Network Simulation with OMNet++

Simon Bachmeier, Benedikt Jaeger*, Kilian Holzinger*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: Simon.Bachmeier@in.tum.de, jaeger@net.in.tum.de, holzinger@net.in.tum.de

***Abstract*—**
In view of the current development, computer networks are increasing in size and complexity. For development reasons simulations became more important in this area. OMNeT++ is a discrete event simulation environment. It was available and present for over 20 years and designed not as a network simulator, but for all-purpose use. This results in its area of application being not only in computer science fields. Over the years OMNeT++ has built a revenue for being able to simulate all kinds of areas. From queuing to wireless networks simulation, everything is possible. In this paper the OMNeT++ framework is presented with a detailed view into network simulation. Also, OMNeT++ is compared to other popular network simulators, where its strengths and weaknesses are discussed.

***Index Terms*—OMNet++, Network Simulation, NS-3, INET, MiXiM**

1. Introduction

The Internet backbone changed from a connection between research communitys to commercial use. This means already after eight and a half years of its lifetime, it developed from six nodes with 56 kbps links to 21 nodes with multiple 45 Mbps links. During the early stages, the Internet already grew to over 50,000 networks, including outer space and all continents [1]. 2016 the LTE networks nearly had 1.5 billion users with peak data rates of 450 Mbps [2]. The number of deployed devices using the Internet by 2020 has been estimated to be as high as 20.8 billion [3]. These numbers show that it is crucial that networks are functioning without errors. For development reasons testing is needed when working with networks. Testing with real networks can be very expensive and does not scale, therefore network simulations became very popular. Much hardware is needed for an experiment. The size of the current networks makes it hard to build a real simulation. A simulation breaks down the functionality of a network and is able to build large networks. A simulator capable of such is OMNeT++. OMNeT++ stands for Objective Modular Network Test bed in C++ [4]. It was introduced in 1997 [5] and is a C++ based discrete event simulator. OMNeT++ software is free for academic and non-profit use. Its commercial version OMNEST can be obtained on the official website [6]. As Vargas states, OMNeT++ can be seen as a gap filler [5]. It positions itself between open-source, research orientated software and expensive commercial alternatives. To get a better understanding of OMNeT++, Section 2 gives an overview. OMNeT++'s approach to simulation is quite different from other simulators like NS-3 [7] or JiST [8]. OMNeT++ does not provide simulation components directly, but gives a framework, which can be seen as a basic building block. It also differs in its modularity, components can be reused in all kinds of simulations. In Chapter 3 there are examples and an explanation of extensions for OMNeT++, so called frameworks, related to network simulations. These frameworks, e.g. INET [9] or MiXim [10] are developed independently and follow their own update cycles. Over the years a many simulation models, have been developed in a broad field, e.g. IPv6-, peer-to-peer-, storage area-, and optical networks. Also companies like IBM, Intel and Cisco are using OMNeT++ successfully in commercial projects or for in-house research [5]. In this paper the author gives an overview of OMNeT++, in order to understand why it was able to remain and be popular over all those years since existing. Therefore, it is important to understand how it works and what makes it different from its competitors.
The papers's structure is divided into to the following sections: Section 2 gives an overview of OMNeT++, to provide a basic understanding. How network simulation is done, is shown in Section 3. Finally, Section 4, compares OMNeT++ to other simulators in its class.

2. Structure of OMNeT++

OMNeT++ is a multipurpose network simulation framework. It provides a building kit for many kinds of simulations. At its core are reusable modules, which is the reason why OMNeT++ is considered a component architecture. The structure can be compared to LEGO blocks [11], which is a good illustration. When there are well implemented modules, they can be reused in various ways in all kinds of simulations, just like LEGO blocks. The following sections describe the functionality of OMNeT++'s internal and external structure.

