Software Tools for Network Modelling and Simulation

Marek Huczala

Abstract – **The presented paper introduces Java SSFNet platform for computer network modelling and simulations. The SSFNet platform is a Java-based software interface that might be implemented on its own or with a co-operation with an upper level software kit. First, we discuss the library classes for the network scheme development and the ways of launching the SSFNet network simulation process. Later chapters provide an overview to building an upper level software interface to interact with SSFNet models.**

Keywords – **Java, SSFNet platform, NetSim, network, simulation, modelling.**

I. Introduction

Network modelling and simulation plays an important role in today's computer network design. The main intention of the following paper is, however, to outline the "software paths" that would lead into an improved network model design and a better overview of the simulation results. The paper could be a developer's key to organizing and running a new simulation process under the SSFNet platform.

Since all the SSFNet software has been written in Java programming language, here mentioned software solutions are also based on Java. Using the solution, any developer will be now able to easily define a new network model and launch the simulation by calling SSFNet.

II. Overview of Java SSFNet Platform

The SSFNet is a collection of Java SSF-based components for modelling and simulation of Internet protocols and IP based computer networks. By default, the SSFNet components are represented by Java pronsipal classes that were later united into the following two main software frameworks:

- SSF.OS is used for modeling of the host and operating system components. Network and transport layer protocol such as SSF.Net.IP and SSF.Net.TCP are laid on top of SSF.OS class.
- SSF.Net is used for modeling network connectivity, creating nodes and link configurations. It loads all the model's configuration file and controls the orderly instantiation of the entire model: hosts and routers with their protocols, links connecting hosts and routers, as well as traffic scenarios and multiple random number streams.

Marek Huczala, Ing., PhD student at Telecommunications Department, Faculty of Electrical Engineering and Communication Technologies, Brno University of Technology; Purkyňova 118, 612 00 Brno, Czech Republic. E-mail: huczala@kn.vutbr.cz.

III. Network Model Design

The network configuration is stored in DML scheme definition file. New DML file uniquely describes the complete network architecture from both hardware and software aspect. The configuration file follows the DML syntax structures that allow keyword, value specifications. DML syntax grammar is based on standardized well-known XML structure.

Network file scheme definition begins with keywords scheme and Net as it follows:

```
schemas [
 Net [
        frequency  simulation_runtime
```

The frequency attribut specifies the total time of simulation process. Network, always defined by the Net keyword, represents a set of hosts, routers, links and when modelling more complex network environments even subnets.

Here is an example of a subnet definition using the Net keyword:

```
Net [
  id id_no
  idrange id_no from --- to ---
  .
  .
  .
  ip net_mask_def
  _extends .schemas.Net
]
```

The id and idrange attributes are used for subnet identification while ip attribute passes on network mask definition. The _extends attribute specifies the higher level syntax used - in current example it follows the default Net scheme. The SSF.Net.host intruduces the following host configuration scheme:

```
host [
  id id_no
  idrange id_no from --- to ---

  ]
```

Router definition very likely follows the host definition. Interfaces of the implemented network elements are described by a local keyword interface followed by a set of attributes such as bitrate, latency or virtual. Links connecting network nodes are set up by link keyword whereas the traffic flow between nodes is defined by traffic keyword and its attributes. Every fragment of the DML definition file is processed by a

corresponding class. For example host definition is being handled by SSF.Net.host class while links by SSF.Net.link principal class.

By default, the network definition scheme is interpreted by SSF.Net.Net class. It loads the complete network model, network elements such as routers, hosts, protocols and links specifications as well as network traffic scenario.

IV. Overview of SSFNet Pre-simulation Process

Pre-simulation process is usually invoked by the main function of SSF.Net.Net class. The process itself comprises of several stages. The first stage involves schemantical DML scheme check:

```
if (doSchemaCheck)
  netconfig.check();
  Configuration netcfg =
(Configuration)netconfig.findSingle(".Net");
 if (netcfg == null) {
      System.exit(-1);
      }
if (null != netconfig.findSingle(".link") ||
      null != netconfig.findSingle(".router") ||
      null != netconfig.findSingle(".host")) {
               System.exit(-1);
            }
```

Within this pre-simulation stage some essential parameters are being checked, such as link, router or host. The IP address space of all networks and subnets is being allocated in the 2nd stage. Routing informations are added subsequently. Links and connections between network nodes are being checked in the end of the pre-simulation process.

