Survey of simulators of Next Generation Networks for Studying Service Availability and Resilience

L. Begg, W. Liu, K. Pawlikowski, S. Perera and H. Sirisena

Technical Report TR-COSC 05/06

Department of Computer Science & Software Engineering

University of Canterbury

Christchurch, New Zealand

15 February, 2006

Survey of simulators for studying service availability and resilience mechanisms in Next Generation Networks

It is expected that discrete-event simulation will be an important method of our study of service availability and resilience in the Next Generation Networks (NGNs). Thus, application of an efficient simulator, which can allow gradual addition of the required elements of NGNs and their functionalities, for evaluating quality of services the NGNs could offer, is an important part of this project. Such a discrete-event simulator, used in modelling and evaluation studies of service availability and resiliency mechanisms in NGNs, will be further referred to as an **NGN simulator**. This document presents results of a survey of the most popular simulators of telecommunication networks, including simulators suggested by other teams participating in this NGN project. A custom-built simulator is also included.

The results of this survey should help to select the most appropriate simulation tool(s) needed in research leading to Outcome 4 (Measurements), Outcome 6 (Service Availability Model Definition), Outcome 7 (Network Component Selection) and Outcome 8 (Prototype Development).

This report does not include discrete-event simulators/emulators with “software-in-loop”, which have been separately surveyed by our colleagues from the University of Waikato.

First, we will present the criteria used in our evaluation of selected simulators and then the results of our survey in which these criteria have been applied.

CONTENTS

1. Evaluation criteria 4
1.1 Modeling Capabilities 4
1.2 Credibility of Simulation Models 5
1.3 Credibility of Simulation Results 5
1.4 Extendibility 7
1.5 Usability 7
1.6 Costs of Licenses 8
2. Evaluation of simulators 9
2.1 OPNET 10
2.2 NS2 17
2.3 OMNET++/OMNSET 22
2.4 GLASS/SSNFNet 28
2.5 QualNet 33
2.6 JSim (version 1.3) 37
2.7 TOTEM (version 2.0) 42
2.8 Custom-made simulator 46
3. Comparison of simulators 49
4. A simulation case study 53
4.1 Simulation in OPNET
4.2 Simulation in NS2
4.3 Simulation in QualNet
4.4 Conclusions
4. Final conclusions
5. References 85
6. Appendix A. List of requirements 86

1. Evaluation criteria

We have assumed that, when surveying various simulation tools which could be used in evaluation studies of service availability and resiliency mechanisms in NGN, their following features should be taken into account:

- Modelling Capabilities,
- Credibility of Simulation Models,
- Credibility of Simulation Results,
- Extendibility,
- Usability,
- Costs of Licenses.

In addition, each potential simulator will be used in a simple simulation of an NGN, to gain practical experience and to assess the level of user-friendliness of simulators considered.

Let us consider the features listed above in more details.

1.1 Modeling Capabilities

This criterion is related to the specific use of the simulation tool in our research project on service availability and resilience in NGNs.

Our NGN simulator would be used for simulating converging network technologies based on PSTN and IP networks into NGN. Thus, it should support various NGN elements and functionalities. It should also allow evaluation of service availability of Voice-over-IP (Void) and other multimedia services offered by NGN, as well as to explore resiliency and QoS strategies in NGN. Thus, its users should be able to freely use:

R1: Models of VoIP protocol stacks.
R2*:* Models of MPLS-TE and RSVP signaling protocols.
R3*:* Models of typical failures and different routing protocols.
R4*:* Models of optical layer components.
R5*:* Models of different teletraffic scenarios.
R6*:* Models of Quality of Service (QoS) mechanisms.
R7*:* Models of different network architectures.

As in any quantitative studies of networks, the NGN simulator should also allow to analyse various performance measures. Thus, the next requirement is:

R8: Analysis of typical performance measures.

Finally, since our plans involve simulation studies of large networks, setting initial network configurations manually would be too time consuming, if it was possible to do at all. Thus,

ability to input and output data using a standard file format is desirable. It has been decided earlier in this research project that XML files should be used. Thus, the NGN simulator should accept:

R9: Input/output data in XML format.

These features/requirements are described in details in Appendix A.

1.2 Credibility of simulation models

Simulation models are credible if they are valid and verified. A model is valid if it represents a given system (e.g. a communication protocol) accurately, at the required level of details. Verification is concerned with determining whether the simulation model has been correctly translated into a computer program[1].

When assessing credibility of simulation models used by a given network simulator, one needs to assess his/her level of confidence that the simulation models supported by that simulator conform to the existing standards.

In general, distributors of commercial simulators take full responsibility both for validation of simulation models and verification of their software implementations. On the other hand, in the case of simulators distributed as freeware, nobody takes responsibility for offering valid models encoded in verified programs. Then, ensuring credibility of simulation models is left for individual users of a given simulator.

R10*:* Credibility of simulation models

1.3 Credibility of simulation results

Assuming that a network simulator uses credible simulation models, credibility of the final results it produces depends on (i) the quality of its sources of randomness (i.e. the quality of its pseudo-random number generator(s) or statistical representativeness of traces of real teletraffic) and (ii) statistical accuracy of the final simulation results, [2].

These two aspects of credibility of the final simulation results can be summarized by the following requirements. The first is to use a high quality of sources of randomness. Thus, selecting a simulator one should assess

R11*:* Quality of sources of randomness

The current computing technology has made any linear **Pseudorandom Number Generator (PRNG)** with the cycle length shorter than 2^{70} obsolete (if one uses a workstation with the CPU clock at about 3 GHz, and simulations last 1 hour or longer), [3]. Classical PRNGs with cycle lengths of the order of 2^{31} could be still acceptable only when using much slower CPUs (but note that the simulations could not be too long then too, as this could make such PRNGs useless too).