2.1. Modeling and Modules

OMNeT++'s architecture is built of modules communicating by messages. There are two types of modules, the active modules, referred to as simple modules and compound modules. Simple modules can be grouped into compound modules, the second module type. Compound modules group other modules an can be therefore seen as passive only representing the internal functionalities. Hierarchy levels are not limited, that is why OMNeT++'s

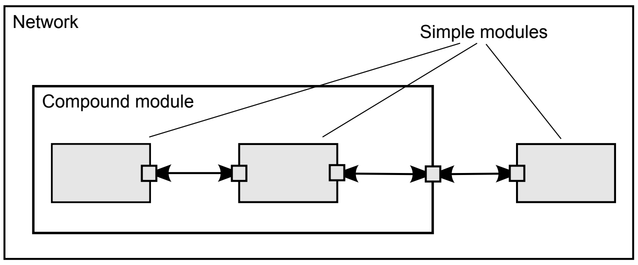


Figure 1: Simple and compound modules. Arrows represent connections between modules via gates [11].

structure is considered hierachical [11]. The entire simulation model represented by a network, therefore is a compound module. Simple modules are written in C++. The communication of modules is done by messages. In order to send messages, simple modules have input and output gates for sending and receiving. It is also possible to send messages directly to their destination, but normally it is done using gates. In a simulation, messages can represent customers in queues or packages in a network. Input and output gates can be linked by so-called connections. Compound modules also have such gates and connections to communicate with external modules. Connections can only be established in a single level of module hierarchy i.e sub modules can have connections to the compound module gate or other sub modules inside a compound module , but not to the outside. That is needed to maintain the reusability of modules. These connections can be seen as an own entity with parameters like bit error- rate or delay. Connections with predefined parameters, so called termed channels, can also be reused [11].

2.2. Design of a Model

An OMNeT++ model has a starting module, the system module. From the system module at the top level there are nested modules with a hierarchical structure with no boundaries to the bottom. With no limits in depth the user has the opportunity to build every structure he wants. The model structure is written in OMNeT++ network description language, short NED. The simple modules at the bottom are programmed in C++ using the simulation class library provided by OMNeT++. Listing 1 shows a C++ code example by TU-Ilmenau [12].

```
#include "omnetpp.h"
class MyModule : public cSimpleModule
{
    //a macro that calls constructor
   and sets
    //up inheritance relationships:
    Module_Class_Members (MyModule,
   cSimpleModule, 0)
    //user-specified functionality
   follows:...
};
Define_Module (MyModule); //announce
   this class as a module to OMNeT
```

Listing 1: Example of a simple module's C++ code. The class MyModule describes the functionality of a simple module. User can specify how to react on messages.

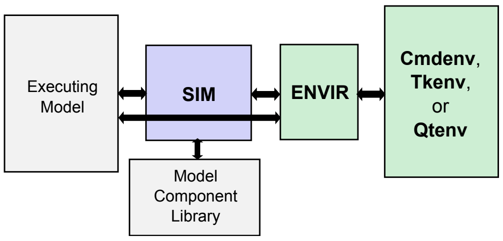


Figure 2: Logical architecture of an OMNeT++ simulation programm [11].

2.3. Architecture

The previous sections gave a short overview of OMNeT's internal structure. Figure 2 gives a understanding of the high level and logical architecture. ENVIR, CMDENV and TKENV describe the user interface libraries. This is the environment where the simulation is executed. It also describes the origin of input data and where simulations result go to. ENVIR is a library, which presents itself as a connection, containing all common user interface code. CMDENV and TKENV are ENVIR based libraries that have a specific user interface implementation. The all-purpose interface is CMDENV, known as the command line user interface. It is a small portable and fast interface that works on all platforms, which provides a console e.g. linux and is mainly used for batch execution. TKENV is a graphical user interface, for debugging and visualization of simulations. This environment describes the simulation's execution, from debugging to visualizing. It is possible to embed a OmNeT++ simulation into a larger application by replacing the user interface. The component library is built by the code of compiled simple and compound modules [5]. At the beginning of the simulation execution phase, the simimulation kernel builds the simulation model, which is the model created for the simulation. [13]

3. Network Simulation with OMNeT++

As stated in the previous sections OMNeT++ itself is merely a framework. That means in the base version only supports routing messages [5]. This feature traversions of a network and building a graph data structures. In addition, a user could further use graph algorithms to traverse it. But with nothing more provided the user has to implement a whole network with protocols and network associated structures himself. With the design of OMNeT++, however, a simulation can be easily extended by a network framework. Network simulation with OMNeT++ is simplified by extending it with existing frameworks such as INET [9] elaborated in section 3.1 or, specialized on mobile and wireless networks, the MiXiM framework, elaborated in section 3.2 [10]. Importing a framework is similar to importing libraries in Java [11]. Framework files are added to the project and can be accessed via the import [11].

