Simulation Environment for Ad-Hoc Networks in OMNeT++

S. Imre, Cs. Keszei, D. Hollós, P. Barta,Cs. Kujbus

Imre@hit.bme.hu, kecsa@rlan2.hit.bme.hu
Department of Telecommunications Budapest University of Technology and Economics
Pazmany P. 1/D, Budapest, H-1117

OMNeT++ is a powerful object-oriented modular discrete event simulator tool. Our team is working on a simulation environment for ad-hoc networks based on the resources provided by OMNeT++. In this paper we will introduce our environment and our first ad-hoc scenario inspected by this new tool.

I. Introduction

Today's and tomorrow's popular and in this way widely spread wireless LAN solutions are mainly infrastructure based. However, the IEEE 802.11b standard has ad-hoc option and the ad-hoc home extension in HiperLAN/2 proves that there is a need for ad-hoc networking [H2]. We know, that the ad-hoc solutions in the market are restricted to small networks for the time being, where the whole network is not bigger than the coverage area of one mobile node's radio range (802.11b ad-hoc mode), but experts and engineers are thinking of larger networks in the near future. Keeping in mind the new challenges [GUP00] [CHA01] designing of large ad-hoc networks requires suitable simulation environment which is able to investigate different medium access, routing, mobility prediction, etc. algorithms. Therefore the Mobile Communications Laboratory at Budapest University of Technology and Economics launched a project in 2000 to build simulation environment for ad-hoc networks. Our solution is based on OMNeT++ discrete event simulator which was developed by several European university teams *(*BUTE – Hungary, TU Delft – The Netherlands, Monash University – Australia, University of Karlsruhe – Germany, TU Berlin – Germany, etc.) [OMN01].
This paper is organised as follows: In Section II. we give a short survey of OMNeT++ introducing its main features. Section III. describes our ad-hoc network model and the implementation of the model in OMNeT++. Finally we summarise the current status of the simulation environment. We have implemented an ad-hoc multihop demo simulation in our environment and present some measurement results just to show it in work. At the end our future plans are presented.

II. OMNET++

OMNeT++ is an object-oriented modular discrete event simulator. The name itself stands for Objective Modular Network Testbed in C++.
Although this program can be used to simulate everything what is based on discrete events (communication protocols, computer networks, multi-processor and distributed systems, games, etc.), it is very easy to learn and use it.
The main advantages of OMNeT++ are:

- **You do not have to learn new programming languages**, so you can write all your code in C++, in spite of that:
- you are offered to use a **graphical user interface**,
- and the whole **simulation is portable between Unix-based systems, Windows (9x and NTs) and DOS**.

- Many different structures can be simulated without modifying the source code and rebuilding the system, using parameters.
- You do not have to write the C++ code to build the modules used in the simulation (objects, gates): The very easy NED (Network Description Language) compiler does it.
- The **GUI offers many possibilities to follow and debug the simulation** flow.
- You are offered to use **parameter-watching functions and plot diagrams** of them.
- You can use many pre-defined classes (**topology support** which contains graph algorithms, **finite deterministic state machine support**, etc.), and many other implemented modules (**TCP/IP, routing models**) under OMNeT++, their number grows dynamically.
- OMNeT++ has clear, well-defined, documented and hierarchically nested modules, which are available in source code, so you can debug and/or extend them.
- You can run the simulation on many machines at the same time using the PVM support.
- Only **two days are enough to learn the most important things** and implement your first simulation

The basic workflow using the OMNeT++ is the following:

1) Give the network structure to the framework (e.g., using the NED language, may make the connections parameter-based),
2) implement the modules which stand at the lowest hierarchy level (in C++),
3) compile and link them,
4) afterwards let the framework call your modules and handle all the message queues, etc.
5) In the end you can evaluate results generated by the built-in parameter-watching functions, and the figures graphically produced by GNU plot.

Since the connections and module types can be parameter-dependent (e.g. this means, you can implement a terminal module more than once and there is a condition of some parameters which one shall be executed and how they shall be connected to each other) you can rerun the simulation and evaluate results using different “versions” of modules and connections. You are also able to make it in a loop if you would test a large amount of alternatives.
To learn the basics takes about two days, which is very short compared to other systems. Everything is explained in the manual and there are also several sample simulations in the package.

III. Simulation of ad-hoc networks in OMNET++

The main goal of our simulation system is to build a sophisticated ad-hoc network model and suitable testing environment, which provides answers for ad-hoc related routing, topology, security, mobility, etc. questions.
The ad-hoc simulation environment consists of several modules (see Fig. 1.).