It is generally known that the only practical way of obtaining meaningful/accurate final results from any simulation is to analyze output data on-line, during simulation. Then, the simulation can be stopped when the statistical errors of the estimates become sufficiently small. Such approach is known as **sequential on-line analysis of simulation output data**. If a simulator does not support on-line analysis of simulation output data, then the only way of obtaining the final estimates with acceptably small errors is to repeat simulation a number of times, using statistically independent sequences of pseudo-random numbers. Such technique of simulation is known as **Independent Replications**. After simulation of a number of replications is finished, one can analyse the collected observations (sequences of simulation output data) off-line. The problem with such approach is that one cannot guess in advance how many replications is needed for securing small errors of estimates, and if the errors are found to be too large, simulations need to be repeated. This is referred to as **off-line sequential analysis of simulation output data**. Of course, this is not a very efficient way of data analysis.

R12*:* Sequential (on-line or off-line) analysis of simulation output data.

Note that conducting Independent Replications is fully possible if users of a simulator can access consecutive numbers generated by its PRNG, to ensure that replications are really statistically independent. If the last pseudo-random number used in a series of replications is not known and the number of replications were too small (i.e. the errors of results are still too large), then simulation needs to be repeated from the beginning, over large number of replications.

There is still one solution for controlling final errors of simulation results even if a simulator does not support sequential on-line analysis of output data: **the simulator can be linked with a sequential controller of stochastic discrete-event simulation** that allows non-sequential simulator to execute simulations sequentially. Currently there exists only one such controller of simulation. It is known as Akaroa2 and has been designed by the Simulation Research Group, at the University of Canterbury in Christchurch [4]. This simulation controller has been widely used by scientists at many universities and research laboratories in over 70 countries over the world. Commercial applications of Akaroa2 require a license which should be discussed with the Canterprise, a commercial arm of the University of Canterbury in Christchurch; see www.cosc.canterbury.ac.nz/research/RG/net_sim/simulation_group/ . The cost of its commercial license would be a few thousand dollars.

Note that Akaroa2 allows not only to sequentially control errors of results produced by non-sequential simulators but it is also to significantly speedup such simulations. The fact is that network simulations producing estimates with small errors may need be very long; especially when simulated processes are correlated (as we know, many processes in modern telecommunication networks are highly correlated). Thus, Akaroa2 reduces the total simulation time by employing parallel processing. It automatically launches multiple replications of simulated network scenarios on multiple workstations of a LAN. This solution for fast sequential simulation is known as MRIP (Multiple Replications In

Parallel). Akaroa2 is able to automatically launch hundreds of replications in parallel, and the resulted speedup of such simulation can be linear [5].

Thus, one can formulate the following requirement for achieving credible final results even if the original simulator is unable to do it:

R13: Possible cooperation with Akaroa2, an automated sequential controller of simulation in MRIP scenario.

1.4 Extendibility

Extendibility of a simulator means its ability to be expanded, for example by adding new features to existing simulation models or adding new models. An important factor here is the amount of work / time needed to extend the existing simulation models.

In the context of our research on NGN, it would be important to be able to combine protocols under development/investigation with already established/standardised protocols at different layers. In particular, this should also include integration of VoIP (Voice Over IP) protocol stack and various multi-layer resiliency mechanisms.

Thus, the next requirement can be:

R14: Ability of extending/adding new simulation models of protocols and mechanisms in a reasonable time.

1.5 Usability

Usability of a given simulator is measured by the level of its user-friendliness.

A user-friendly simulator should be equipped in a Graphical User Interface (GUI) which could be used both as an input interface, possibly for constructing the simulated network scenario in a graphical way, and as an output interface, for graphical representation of output results. During simulation, the GUI could be used for showing evolution of simulated processes and/or intermediate values of analyzed performance measures. Thus, we can require

R15: Existence of GUI, which can be used as input/output interface.

Another important aspect of user-friendliness of a simulator is existence of a good manual, together with an introductory tutorial allowing users to learn how to use that simulator in a short time. An access to an on-line technical support or user group, where problems related with use of a given simulator could be addressed, would be also beneficial.

R16: Existence of a good manual and an introductory tutorial, explaining how a given simulator can be used.

1.6 Cost of licenses

The cost of a simulator is simply the price of acquiring this tool and using it for commercial purposes.

The costs of some commercial licenses of network simulators are known to be substantial. Thus, when comparing simulators, it is also important to determine:

R17*:* The cost of a commercial license of a given simulator.

2. Evaluation of Simulators

We will survey the following network simulators, assessing them in line with the requirements R1-R17:

- OPNET, see http://www.opnet.com ;
- NS2 , see http://www.isi.edu/nsnam/ns/ ;
- OMNet ++ , see http://www.omnetpp.org/ ;
- GLASS/SSFNet, see http://www-x.antd.nist.gov/glass/ ;
- QualNet, see http://www.qualnet.com/ ;
- J-Sim, see http://www.j-sim.org/ ;
- TOTEM, see http://totem.run.montefiore.ulg.ac.be/ ; and
- a custom-made simulator.

Simulators deemed as unsuitable candidates for the NGN simulator will be eliminated if:

1. a given simulator does not have a satisfactory library of simulation models for protocols and network mechanisms needed for investigating network resilience issues in NGN;
2. the credibility of simulation models is questionable;
3. it is difficult to extend the size of simulation models or add new features to simulated networks;
4. credibility of the final simulation results is questionable, or producing credible final results is difficult.

While assessing simulators we will use the following grading scheme:

+++	**excellent**
++	**satisfactory**
+	**poor**
-	**unsatisfactory, or a missing feature**
?	**unknown**

Each simulator will be individually assessed and then they will be mutually compared. Their main similarities and differences will be assembled in one table. Simulators selected as candidates for our NGN simulator will be further assessed in a special simulation case study, a practical trial during which a simple NGN-related case will be simulated. The

experience gained during these trials should additionally help us to choose the most appropriate simulator for our studies of problems related with service availability in NGN.