At programmer's view, the SSF.Net.Net object will use the services of the DML library to load the content of the configuration files and net.dml (by default) into a runtime Configuration database object. After that, SSF.Net.Net will systematically instantiate and configure all simulation objects such as hosts, routers, protocols, and network links. Once all simulation objects have been instatiated, the initialization phase begins by calling init() methods of all the entity subclasses. Finally SSF.Net.Net invokes its method startAll(), and the simulation begins with simulation time value equal to 0 sec. There is a lot of verbose output, including the automatically generated IP address blocks etc., that may be suppressed by command line options to SSF.Net.Net.

The simulation runtime may be specified as a command line argument to SSF.Net.Net class or directly inside the configuration scheme file. The Java Development Kit 1.3 or higher is required for running all the SSFNet simulations.

V. Developing SSFNet Applications

The simulation results as they come out of the SSFNet simulation process can only be viewed in a text mode. This was the main reason for me to start building a new application that would provide a graphical interface to SSFNet simulation. Knowing this and with a knowledge of Java programming language, now anyone himself could build a new graphical application interface that would show the SSFNet simulation results in a windows-like style frame.

This chapter briefly describes a newly build graphical interface to SSFNet Simulation, called NetSim as well as other ways to extend the SSFNet simulation framework.

Only small adjustments to the SSFNet source code were made while building NetSim. First of all, the pre-simulation phase needed to be a little more transparent. This required adding small code fragments to the SSF.Net.Net class. The purpose of the change was to enable a possibility to monitor the current state of the pre-simulation process. Secondly, I added the dimensions to the host and router elements in network scheme DML file structure which resulted in a contructors' change of SSF.Net.Host and SSF.Net.Router classes. The dimension parameter is now being used by JGraph (see www.jgraph.com) graphical environment to draw the picture of network scheme. The network scheme picture is shown in an internal frame, see Fig. 1.

NetSim is fully open for further adjustments and improvements. Adding the possibility to trace the simulation process could be one of them.

The NetSim application window consists of a network scheme internal frame, state bar showing the current simulation state and the simulation results frame. Figure 1 shows network scheme and results of a simple client-server simulation running on the SSFNet platform. Simple client-server communication model is used in this example. The picture of a client-server scheme is drawn during the model initialization process in the upper frame. The simulation results are shown in the bottom of the main window.

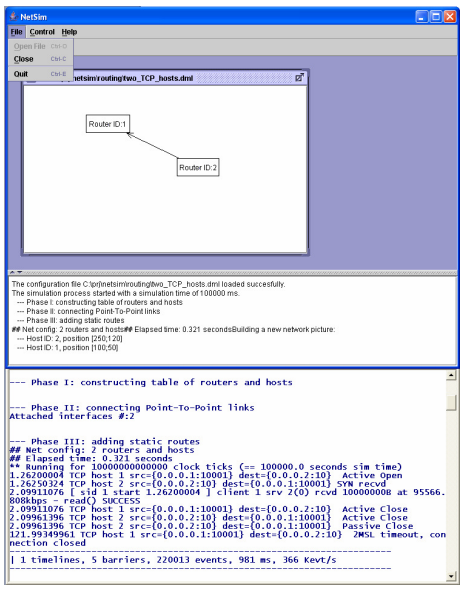

Fig. 1. Sample graphical user interface (GUI) application showing results of the SSFNet network simulation.

The advantages and disadvantages of using NetSim come from the properties of Java programming language. The object-oriented program code is simple to read and easy to change and simulations can be run under different operating systems, including Windows or Linux. The application NetSim and simulation platform SSFNet, however, require a huge memory space when modelling complex network environments.

VI. CONCLUSION

The paper introduces a Java-based platform for network modelling and simulations SSFNet. SSFNet kernel and its source code can be easily downloaded and installed from www.sffnet.org.

NetSim is a Java-based graphical user interface that was developed to graphically demonstrate the results of SSFNet simulation process. The Jgraph (www.jgraph.com) graphics library is used to picture the detailed network scheme.

Only small adjustments to the SSFNet source code were made while building NetSim. The application and the SSFNet platform is fully open for further adjustments and improvements. Adding the possibility to trace the simulation process could be one of them.

The NetSim package and the installation instructions can be downloaded from the authors web site (http://hawk.cis.vutbr.cz/~huczala/vizualizace).

REFERENCES

[1] HUCZALA, M. Vizualization of routing algorithms in TCP/IP network environments, final report to the FRVŠ grant project, Brno 2005.

[2] SSFNet Community. SSFNet 1.3 DML Reference, www.ssfnet.org.

[3] SSFNet Community. Implementation and Validation Tests, http://www.ssfnet.org/Exchange/tcp/index.html.

[4] SSFNet Community. SSFNet software exchange, Package overview, http://www.ssfnet.org/exchangePage.html.