3.1. INET

INET is a model library, which is specialized in designing new protocols [14]. With INET the user can build

hosts, routers and other network devices by using reusable components. INET provides components that are easy to modify and understand so that users can build their own components [14]. Some of INET's features are: IPv4/IPv6 network stacks, pluggable protocol implementation for various layers, wired/wireless interfaces and many more. In the following a project by Mohammed Alasmar and George Parisis will be presented which uses INET [15]. It uses the OMNeT++ with the INET framework to evaluate modern data center transport protocols.
Data center networks (DCNs) are getting more important, because they are used for the infrastructure of search services such as Google, social networks, cloud services and video streaming (e.g. Disney+). Alasmar et al. state that OMNeT++ in addition to INET is perfect for simulating and evaluating data transport protocols in DCNs [15]. The reason lies within INET's flexibility. New protocols can be easily added to the project and network devices (e.g. switches, routers etc.) are already provided. For example, there are simple module implementations for TCP and UDP, which users can use and extend. Within this project, a simulation model for the neighbor discovery protocol (NDP) and a FatTree network topology generator was built. Also, a data center environment that can be used to evaluate and compare data transport protocols, such as NDP and TCP, was developed. This outlines the possibilities with OMNeT++, not only creating simulations, but also creating a framework for other evaluations. In the simulations specialized on NDP protocols they used OMNeT++'s save tool command line to process and compare the results [15].

3.2. MiXiM

Another example for a network simulation framework which runs on OMNeT++ is MiXiM (mixed simulator). This framework is specialized for mobile and wireless networks. It offers detailed models of radio wave propagation, interference estimation, radio transceiver power consumption and wireless MAC protocols [10].
Following there is a short example of a simulation using the MiXiM framework, in order to improve the Quality of Service (QoS) in a wireless sensor network (WSN) [16]. This paper was written by Kodali et al [16]. The reason behind it was, that in a real time environment WSN are often unreliable and inaccessible, which has a bad effect on the QoS. In the paper the authors discuss adjustments of power, range and bit rates to attain adaptive topology control (ATC), in order to maintain the QoS of a WSN, which have a broad spectrum of usage. For example, it is used by the military or for weather monitoring, with the core concept of sensing, computing and communication. MiXiM was chosen because it supports the mobility framework which is used for the WSN network. The performance of Carrier Sense Multiple Access (CSMA) was measured in packet end-to-end delivery ratio and throughput. As Kodali et al state in their conclusion, they were able to improve the network performance by 29%, by varying the range and power adaption [16].

3.3. Recent Use of OMNeT++

This section provides an overview of projects featuring OMNeT++.

- UAV (Unmanned aerial vehicle) Swarm Network: The paper, deals with a swarm intelligence-based damage-resilient mechanism for UAV swarm networks. The authors build a simulation in OMNeT++, which presents a mechanism to recover a unified UAV swarm network after suffering damage [17].
- Secure Data Transmission and Sinkhole Detection: OMNeT++ is used for evaluating better methods for improved security in a multi-clustering wireless sensor network by homomorphic encryption and watermarking [18].
- Novel Segment Based Safety Message Broadcasting in Cluster-Based Vehicular Sensor Network: With multiple vehicles sending messages in vehicular sensor networks, data collisions occur more often, leading to corrupted packages. To counteract, OMNeT++ ist used for validation [19].