2.1 OPNET

OPNET (Optimized Network Engineering Tools) Technologies offer a large number of different tools supporting modelling and simulation of networks in various technologies. They are available for users who buy appropriate licences. There are two main types of licences: academic or commercial. We will consider the main product of OPNET Technologies, known as **OPNET Modeler** (in its version 11.0).

2.1.1 OPNET Modeler (version 11.0) : Overall description

OPNET Modeler, further simply referred to as OPNET, has a GUI with a (high) number of various editors for creating, modifying and verifying models, for running simulations, as well as for displaying and analyzing simulation output data. User processes run on top of a C compiler. Simulation models are built in a hierarchical fashion. Models can be built of existing components in either top-down or bottom-up way, and each level represents the internal structure and functionality of the level above. The following four levels are used:

- Network level[1]
  Modelling of network topologies and overall configuration takes place at this level of modelling. Network elements such as communication links and node devices are used to build the model. In addition, node/link failure/recovery processes can be modelled at this level.
- Node level
  At this level the internal structures of network level devices are modelled. Elements used for modelling include: generic processor modules, queue modules, receivers and transmitters. These are interconnected by streams or statistic wires.
- Process level
  Functionalities of node level devices are modelled at the process level, by means of finite state machines (FSMs).
- Proto-C level
  The lowermost modelling level is called the proto-C level. Proto-C is an extension of the C (or C++, depending on the underlying compiler) programming language. A large number of kernel procedures are available. All built-in models are available at this level as source code.

[1] Unrelated with OSI Layer 3

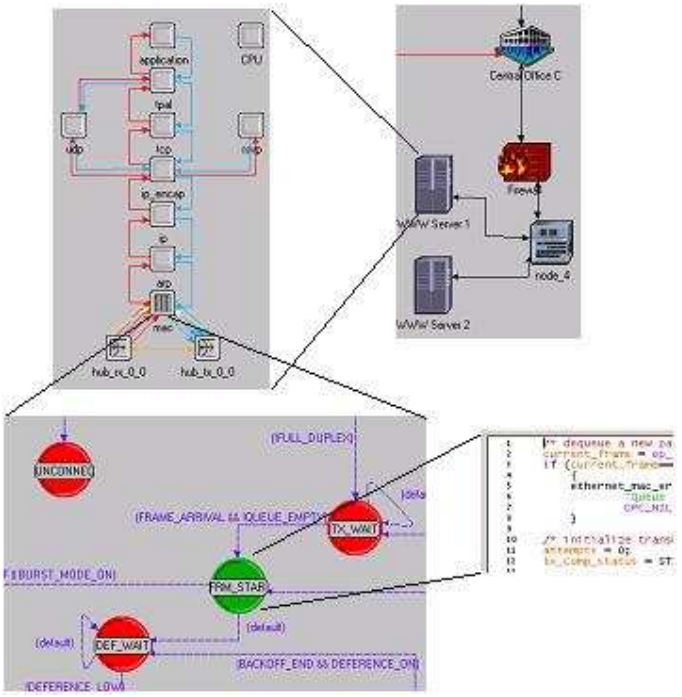

Figure 1: The hierarchical structure of OPNET models

This hierarchical level structure is shown in Figure 1. Models can be edited at each level, and more details can be incorporated at any level. A vast number of models of protocols and devices are available, as one can use libraries of other products of OPNET Technologies (if additional licenses are purchased).

Simulation is carried out from the GUI or from the command line. Before running a simulation, the desirable output statistics should be selected. During simulation the statistics are written to files – either scalar files or vector files depending on the type of output data.

The simulation kernel itself is very efficient – the version 11.0 of OPNET Modeler has been carefully optimized and can run both on single and multi-processor computers. However, its complexity causes that in more complicated, large-scale network studies the simulation executed under OPNET can be slower than under other, "lighter-handed" simulators, such as for example NS-2 [6].

2.1.2 OPNET Modeler: Modelling capability

- **R1: Models of VoIP protocol stacks.**

Grade: ++ (satisfactory, credibility of some models needs to be confirmed)

Partially supported

OPNET Modeler library contains models of SIP and RTP. Some other models, such as models of MGCP and H.248/Megaco can be found too, although they have been developed by OPNET's user community, however OPNET Technologies take no responsibility for their correctness.

- **R2: Models of MPLS-TE and RSVP signaling protocols**

Grade: +++ (excellent, fully supported)

OPNET MPLS model incorporates all details of MPLS technology and related traffic engineering policies. Based on Internet standards and developed in collaboration with industry experts, MPLS model features include:

  - configuration of Dynamic / Static LSPs;
  - protection and restoration mechanisms such as e.g. fast reroute;
  - compatibility with Layer 3 of VPNs;
  - full set of traffic engineering policies;
  - signaling related with LDP and RSVP-TE.

- **R3: Models of typical failures and different routing protocols**

Grade: +++ (excellent)

OPNET Modeler fully supports modelling of node/link failures and recovery. Nodes or links can be set to fail or recover at a specified time during simulation.

OPNET Modeler offers also models of different routing protocols, including OSPF, IS-IS, BGP and RIP.

- **R4: Models of optical layer components.**

Grade: ++ (satisfactory, another OPNET product needs to be bought)

Not supported by OPNET Modeler, but fully supported by WDM Guru, another OPNET product. OPNET Modeler offers only SONET physical model, as it does not support optical layer simulation. WDM Guru supports services and infrastructure at the SONET/SDH, wavelength and fiber levels.