- **Mobile client module:** this entity represents the user terminals in the system. Every terminal is characterised by means of several physical and higher layer parameters, such as transmission power, implemented protocols (routing, security), etc. The mobile client module consists of the following submodules: The traffic generator emits packets according to the type of the client (e.g., speech, FTP, etc.) The physical block has a sender and a receiver entity with queues where the received and generated packets are stored. Finally the communication submodule implements the rules of the defined protocol stack (i.e., generates signalling messages). Of course not all the details of a given layer are implemented bit by bit only those which are required by a given investigation. However this fact implies that the stack becomes more and more sophisticated during the evolution of the simulation environment. On the other hand large amount of effort was (is) made to implement different protocols by other teams interested in OMNeT++ which can be applied in our simulator too (E.g. TCP/IP and HiperLAN/2 are near to the finalisation). Every submodule has one or more input and output ports, which ensure the connections between the submodules.

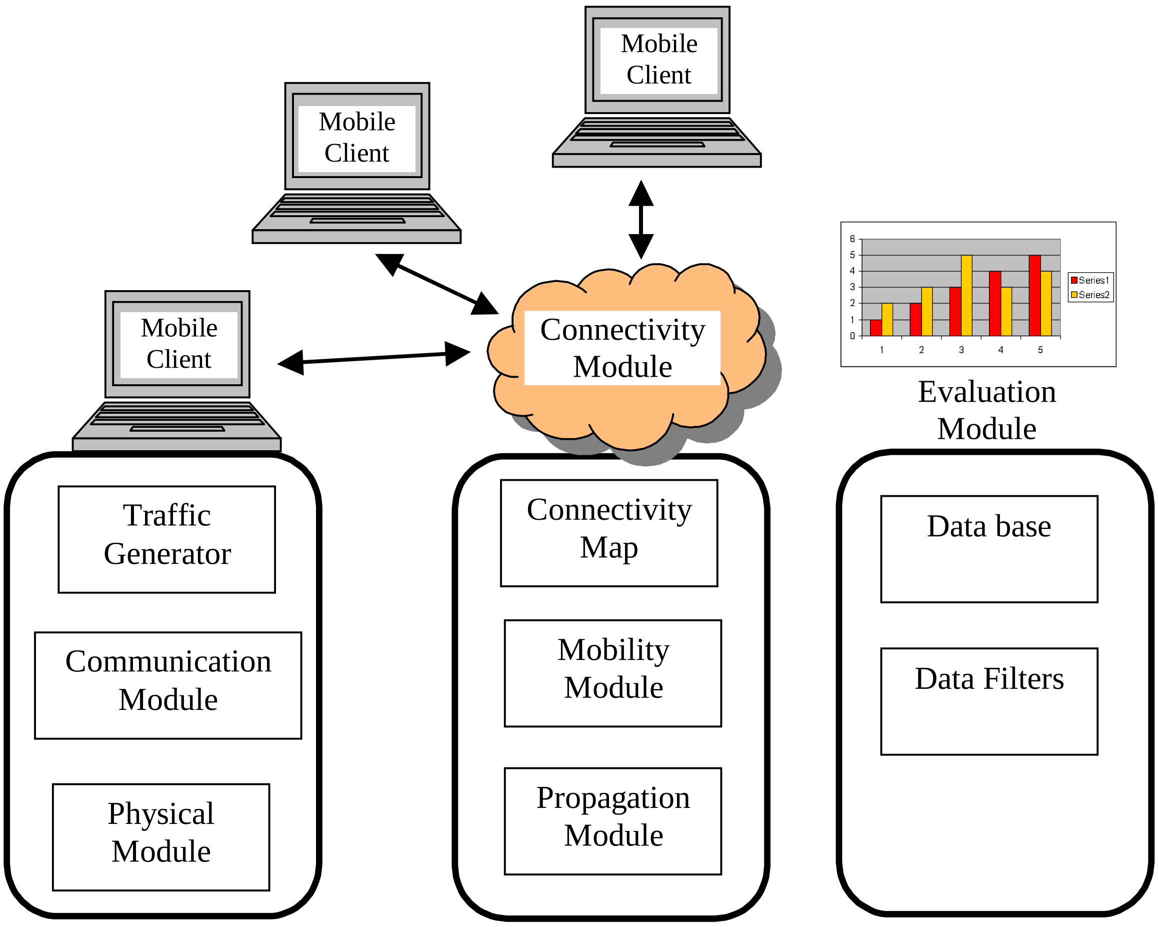


Figure 1. Logical architecture of the ad-hoc simulator

- **Connectivity module:** the connectivity module realises the connections among the mobile clients. This module knows the topology of the network. The network is described by means of a graph. The graph can be generated randomly or according to a predefined topology (so we can simulate a real scenario e.g. in a building) during the initialisation phase. After the initialisation the connectivity graph is modified according to the mobility module.
- **Radio propagation submodule:** contains different radio propagation models for indoor and outdoor environment, such as open air, Rayleigh and Rice fading, etc. By means of the radio range of a given terminal can be calculated.
- **Mobility submodule:** this module describes the mobility customs of the users. At this stage of the system our network has fixed topology.
- **Evaluation module:** it is responsible to collect and store measurement data during a simulation and the presentation of the results. It contains filters which work on the stored data to provide the required information.