4. Comparison to other Network Simulators

Over the last years computer hardware has significantly improved. Also, networks as a topic got more attention, with the world being more connected than ever. The complexity of systems and networks topology is increasing drastically. To implement new network structure or protocols, it is indispensable to test, before actually deploying them. For example, when an untested defective new structure is pushed into an established network, fixing it will be costly and effort to fix that, while the system is still running. There are two kinds of test environments. A real system with existing hardware or a simulation. As already mentioned, the increased complexity makes it very expensive to build real systems for testing. Therefore, simulations have gained a lot of attention over the past years. Two network simulators which gained a lot of popularity during the last years are OMNeT++ and NS-3 [20]. In the following sections a comparison between these two simulators takes place. Also taking OMNeT++ design decisions, described in the previous sections, into account described by A. Vargas and R. Hornig [5] and comparing them to other simulators. After the comparison to NS-3, the paper presents a short comparison to other popular simulators.

4.1. Comparison to NS-3

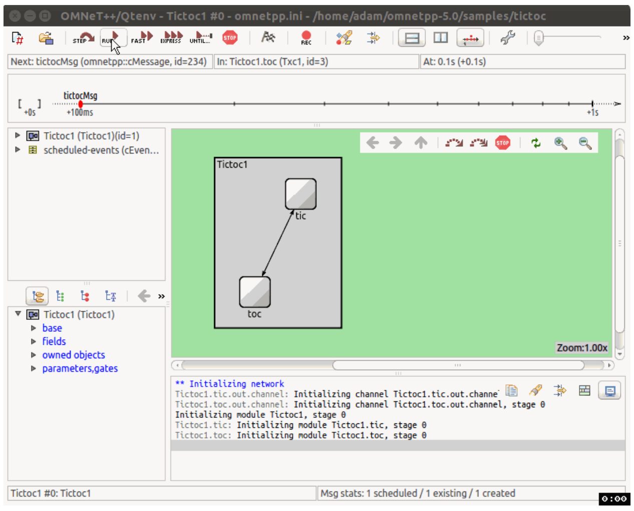
NS-3 (network simulator version 3) is the successor of the of NS-2, which is not supported anymore nor has backwards-compatibility. It is an open source discrete event network simulator [21], that tries to maintain an open environment for researchers to share their projects. The simulator itself is a C++ library and an integrated development environment is needed. During a simulation the network data traffic can be collected in a pcap-file. Then it is possible to analyze the file via wireshark or similar software. Wenhan Yao compared these two simulators in his master thesis [21]. Other sources are the general comparisons to other simulators in various scenarios [22]–[24]

4.1.1. Structural Differences of the Simulators. In OMNeT++, a NED file is used to describe the structure of a model, while the behavior of modules is written in C++. For simulations, the user has to choose a user interface (e.g. CMDENV,TKENV) see Figure 2 and define the starting parameter in an `omnetpp.ini` file. Then the result of a simulation is written into vector and scalar files, which can be evaluated by external software. For a simulation in NS-3, the user starts with a script written in C++ or Python. The script defines the topology of the network and how the simulations is run. A network is described by its nodes, connections and devices and their usage [21]. The models, which can be used for simulation, can be found in the NS-3 library. All the parameters of simulation are also written into the script, for example data rate, IP address and positioning of nodes. The resulting data can go to a pcap file, which can also be analyzed.
Both simulators are using C++ to describe their modules. NS-3's connection to modules is more direct than OMNeT++'s. In NS-3, the script can directly access the modules from the library, while in OMNeT++ the NED files are interposed. In other words, in OMNeT++ you need NED files to create and access modules, which makes the simulation more complex. OMNeT++ has visualization options already included. This simplifies network simulation decisions for less experienced users, compared to NS-3 with no included visualization. It is possible, though, to visualize the output files from NS-3, in order to have a visualization of graphs. This works similar to OMNeT++, where the output files can be processed by various programms [21].

4.1.2. Differentiation between OMNeT++ and NS-3. The GUI is one of OMNeT++'s main advantage, compared to NS-3. It can picture signals e.g TCP packages and components of a network structure. An example of a GUI screen can be seen in Figure 3. In simulations OMNeT++ always uses more computer resources than NS-3 [21], [25]. Also, OMNeT++ takes longer to execute. Another core difference is that NS-3 was built for network simulation, while OMNeT++ is multipurpose. Therefore, it is less workload for the user to create a simulation in NS-3, because the whole structure can be written in one script rather than having to implement modules, NED files and specify user interfaces. Both simulators deliver good performances, with similar workloads on the testing system and messages throughput. An outstanding feature is OMNeT++ visualization. The integrated GUI and the possibility to visualize the simulation, debugging process or even the structure of a simulation are one of the outstanding features of OMNeT++ [21], [22], [24]. But it is important to mention that both simulators are giving good simulation results for similar simulations in execution time and workload of the system [20]–[22], [24].