WDM Guru is an advanced network planning tool aimed at service providers and network equipment manufacturers, for designing resilient, multi-layered and cost-effective optical and SONET networks. At each layer, advanced network design algorithms enable to minimize investment costs and optimize operational efficiency in mesh and ring-based architectures.

- **R5: Models of different teletraffic scenarios.**

Grade: +++ (excellent)

OPNET provides a number of models for traffic characterizing typical applications, such as e.g. SIP, Video, HTTP, FTP. In addition, fully customized models of traffic scenarios can be easily developed.

- **R6: Models of Quality of Service (QoS) mechanisms**

Grade: +++ (excellent)

QoS mechanisms, as referred to a set of service requirements to be met by the network while transporting a flow of traffic, have been fully implemented in OPNET Modeler. The features of the QoS model include:

- Traffic Classification
- Marking
- Committed Access Rate (CAR) / Traffic Policing
- Random Early Detection (RED and WRED)
- Priority Queuing (PQ)
- Low Latency Queue (LLQ)
- Custom Queuing (CQ)
- Weighted Fair Queuing (DWFQ and CB-WFQ)
- Weighted Round Robin (DWRR , MDRR and MWRR).

- **R7: Models of different network architectures.**

Grade: +++ (excellent)

OPNET Modeler supports different kinds of network architectures, for various freely configurable, interconnected, hierarchical and/or non-hierarchical networks.

- **R8: Analysis of typical performance measures.**

Grade: +++ (excellent)

OPNET Modeler allows to analyse the full spectrum of performance measures. Additionally, it is possible to specify a set of simulations and then sweep over a range of input parameters and/or traffic scenarios. During simulation, one can collect statistics about network performance at different levels, e.g. to assess the global performance of a given network as well as the performance of its nodes and links. In terms of output analysis, either the built-in features for gathering statistics can be used or all data can be exported to an external analysis tool.

- **R9: Input/output data in XML format**

Grade: +++ (fully supported)

OPNET provides different options for importing network input data from an external data source. The following list shows data sources from which input data can be imported:

- ATM text files
- VNE Server
- HP Network Node Manager
- XML files
- Device Configuration Data

2.1.3 OPNET: Credibility of simulation models

- **R10: Credibility of simulation models**

Grade: ++ (satisfactory, credibility of some models needs to be confirmed)

OPNET is a commercial simulation tool, they have full responsibility on credibility of simulation models, but some models are contributed by its user community, which are not verified and maintained by OPNET.

2.1.4 OPNET: Credibility of simulation results

- **R11: Quality of sources of randomness**

Grade: + (potential problems in large scale simulations)

The pseudo-random number generator (PRNG) used by OPNET has not been revealed by designers of OPNET. However, a discussion within Modeller Community at OPNET User Forum (see http://forums.opnet.com) suggests that OPNET uses a ported version of the **random()** function found in UNIX BSD source distribution. This is a nonlinear additive feedback random number generator with the cycle length of approximately $16*(2^{31}-1)$. Thus, in larger and stochastically dynamic simulations on fast processors, when larger number of events needs to be simulated, or when dealing with strongly correlated stochastic processes, or when larger samples of output data are needed for producing accurate estimates, this generator can have a too short cycle for avoiding repetition of the randomness. However, this PRNG should be of adequate quality for smaller to medium scale simulations of not-too-strongly correlated processes.

- **R12: Support of sequential, on-line/off-line analysis of simulation output data.**

Grade: + (poor, potential problem with producing sufficiently accurate final estimates of estimates with larger variance).

There is no support for on-line sequential analysis of output data. Only off-line data analysis is supported, but for larger and stochastically dynamic, when larger number of events needs to be simulated or larger samples of output data are needed for producing accurate

estimates, this approach can be unsatisfactory, since there can be a problem with producing sufficiently accurate final estimates.

- **R13: Possible cooperation with Akaroa2, an automated sequential controller of simulation in MRIP scenario**

Grade: - (not supported; accuracy of the final estimates cannot be secured by using an external on-line controller/analyser of simulation output data)
OPNET does not open the source code to allow linking it with such an external on-line controller/analyser of simulation output data as Akaroa2.

2.1.5 OPNET: Extendibility

- **R14: Ability of extending/adding new simulation models of protocols and mechanisms in a reasonable time**

Grade: +++ (excellent)
Fully Supported
OPNET is inherently extendible because it runs on top of a C-compiler. All provided models are available as source code, and can thus form a basis for further model development. The simulation kernel itself is very efficient – the version 11.0 of OPNET Modeler has been carefully optimized and can run both on single and multi-processor computers. However, its complexity causes that in more complicated, large-scale network studies, the simulation executed under OPNET can be slower than under other, "lighter-handed" simulators, such as for example NS-2 [6].

2.1.6 OPNET: Usability

- **R15: Existence of GUI, which can be used as input/output interface.**

Grade: +++ (excellent)

First, the GUI for OPNET presents several options to the user, such as a new project model, node model, process model etc. For example, the user can create a new project model or edit an existing one. Then, the GUI brings up a "world atlas", where users can view the existing network model, add new node models to the existing network or create a new network. By double-clicking on a node model inside the network model, users can access the processes modelled inside that node. Inside the node model, there are options for creating packet streams, processors, servers, queues, sinks etc. These are process models which can be user-defined via state-machines. The properties of these models can be set or users can pick models from an existing library. Inside the process models, there are state-machines, which consist of proto-C code. They can be edited in order to change their functionality.

- **R16: Existence of a good manual and an introductory tutorial, explaining how a given simulator can be used.**

Grade: +++ (excellent)

OPNET provides a good manual and an introductory tutorial allowing users to learn how to use that simulator in a short time. Also their response time to technical questions is within three working days.

