measurements in a real world network e.g. the internet. Using active measurements, the topology of a network part is analysed by sending packets to other machines and determining the transfer rate or elapsed time. In contrast to this, passive measurement describes oberserving the traffic and packets, passing a specific node without sending packets [12]. As these measurements do not require to build a new network, it comes with advantages in costs. In addition, results are gathered on a real network making them representative. On the other hand, results can vary according to the workload in the network.

4.3. Emulated Hardware

Besides network simulation and testbeds using real hardware, there also is a method taking parts of both. Research approaches using emulated hardware would result in a simulator controlling the experiment, but sending the data over real, physical hardware [11]. Emulated hardware allows tests to be more affordable than using only real hardware. Nevertheless it has the same drawback with respect to non-determinism, because data is transferred using the same real world channels as approaches using only real hardware. ns-3 supports such a research method, as it can by used as the simulator sending its data over the hardware [8].

5. ns-3 in Research

As mentioned above, ns-3 is a network simulator aiming towards research. This section provides an overview of recent work using ns-3. All these approaches have some specific requirements that differ from the classic ones which most common networks do not have. As ns-3 is able to match all these requirements, this overview should help to decide, whether ns-3 is the correct tool for a research task or not. **Quantum key distribution.** ns-3 is used by Mehic et al. in their paper [13] providing a module for quantum key distribution. Quantum key distribution requires a specific topology containing two communication channels. One referred to as 'Quantum channel' is used for the transfer of quantum key material. The other one is used for the transmission of encrypted data and verification. As described by the authors, the quantum channel always has to be a point-to-point connection. Additionally the network has to meet some specific security requirements. Quantum key distribution on real hardware is limited to a transmission distance of about 100km using optical fibres. Building a testbed using this technology is expensive as well. As a result of this and the fact that ns-3 can deal with all requirements, follows that ns-3 is a good choice for such research tasks [13]. It should be mentioned that the module described by the authors is made available with their paper. **Electric vehicle.** According to Sanguesa et al. in [14], network simulation becomes more and more important for electric vehicle development. This is, because simulating scenarios for vehicular networks is too expensive to be done on real hardware. As cars become more and more

doi: 10.2313/NET-2020-11-1_13

interconnected, new and improved ways for this communication have to be found. This connection takes place to enable better traffic management or autonomous driving. ns-3 provides a perfect base for the development of such protocols. Electric cars are dependent on electric energy for all their operations. To provide a realistic simulation of these vehicles, the authors of [14] provide a module for energy consumption measurement in their paper. This module provides a helper class to create an energy consumption model for each node in the ns-3 simulation. This module takes into account all important information like weight or battery level during its evaluation. To proof the good accuracy of their module, the results are compared to a similar simulation using the traffic simulator SUMO. For taking into account that the nodes can move around dynamically as it might be required, the next paper FlyNetSim provides a good approach.

FlyNetSim. ns-3 in not only a stand-alone network simulator, but rather capable to incorporate with other software, as shown in [15]. The authors use ns-3 in cooperation with Ardupilot, an open source software for unmanned vehicles navigation. With their paper they provide FlyNetSim. A simulator capable of simulating multiple unmanned aerial vehicles. ns-3 enables their tool to provide simulations for different communication ways between the vehicles, using LTE or device-to-device communication, taking into account that the vehicles are moving and distances between nodes are changing. The authors choosed ns-3 because of its interfaces for external systems. A challenge the authors had to deal with is the different time management used by ns-3 and Ardupilot. As mentioned, ns-3 is a discrete event simulator. As Ardupilot performs operations in real time, incorporating both software is challenging. However, binding ns-3 to a real-time scheduler allowed both ns-3 and Ardupilot to incorporate with only a lack of accuracy in situations with many events in a short amount of time [15].

ns3-gym. ns-3 can not only work in cooperation with other software, as described above, but rather simulations can be controlled by external software, allowing for even more complex cooperations. The authors of [5] have created a framework that combines ns-3 and OpenAI to support research for optimized networking algorithms using machine learning based on reinforcement learning (RL). Therefore, they incorporate ns-3 and OpenAI using a custom middleware. This middleware is used, to enable OpenAI to control specific nodes in the simulation. As RL works by breaking the situation into steps executed on after another, the middleware allows to start and stop the simulation after such step in order to report the state back to OpenAI. This allows OpenAI, to evaluate and improve the learned after each step. In addition, ns3-gym allows to evaluate knowledge gained from simulations to be executed on real hardware afterwards, to improve know-how in real world situations.