4.2. OMNeT++ and Other Simulators

This section provides a summary of the comparison from OMNeT++, NS-3, NS-2, SimPy and JiST, done by Weingartner et al [20]. SimPy incorporates a different


OMNeT++/Qtenv - Tictoc1 #0 - omnetpp.ini - /home/adam/omnetpp-5.0/samples/tictoc

Next: tictocMsg (omnetpp::cMessage, id=234) | In: Tictoc1.toc (Txc1, id=3) | At: 0.1s (+0.1s)

```
** Initializing network
Tictoc1.tic.out.channel: Initializing channel Tictoc1.tic.out.channe
Tictoc1.toc.out.channel: Initializing channel Tictoc1.toc.out.channel, stage 0
Initializing module Tictoc1, stage 0
Tictoc1.tic: Initializing module Tictoc1.tic, stage 0
Tictoc1.toc: Initializing module Tictoc1.toc, stage 0
```

Tictoc1 #0: Tictoc1 | Msg stats: 1 scheduled / 1 existing / 1 created

Figure 3: Example of a GUI in an OMNeT++ Simulation [26].

simulation approach as a process-oriented discrete-event simulator and is written in Python [20], [23]. Another different simulation approach is presented by JiST. It is a high performance Java based simulation environment and just a simulation kernel [5], [20]. The authors built equal simulation scenarios for all the simulators, to compare them to each other. Their result was that only SimPy had slightly higher loss rates, still acceptable results thou. Afterwards they went to performance testing of the different simulators under equal conditions. Overall, the conclusion was that NS-3 has the best performance, while JiST advantages are in simulation run time. But OMNeT++ is the only simulator providing a GUI. JiST and NS-3 simulations are source code developed. The authors wrote that all of them are equally suited for network simulations, but it depends on the situation which one to choose [20], [23], [24].

5. Conclusion

Varga and Hornig describe in their paper the core elements of OMNeT++'s design, which is also crucial for large scale simulation [5]. The hierarchical structure with its reusable components, takes a huge part in its success. It means less work for the user by just reusing finished modules as much as possible. Another key point which differentiates OMNeT++ to many other simulators is its GUI and the possibility to visualize the simulation, debugging and the structure. Debugging consumes a lot of time in projects. Visualization often helps and reduces time consumption. Simulations are modular customizable allowing a user to embed an emulation into a larger application. This integration opens up many areas of application. As other simulators, OMNeT++ can produce output files to have a better analysis with other programs. However, the structure of OMNeT++ with its layers makes small simulations often more complex than they need to be. For example, when a user just wants to create a test simulation, where two nodes send each other a package, in OMNeT++ you need to define the kernel and structure of al parts from the system . The NED language was designed to scale well. With increasing complexity in networks, however, it reaches its limits. Therefore, improvements to the software itself had to be made, where the user has to adjust too. Furthermore, the environment is clearly divided in parts like NED modules which requires more computer resources for network simulations. Overall, however, OMNeT++ follows a clear structure, which makes it easy to handle, also within large simulations.

References

[1] B. Leiner, V. Cerf, D. Clark, R. Kahn, L. Kleinrock, D. Lynch, J. Postel, L. Roberts, and S. Wolff, "A brief history of the internet," *Computer Communication Review*, vol. 39, pp. 22–31, 10 2009.

[2] D. López-Pérez, D. Laselva, E. Wallmeier, P. Purovesi, P. Lundén, E. Virtej, P. Lechowicz, E. Malkamaki, and M. Ding, "Long term evolution-wireless local area network aggregation flow control," *IEEE Access*, vol. 4, pp. 9860–9869, 2016.

[3] Z. Yan, H. Li, S. Zeadally, Y. Zeng, and G. Geng, "Is dns ready for ubiquitous internet of things?" *IEEE Access*, vol. 7, pp. 28 835–28 846, 2019.