2.1.7 OPNET: Cost of licences

- **R17: The cost of a commercial license of a given simulator.**

Grade: + (relatively expensive)

According to an OPNET representative for New Zealand and Australia, in January 2006, the cost of a commercial license of the basic package of Modeler was US$40,000, plus annual maintenance and support at 18%; total US$47,200. The MPLS module costed US$25,000 plus annual maintenance and support at 18%; total US$29,500. On January 24, 2006, to acquire both (Modeler with MPLS) would cost approximately NZ$115,000 dollars, subject to exchange rate.

2.1.8 OPNET: Strengths and Weaknesses

OPNET is a powerful discrete-event simulation tool that is widely used in industry because of its large library of simulation models and user's friendliness of its GUI the which makes modeling of networks very easy. It has also libraries of models of real-life equipment used in communication networking, such as typical routers and switches, including those used in wireless networks. These libraries are used to implement different protocols with varying input, output and behavior. OPNET has a broad portfolio for modeling, design, simulation and real-time assurance in context with detailed insight into infrastructure requirements . The only problem is that it does not support sequential, on-line analysis if simulation output data, so a "try-and-repeat" simulation is the only possible strategy for producing sufficiently accurate estimates by OPNET users.

Main strengths of OPNET:

- Large library of simulation models of communication protocols and equipment.
- Professional support
- Good documentation
- Large community of users including universities and industries

Main weaknesses of OPNET:

- It does not support sequential, on-line analysis if simulation output data
- Relatively high price

2.2 NS2

NS2, also written "ns2" (Network Simulator in its 2nd edition) began as a variant of the REAL network simulator in 1989 and has evolved substantially over the past few years. In 1995 NS development was supported by DARPA through the VINT project (Virtual InterNetwork Testbed) at LBL, Xerox PARC, University of California in Berkeley, University of California in San Diego and its Information Sciences Institute. Currently NS development is support through DARPA with SAMAN (Simulation Augmented by Measurement and Analysis for Networks) and through NSF with CONSER (Collaborative Simulation for Education and Research), both in collaboration with other research projects, including ACIRI (the Center for Internet Research at the ICSI , University of California in Berkeley). Additionally, NS2 has always included substantial contributions from other researchers.

2.2.1 NS2: Overall description

NS2 is a discrete event simulator for networks. It uses the OTcl programming language to define the simulation scenario. The core of the simulator and most of the network protocol models are written in C++, and the rest is in OTcl. As source code is available, there are many external modules for NS2 that provide models that NS2 itself does not yet provide, and allows the development of models that are experimental, and are not yet supported by industry.

The design of the simulator separates the "data" from the control by using: (i) C++ for "data" (per packet processing, core of ns, fast to run, detailed, complete control); and (ii) OTcl for control (simulation scenario configurations, periodic or triggered action, manipulating existing C++ objects, fast to write and change). In practice, some of the protocol models are in OTcl and there is not such a clean split between C++ and OTcl.

2.2.2 NS2: Modelling capability

NS2 focuses on modeling various network protocols for wired, wireless and satellite networks, including TCP, UDP, SCTP, supporting multicast and unicast, related with Web, telnet, FTP, CBR etc. It also provides a reach infrastructure for simulation related with statistics, tracing, error models, etc.

NS2 functionalities comprise routing, transportation (TCP and UDP), traffic sources (web, ftp, telnet), queuing disciplines, QoS (IntServ and Diffserv) and emulation for wired systems. For the wireless part, NS2 provides ad hoc routing and mobile IP. It also provides support for traffic traces and simple trace analysis.

- **R1: Models of VoIP protocol stacks**

Grade: ++ (satisfactory, although components can come from many contributors and their credibility needs to be checked).

NS2 does not come with any VoIP stack built in, but there are models available as patches to NS2, such as the SIP patch from the National Institute of Standards and Technology (http://www-x.antd.nist.gov/proj/iptel/). VoIP could be simulated with the right parameters to generate traffic that is similar to the real pattern.

R2: Models of MPLS-TE and RSVP signalling protocols

Grade: ++ (satisfactory, although components can come from many contributors and their credibility needs to be checked).

While MPLS and LDP support come with NS2, RSVP-TE and MPLS with hierarchical addressing support are a separate patch available (free) from Technical University of Ilmenau, Germany, see
http://wcms1.rz.tu-ilmenau.de/fakia/RSVP-TE.1573.0.html?&L=0.
.

- **R3: Models of typical failures and different routing protocols.**

Grade: +++ (Very good support).

Ns2 supports the simulation of link and node failure, and provides models (in a partially abstracted form) for different routing protocols.

- **R4: Models of optical layer components**

Grade: + (poor support).

NS2 does not support optical network layer simulation directly. It is assumed that the optical network can be treated as a collection of point to point links. Loss models, MAC protocol, bit rate and propagation delay are configurable to give a close resemblance to various kinds of links.

- **R5: Models of different traffic scenarios.**

Grade: ++ (some support)

NS2 supports some traffic models, such as FTP, telnet, HTTP, and constant bit rates. Real traffic traces can be used after preprocessing. Traffic models can be programmed in C++ or OTcl.

- **R6: Models of Quality of Service (QoS) mechanisms**.

Grade: +++ (good)

NS2 supports IntServ and DiffServ, and can use different queuing models on each node, including DropTail and RED.

- **R7: Models of different network architectures.**

Grade: +++(good \)

Various types of networks are supported by NS2, including Point to point, LAN, Wireless LAN, MANET, and satellite networks. No topology is forced on a simulation.

- **R8: Analysis of typical performance measures**

Grade: ++ (satisfactory)

Processes can be set up in OTcl so that when an event to be measured occurs, the function is called and the measurement carried out. It can be difficult to set up some types of measures but the use of OTcl as a general purpose programming language allows any calculatable function to be performed.