Now in the following we summarise the simulation steps:

Initialisation

During the initialisation the network topology and the terminal nodes are generated according to the different modules and several global parameters, such as the number of terminals, the routing protocol, etc. Based on the radio propagation submodule, transmission power and the connectivity graph in our demo the program calculates the distance among the terminals. This procedure is executed for every node. If the distance between two nodes is less than the radio range, the nodes will become to neighbours. Two nodes can communicate with each other, if one of the nodes is within the radio range of the other node.

Running the simulation

After starting the simulation the mobile clients initiates traffic according to the traffic model set up. During the traffic generation they move if the mobility module force them to do it. The delivery of protocol and data messages can be followed on the screen and parallel the evaluation module stores them in a database. Of course the movement of mobile clients may cause the modification of the connectivity graph. In Figure 2. a typical ad-hoc scenario can be seen. The white cloud in the middle represents the air (connectivity module) and the small computers around it refer to the mobile clients. In this view the physical structure of the network is hidden into the connectivity module only logical structure is shown. The black arrows show the logical connection to the connectivity module and the red circles denote the protocol messages travelling between clients.

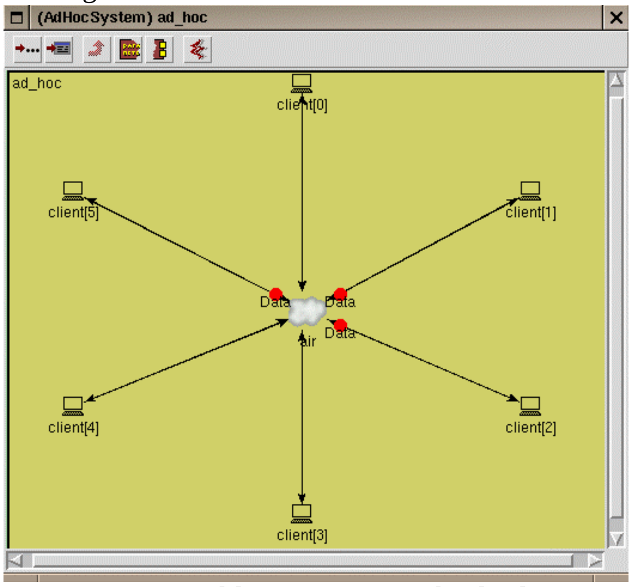


Figure 2. Ad-hoc scenario on the display

In order to provide address resolution, in this demo every node sends a broadcast message to its neighbours. The broadcast message contains the address of the sender node. Thus the nodes will know the address of their neighbours. When the receiver module receives a broadcast message, it gets an “access info” from it and it sends to the sender module. This info contains the data of the neighbour node and it is stored in a dynamic array in the sender module.

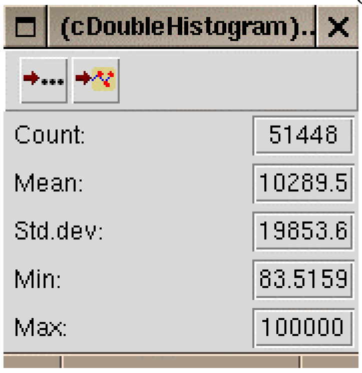
When the receiver module of the given client receives a data message it checks the “destination address” of the message. If the address is equal to the address of the mobile then it processes the data else it puts the data into the queue. If there is data in the queue, the sender module checks the “destination address” of the data and starts the routing procedure. The selected routing procedure searches the path to the destination and it gives the next hop too. When the channel is free the sender sends the data to the next hop via the connectivity module.

If two nodes are within each other radio ranges they must not send data simultaneously, because they would jam each other. Our medium access protocol in our demo is the following: An RTS message is sent to all neighbours before the node sends data. Parallel RTSes are discarded. The destination node sends the CTS and the data transmission starts. A “channel free” message is sent to all neighbours when the node finished the data sending. Thus the CTS message indicates to the neighbours that they must not send data and the final message indicates that the channel is free. Exponential backoff is used since this is an emulation of the CSMA/CA medium access protocol.

