NETWORK SIMULATIONS WITH JAVA SSFNET PLATFORM

Ing. Marek HUCZALA, PhD Degree Programme,
Dept. of Telecommunications, FEEC, BUT
E-mail: huczala@kn.vutbr.cz

Supervised by: Dr. Vladislav Škorpil

ABSTRACT

The following 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 cooperation with an upper level software kit. First, we discuss the library classes for the network scheme development and the ways of launching the network simulation process. Later chapters briefly describe a newly developed graphical user interface into the SFFNet platform. The use of the simulation sofware is outlined in the end of the paper.

1 INTRODUCTION TO NETWORK MODELLING AND SIMULATIONS

Network modelling and simulation plays an important role when designing large network infrastructures both heterogenous and homogenous. The main intention of here presented simulation software toolkit is, however, to graphically demonstrate the funcionality of different network standards. Hopefully it will help students to understand the basics of network communication.

The new simulation application was built in Java programming language on top of SSFNet platform for network modelling and simulations. Any user of the application can now easily define a new network model using DML syntax and launch the simulation either by calling SSFNet simulation process or via NetSim window's menu.

Next chapters provide an overview of both SSFNet platform and NetSim application.

2 JAVA SSFNET PLATFORM

The SSFNet is a collection of Java SSF-based components for modelling and simulation of Internet protocols and IP (Internet Protocol) 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.

3 BUILDING A NEW NETWORK SCHEME

The network configuration is stored in DML (Dynamic MarkUp Language) 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.

4 LAUNCHING 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.

5 GRAPHICAL USER INTERFACE – NETSIM

The simulation results as they come out of the SSFNet simulation process can only be viewed in a text mode. The main purpose of building a new graphical application NetSim is to be able to demonstrate the results in a windows-like shape.

The application window consists of a network scheme internal frame, state bar showing the current simulation state and the simulation results frame. The network scheme is being drawn during the pre-simulation process inside the scheme internal frame. The information included in the network configuration file is used for this purpose. The window's state bar shows the the current simulation state. The bar is invoked by the pre-simulation process. The

simulation results are shown in the very bottom part of the application window. The application's toolbar consists of two icons which might be used in later development stage for for tracing the simulation process. Figure 1 shows network scheme and results of a simple client-server simulation running on SSFNet platform.

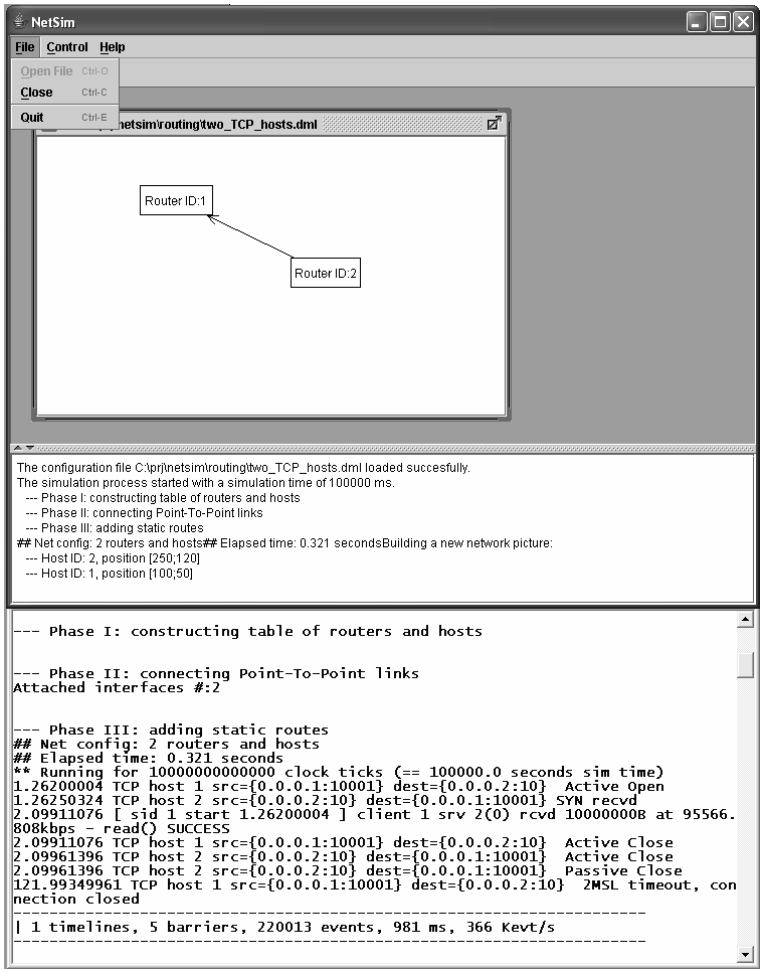

Fig. 1: *The NetSim application window shows results of a simple SSFNet client-server simulation.*

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.

6 CONCLUSIONS

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.

NetSim 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.