- **R9: Input/output data in XML format**

Grade: + (possible; but could be a time consuming exercise)

NS2 uses OTcl files to set up and run simulations. These OTcl files could be used to parse and run simulations specified in XML files. The output could also be formated into XML. It could take a lot of development to create such an OTcl program to do so.

2.2.1.1. NS2: Credibility of simulation models

Many of the protocol models in NS2 are abstracted to cover many similar protocols in one implementation. Examples of this include the routing protocols DV and LS, which are similar to RIP and OSPF, respectfully.

If the credibility of a protocol model is questioned, the source code can be examined to determine how closely it relates to the real protocol, and changes made if necessary.

- **R10: Credibility of simulation models**

Grade: ++ (satisfactory, although components can come from many contributors and their credibility needs to be checked).

While no commercial developer has taken responsibility for the credibility of NS, the number of people who use it, the on-going development, and the vast verification suite do

add to the credibility of the NS2 simulation models. However, one needs to be careful when using external modules and patches.

2.2.4 NS2: Credibility of simulation results

- **R11: Quality of sources of randomness.**

Grade: +++ (good, with patch)

NS2 supports pluggable PRNG. The default is not satisfactory for long simulations, but with the Akaroa2 patch, uses the Akaroa2 PRNG which is very suitable.

- **R12: Sequential (on-line or off-line) analysis of simulation output data**

Grade: +++ (good)

While NS2 itself does not support sequential on-line analysis of output data, a patch exists for NS2 that lets the Akaroa2 simulation controller to use NS2 in Multiple Repetitions in Parallel (MRIP), to establish results with low statistical error and in distributed (fast) mode. Also, the OTcl scripting language and command line driven design allow scripting to be used to run many simulations automatically.

With programming, the OTcl simulation scripts could do on-line analysis of output data, and run simulations again if needed. This could also be archived using shell scripting to run the simulation OTcl file.

- **R13: Possible cooperation with Akaroa2, an automated sequential controller of simulation in MRIP scenario**

Grade: +++ (fully supported, with patch)

The ns2-akaroa patch provides full Akaroa2 support to NS2. It is available from Simulation Research Group at the University of Canterbury in Christchurch, at www.cosc.canterbury.ac.nz/research/RG/net_sim/simulation_group/akaroa, or the University Technical University of Berlin in Germany, at www-tkn.ee.tu-berlin.de/research/ns-2_akaroa-2/ns.html

2.2.5 NS2: Extendibility

- **R14: Ability of extending/adding new simulation models of protocols and mechanisms in a reasonable time**

Grade: +++ (full support)

With the source code available, there are a number of existing extensions to NS2 and it is fairly easy to add new extensions. It is possible to write them in either OTcl or C++. There can sometimes be difficultly with merging a large number of patches, but that is fairly minor and one-off concern.

2.2.6 NS2: Usability

- **R15 : Existence of GUI, which can be used as input/output interface.**

Grade: + (little support)

NS2 simulations are set up using the OTcl programming language. OTcl is an object orientated extension to the Tcl programming language. Since it is a full programming language, there is great flexibility in what and how set up and processing is done. OTcl is hardly used outside of NS2, and Tcl on which it is based is not as well known as it was in the past.

There is a "Network Builder" that can be used to graphically set up simulations, but does not utilize the additional modules that have been added nor it uses the extra flexibility of the OTcl language.

For output, there is NAM – the Network Animator – that visualizes the trace of packets through the network and where packets are lost. It is helpful for finding out what happened and maybe finding the cause of an event, but does not provide any help for producing statistically accurate results. Xgraph is also provided to graph output from a simulation on the screen.

- **R16: Existence of a good manual and an introductory tutorial, explaining how a given simulator can be used.**

Grade: +++ (good).

There is the NS Manual, which outlines most of the features and how to use them in. There is also a simple tutorial for starting to use NS2. Both are on the NS website, and many other pages on how to use NS2 are available online. There is a large community of users who help each other on the ns-users email list, where most sensible queries are answered quickly.

2.2.7 NS2: Cost of lincenses

- **R17: The cost of a commercial license of a given simulator**.

- **Grade**: +++ (free license for NS2, plus the cost of commercial License for Akaroa2 if error control is requitred)

NS2 is released under a collection of licenses that are GPL compatible, thus NS2 is Free Software and available for free download from SourceForge and the nsnam project homepage at http://www.isi.edu/nsnam/ns/ . The RSVP-TE implementation is GPL licensed and available free on request. Development is continuing on NS2, and NS3 is in early planning stages.

Full application of NS2 in the NGN research project would mean execution of on-line sequential analysis of output data during simulation. This would require linking NS2 with Akaroa2, an automated controller of discrete-event simulation. A single license of Akaroa2 for commercial costs a few thousand dollars.

2.2.8 NS2: Strengths and Weaknesses

Main strengths of NS2:

- Powerful and flexible scripting and simulation setup
- Well known and widely used (big user group)
- Many protocols implemented
- Ongoing development
- Source code available, freely (but some extensions, like Akaroa2, can require licenses)
- Easy to extend

Main weaknesses of NS2:

- Some protocols and features not well documented
- No well known commercial/technical support
- Patching extensions in is not easy
- No clean separation between C++ and OTcl
- Currently, poor support of optical layer simulation

2.3 OMNET++/OMNEST (Version 3.2)

2.3.1 Overall Description

OMNET++ stands for Objective Modular Network Testbed in C++, which is an

open-source, component-based simulation package, built on C++ foundations. OMNEST is the commercial version name for OMNET++. It offers a C++ simulation class library and GUI support (graphical network editing, animation). The principal author is András Varga from the Technical University of Budapest, with occasional contributions from a number of other people.