6. Conclusion and Further Work

A main goal of network simulation is to provide a platform to develop and test new communication technologies. ns-3 fits this requirement by providing a framework that can be extended by anybody allowing it to become more and more robust. Another advantage of ns-3 is the simplicity to scale topologies and execute deterministic simulations. Both are constraints to develop new robust and reliable technologies. Approaches using emulated or real hardware are not able to match. On top of that, ns-3 uses good approaches to reduce mistakes done by the developer during the simulation development. This avoids the occurrence of memory leaks during simulation execution and results in an optimized memory usage. Comparing ns-3 to OMNeT++ shows that it is more performant than the latter, even though both are implemented using C++. As shown in Section 5 modules for many research fields are available for ns-3 simplifying its usage. In the authors opinion, this enables ns-3 to be considered in many research evaluations using any kind of network. Thus, many research fields can profit from the benefits ns-3 provides.

As mentioned in Section 4.1 the cited performance evaluations of OMNeT++ and ns-3 were all performed between 2009 and 2014. Due to the possible performance improvements in OS and simulator, these results might not be representative anymore. To get insights into their today's performance, some additional test would be required. To aquire such information, performance test using today's OS and simulator versions and the ones of the cited paper could be performed. Running the same simulation structure in the old and current version would allow to determine the performance improvements done in the last years, resulting in a better understanding of today's accuracy of the results drawn from the cited papers. Back then, the cited papers were produced five years apart and the performance comparison was quite similar. This leads us to the conclusion that the performance comparison would not differ significantly today.

References

[1] K. Wehrle, M. Güneş, and J. Gross, *Modeling and tools for network simulation*. Berlin, Heidelberg: Springer, 2010.

[2] E. Weingartner, H. vom Lehn, and K. Wehrle, "A performance comparison of recent network simulators," in *2009 IEEE International Conference on Communications*, June 2009, pp. 1–5.

[3] A. u. Rehman Khana, S. M. Bilalb, and M. Othmana, "A performance comparison of network simulators for wireless networks," 2013.

[4] M. A. Khan, H. Hasbullah, and B. Nazir, "Recent open source wireless sensor network supporting simulators: A performance comparison," in *2014 International Conference on Computer, Communications, and Control Technology (I4CT)*, Sep. 2014, pp. 324–328.

[5] P. Gawłowicz and A. Zubow, "Ns-3 meets openai gym: The playground for machine learning in networking research," in *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ser. MSWIM '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 113–120. [Online]. Available: https://doi.org/10.1145/3345768.3355908

[6] "Ns-3 homepage," https://www.nsnam.org/, [Online; accessed 06-June-2020].

[7] "Using python to run ns-3," https://www.nsnam.org/docs/manual/html/python.html, [Online; accessed 14-June-2020].

[8] G. F. Riley and T. R. Henderson, *The ns-3 Network Simulator*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 15–34. [Online]. Available: https://doi.org/10.1007/978-3-642-12331-3_2

[9] A. Varga, *OMNeT++*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 35–59. [Online]. Available: https://doi.org/10.1007/978-3-642-12331-3_3

[10] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Ya Xu, and Haobo Yu, "Advances in network simulation," *Computer*, vol. 33, no. 5, pp. 59–67, 2000.

[11] B. Blywis, M. Guenes, F. Juraschek, and J. H. Schiller, "Trends, advances, and challenges in testbed-based wireless mesh network research," *Mobile Networks and Applications*, vol. 15, no. 3, pp. 315–329, 2010. [Online]. Available: https://doi.org/10.1007/s11036-010-0227-9

[12] V. Mohan, Y. R. Janardhan Reddy, and K. Kalpana, "Active and passive network measurements: A survey," *International Journal of Computer Science and Information Technology*, vol. 2, pp. 1372–1385, 2011. [Online]. Available: http://ijcsit.com/docs/Volume%202/vol2issue4/ijcsit2011020402.pdf

[13] M. Mehic, O. Maurhart, S. Rass, and M. Voznak, "Implementation of quantum key distribution network simulation module in the network simulator ns-3," *Quantum Information Processing*, vol. 16, no. 10, p. 253, 2017. [Online]. Available: https://doi.org/10.1007/s11128-017-1702-z

[14] J. A. Sanguesa, S. Salvatella, F. J. Martinez, J. M. Marquez-Barja, and M. P. Ricardo, "Enhancing the ns-3 simulator by introducing electric vehicles features," in *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, July 2019, pp. 1–7.

[15] S. Baidya, Z. Shaikh, and M. Levorato, "Flynetsim: An open source synchronized uav network simulator based on ns-3 and ardupilot," in *Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ser. MSWIM '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 37–45. [Online]. Available: https://doi.org/10.1145/3242102.3242118