Evaluation

The evaluation module continuously collects information about different events during the simulation (e.g., message was sent or received, connection was established or released, etc.)

in a log file. Using the data stored in the log file and the filters working on them appropriate graphs can be generated or the whole simulation can be played back step by step or faster then the simulation speed. The simulation environment supports statistic data collecting. It provides some statistical calculations on the collected data for example: minimum value, maximum vale, mean, deviation, average and also histograms can be displayed.
On Figure 3. and Figure 4. two examples among the measurement results are shown. Figure 3. depicts the characterisation of a 100 kbps radio link throughput in terms of minimum, maximum and average throughput and the standard deviation (for one user).

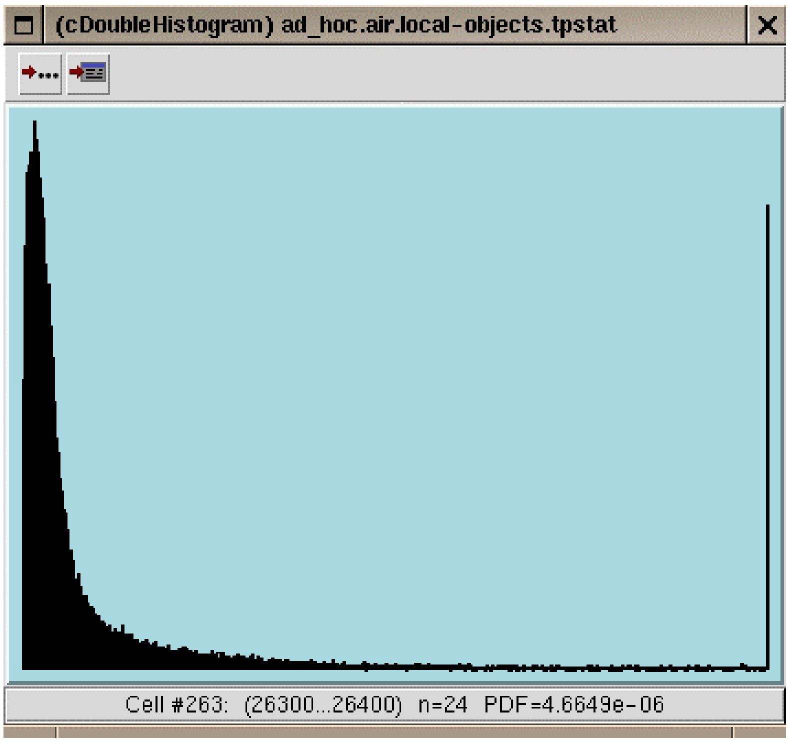


Count:	51448
Mean:	10289.5
Std.dev:	19853.6
Min:	83.5159
Max:	100000

Figure 3. Minimum, maximum, average throughput of the network and its standard deviation

In this long simulation (51448 packets are sent by the surveyed node) 6 nodes were placed and the simulation area was larger than one node's radio range. So this was a multihop scenario where the selected node has to relay packets as well.

On Figure 4. we present a throughput histogram of received packets versus link capacity. The horizontal axis refers to the link capacity from 0 kbps up to 100 kbps in 0.1 kbps steps. The vertical axis contains the number of the received data packets.



Cell #263: (26300...26400) n=24 PDF=4.6649e-06

Figure 4. Histogram of the network throughput (see details in text)

IV. Conclusions and further work

In this paper we gave a short survey about our new ad-hoc simulation environment. We summarised its capabilities to support of testing different ad-hoc protocols and solutions. In the near future we plan to extend the system with the mobility module and we would like to increase the number of implemented routing protocols. Of course new user traffic models should be involved according to the development in mobile computing applications. Finally the TCP/IP protocol layer has been realised in OMNET++, so we plan to build in these layers. Thus we would be able to measure TCP/IP traffic in ad-hoc networks.

References

[GUP00] P. Gupta, P. R. Kumar: "The Capacity of Wireless Networks", IEEEE Transactions on Information Theory, March 2000, Vol. 46. No. 2. Pp. 388-404.

[CHA01] S. Chakrabarti, A. Mishra: "QoS Issues in Ad Hoc Wireless Networks", IEEE Communications Magazine, February 2001, Vol. 39. No. 2. Pp. 142-149.

[OMN01] http://www.hit.bme.hu/phd/vargaa/omnetpp/

[H2] ETSI TS 101 761-4 V1.1.1, "Broadband Radio Access Networks (BRAN); HIPERLAN Type 2;Data Link Control (DLC) Layer; Part 4: Extension for Home Environment"