The simulator can be used for: traffic modeling of telecommunication networks; modeling of protocols, queuing networks, multiprocessors and other distributed hardware systems; validating hardware architectures; evaluation of performance aspects of complex software systems; in general: modeling of any system that can be mapped to active components communicating by passing messages. In some sense, and differently from the other simulators discussed so far, OMNET++ is not specifically designed for telecommunication networks. In addition, OMNET++ has been carefully designed from the software point of view, resulting in a product which is well organized, flexible and easy to use. The main components of OMNET++ are:

- Simulation kernel library
- Compiler for the NED topology description language (NEDC)
- Graphical network editor for NED files (GNED)
- GUI for simulation execution, with links into simulation executable (Tkenv)
- Command-line user interface for simulation execution (Cmdenv)
- Graphical output vector plotting tool (Plove)
- Utilities (random number seed generation tool, makefile creation tool, etc.)
- Documentation, sample simulations, contributed material, etc.

An OMNET++ model of a system (e.g., a network) consists of hierarchically nested modules. The depth of module nesting is not limited, allowing the user to reflect the logical structure of the actual system in the model structure. Modules communicate via message passing. Messages can contain arbitrarily complex data structures. Modules can send messages either directly to their destination or along a predefined path, through gates and connections, which have assigned properties like bandwidth, delay, and error rate. Modules can have parameters which are used to customize module behavior, to create flexible model topologies, and for module communication, as shared variables. Modules at the lowest level of the module hierarchy are to be provided by the user, and they contain the algorithms in the model. During simulation execution, simple modules appear to run in parallel, since they are implemented as coroutines (sometimes called “lightweight processes”). To write simple modules, the user is requested to use C++ programming. OMNET++ has a consistent object-oriented design. One can freely use concepts of object-oriented programming (inheritance, polymorphism etc.) to extend the functionality of the simulator. OMNET++ simulations can feature different user interfaces for different purposes: debugging, demonstration and batch execution.

2.3.2 OMNET++: Modelling capability

- **R1: Models of VoIP protocol stacks.**

Grade: ++ (satisfactory)

Currently, OMNET++ model library only includes SIP, RTP, RTCP models, but it is possible to extend previous models and create new protocol models by custom coding.

- **R2: Support of MPLS and RSVP-TE signaling protocol**

Grade: + (poor; but the situation can change)

In OMNET++, the INET Framework, a simulation model suite for TCP/IP and Internet-related protocols, written for the OMNeT++/OMNEST simulation environment contains models for IP, TCP, UDP, PPP, Ethernet, MPLS with LDP and RSVP-TE signalling, and other protocols. The MPLS model in OMNeT++ was originally developed by Xuan Thang Nguyen from the University of Technology in Sydney. Now the model is maintained by András Varga. The model includes components for MPLS forwarding, LDP, CR-LDP and RSVP-TE.

The MPLS model in INET Framework was developed to study the forwarding mechanisms in MPLS, so some simplification done in the way that the Label Switched Routers was implemented. This made this model as it is today not suitable for failure recovery simulation. The forwarding tables for all routers in the network is implemented in one table, which means that all routers have the same copy of the network topology at the same time. If a failure happens in the network then all routers will be aware of the failure at the same time. This is not what happens in a real case scenario where topology changes like a failure has to be indicated to nodes by sending routing update information from the point of failure. In addition, the current MPLS and RSVP models are unstable and incomplete. The developer promised that it will be replaced by a new implementation, but this work is still in progress and the authors do not mention when it will be finished.

- **R3: Support of failure modeling and different routing protocols**

Grade: ++ (satisfactory)

OMNET++ supports modelling of node/link failure and recovery. Entities can be set to fail/recover at a specified time. Routing protocol models which are provided include Static Routing, AODV, BGP, EIGRP, ISIS, IGMP, OSPF.

- **R4: Models of optical layer components.**

Grade: ++ (partially supported)

OMNET++ based Dynamic Lightpath Establishment (ODLE) simulator is designed to allow parametric studies of various dynamic lightpath establishment protocols and different RWA schemes in optical layer can be studied.

- **R5: Models of different teletraffic scenarios.**

Grade: ++ (partially supported)

Some traffic models supported including VoIP, CBR, FTP, HTTP, and VBR. In addition, any custom applications can be developed in OMNET ++.

- **R6: Models of Quality of Service (QoS) mechanisms.**

Grade: ++ (partially supported)

There is a partially implementation of QoS mechanisms in OMNET ++. They include:

- QoS Protocols: Diffserv Traffic Classifier, Per Hop Behaviors,
- support for drop tail router queues, and queues with QoS support based on DS code point, and
- queueing & scheduling models such as RED.

- **R7: Models of different network architectures.**

Grade: +++ (well supported)

It is possible to create any form of hierarchical topology by using a human-readable and expressive textual topology description format (the NED language). The same format is used by a graphical editor (GNED).

- **R8: Analysis of typical performance measures.**

Grade: + (poor support)

Reporting of simulation results is not well developed and many useful metrics which include all time series measurements need to be custom coded.

- **R9: Input/output data in XML format**

Grade: +++ **(**supported**)**

OMNET provides an XML scenario/topology/model import function. It can also save scenarios in XML format.

2.3.3 OMNET++: Credibility of simulation models

- **R10: Credibility of simulation models**

Grade: +++ (well supported)

OMNSET is a commercial simulator, so they take full responsibility for credibility of simulation models. Some models are contributed by its user community, which are not verified and maintained by OMNSET.

2.3.4 OMNET++: Credibility of simulation results

- **R11: Quality of sources of randomness**

Grade: +++

OMNET++/OMNSET offers a number of PRNGs, including such a powerful PRNG as Mersenne Twister, with provably good probabilistic features and the cycle of over 2^{1900} long. Using this generator, there is no danger that its randomness is repeated. The source code of random generator is open.