[4] "OMNet++ Homepage," http://www.omnetpp.org, [Online; accessed 8-May-2020].

[5] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," 01 2008, p. 60.

[6] "OMNEST Homepage," https://omnest.com/, [Online; accessed 24-May-2020].

[7] "NS-3 Homepage," https://www.nsnam.org/, [Online; accessed 24-May-2020].

[8] "JiST Homepage," http://jist.ece.cornell.edu/, [Online; accessed 24-May-2020].

[9] "INET Framework," https://inet.omnetpp.org/, [Online; accessed 11-May-2020].

[10] "MiXiM Framework," http://mixim.sourceforge.net/, [Online; accessed 11-May-2020].

[11] "OMNet++ Simulation Manual," https://doc.omnetpp.org/omnetpp/manual/, [Online; accessed 9-May-2020].

[12] "C++ code snippet," https://www.tu-ilmenau.de/fileadmin/media/telematik/lehre/Projektseminar_Simulation_Internet_Protokollfunktionen/Material/03_OmNet__.pdf, [Online; accessed 27-May-2020].

[13] [Online; accessed 15-August-2020]. [Online]. Available: https://ewh.ieee.org/soc/es/Nov1999/18/userif.htm

[14] "INET introduction," https://inet.omnetpp.org/Introduction, [Online; accessed 12-May-2020].

[15] M. Alasmar and G. Parisis, "Evaluating modern data centre transport protocols in omnet++/inet," in *Proceedings of 6th International OMNeT++ Community Summit 2019*, ser. EPiC Series in Computing, M. Zongo, A. Virdis, V. Vesely, Z. Vatandas, A. Udugama, K. Kuladinithi, M. Kirsche, and A. F\"orster, Eds., vol. 66. EasyChair, 2019, pp. 1–10. [Online]. Available: https://easychair.org/publications/paper/4xwP

[16] R. Kodali and M. Vijay Kumar, "Mixim framework simulation of wsn with qos," 05 2016, pp. 128–131.

[17] M. Chen, H. Wang, C. Chang, and X. Wei, "Sidr: A swarm intelligence-based damage-resilient mechanism for uav swarm networks," *IEEE Access*, vol. 8, pp. 77 089–77 105, 2020.

[18] H. A. Babaeer and S. A. AL-ahmadi, "Efficient and secure data transmission and sinkhole detection in a multi-clustering wireless sensor network based on homomorphic encryption and watermarking," *IEEE Access*, pp. 1–1, 2020.

[19] I. S. Alkhalifa and A. S. Almogren, "Nssc: Novel segment based safety message broadcasting in cluster-based vehicular sensor network," *IEEE Access*, vol. 8, pp. 34 299–34 312, 2020.

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

[21] W. Yao, "Analyse und vergleich der netzsimulatorenns-3 und omnet++," p. 74, 05 2018, [Online; accessed 21-May-2020]. [Online]. Available: http://midas1.e-technik.tu-ilmenau.de/~webkn/Abschlussarbeiten/Masterarbeiten/ma_yao_sw.pdf

[22] Xiaodong Xian, Weiren Shi, and He Huang, "Comparison of omnet++ and other simulator for wsn simulation," in *2008 3rd IEEE Conference on Industrial Electronics and Applications*, 2008, pp. 1439–1443.

[23] V. Oujezsky and T. Horvath, "Case study and comparison of simpy 3 and omnet++ simulation," in *2016 39th International Conference on Telecommunications and Signal Processing (TSP)*, 2016, pp. 15–19.

[24] A. Zarrad and I. Alsmadi, "Evaluating network test scenarios for network simulators systems," *International Journal of Distributed Sensor Networks*, vol. 13, no. 10, p. 1550147717738216, 2017. [Online]. Available: https://doi.org/10.1177/1550147717738216

[25] S. B. A. Khana and M. Othmana, "A performance comparison of network simulatorsfor wireless networks," pp. 1–6.

[26] [Online; accessed 11-June-2020]. [Online]. Available: https://docs.omnetpp.org/tutorials/tictoc/part2/