- **R12: Sequential, (on-line or off-line) analysis of simulation output data.**

Grade: +++ (Supported)

OMNET++ both supports sequential simulation and parallel simulation through the use of either MPI or PVM3 communication libraries. Moreover, OMNET++ can offer sequential on-line analysis of simulation output data by being controlled by Akaroa2. It includes the same synchronization methods used in PARSEC.

- **R13: Possible cooperation with Akaroa2, an automated sequential controller of simulation in MRIP scenario**

Grade: +++ (fully supported)

OMNET++ opens the source code of random generator and it can be a linked with Akaroa2. The ns2/omnet++ patch is available from Simulation Research Group at the University of Canterbury in Christchurch, at

www.cosc.canterbury.ac.nz/research/RG/net_sim/simulation_group/akaroa,

or from the University Technical University of Berlin in Germany, at

www-tkn.ee.tu-berlin.de/research/akaroa-omnetpp/index.html

2.3.5 OMNET++: Extendibility

- **R14: Ability of extending/adding new models of protocols and mechanisms in a reasonable time**

Grade: +++ (full support)

OMNET++ is inherently extendible because it runs on top of a C-compiler. Its all models are available as source code, and can form a basis for further model development.

2.3.6 OMNET++: Usability

- **R15: Existence of GUI, which can be used as input/output interface.**

Grade: +++ (supported)

The OMNET++ user interface can be used during simulation running. The OMNET++'s design allows to see internals of the model. It also allows the user to initiate and terminate simulations, as well as to change variable inside simulation models. These features are handled during the development and debugging phase of modules in a project. Graphical interface is a user friendly option in OMNET++ and allows access to the internals of the model. The interaction of the user interface and the simulation kernel is through well defined interfaces. Currently, two user interfaces are supported:

- Tkenv: Tk-based graphical, windowing user interface (X-Window, Win95, WinNT etc..) , and
- Cmdenv: command-line user interface for batch execution.

Simulation is tested and debugged under Tkenv, while the Cmdenv is used for actual simulation experiments since it supports batch execution.

- **R16: Existence of a good manual and an introductory tutorial, explaining how a given simulator can be used.**

Grade: ++ (satisfactory)

OMNET++ provides a manual and an introductory tutorial allowing users to learn how to use that simulator in a short time. The manuals are still evolving and have not fully reached the level of detail which is desirable.

2.3.7 OMNET++: Cost of licences

- **R17: The cost of a commercial license**

- **Grade: ?** (license for OMNSET, plus a lincese for Akaroa2, if error control is needed).

OMNeT++ is free for any non-profit use. The OMNSET distributors would be contacted if this simulator were used in a commercial project.
Full application of OMNET++ in the NGN research project would mean execution of on-line sequential analysis of output data during simulation. This would require linking OMNET++ with Akaroa2, an automated controller of discrete-event simulation. A single license of Akaroa2 for commercial costs a few thousand dollars.

2.3.8 OMNET++: Strengths and Weaknesses

Strengths:

- Professional technical support
- Growing customer base
- Hardware requirements are moderate
- Relatively easy to learn

Weaknesses:

- Documentation is not adequate at this stage
- Model design GUI is not detailed enough to be useful
- Simulation results reporting is not adequate.

2.4 GLASS/SSFNet

2.4.1 Overall Description

GMPLS (Lightwave Agile Switching Simulator), shortly GLASS, is a Java-based network simulation tool that facilitates the evaluation of routing, restoration and signaling protocols in an optical environment and allows network planners and researcher to study behavior of algorithms and protocols without the need for building a real implementation. The software originated from cooperation of the High Speed Network Technologies Group and the Internetworking Technologies Group of the Advance Network Technologies Division at NIST. The GLASS simulator represents the next generation of the previous MERLiN Tool that was developed in 1998 at NIST. The basic framework is the Scalable Simulation Framework (SSF) with its extension SSFNet. GLASS is designed on top of SSF/SSFNet and takes advantage of the many capabilities provided by SSFNet, such as the handling of discrete events, the ability to run on multiprocessors, and the scalability to large number of nodes. In addition, SSFNet provides network components such as hosts, routers, links, and a number of network protocols. GLASS extends these components with an implementation of MPLS, optical components such as optical cross connects (OXC), edge routers, optical links, fibers, and lambdas. GLASS uses the Data Modeling Language (DML) to design the topology and derive scripts for the simulation scenarios. DML also allows a very high level description of the components and the configuration topology.

GLASS provides access to entire library of the protocols available in SSF. These protocols run in the non-optical domain as well. On the optical side, GLASS provides access to models of GMPLS, static OSPF over optical networks, IP over optical networks, diverse example implementations of failure propagation and recovery protocols that are not IP based (pure optical signaling). Basically GLASS is a command line simulator that reads its simulation setup out of a script file and offers the possibility to write dumps into binary files. These files that can be post processed by customized readers.

2.4.2 GLASS: Modelling capability

- **R1: Models of VoIP protocol stacks.**

Grade: - (not supported)

So far, GLASS model library does not include any VoIP model.

- **R2: Models of MPLS and RSVP-TE signaling protocols**

Grade: +++ (supported)

Although the name implies a simulator that is developed for GMPLS simulations, it can be used for MPLS simulations as well. The simulator has implemented MPLS forwarding and label distribution with LDP, CR-LDP and RSVP-TE.

- **R3: Models of typical failures and different routing protocols**

Grade: ++ (partially supported)

By default the GLASS framework provides failure and recovery events. GLASS allows protocols designers to use their own customized failure detection mechanisms. This has to be done by replacing the default modules for optical failure detection and non-optical failure detection. These modules are located in the network interface card (NIC) or the ONIC. Each card or connector has its own detection module. This allows users to run different detection strategies per node. The configuration of the failure detection modules will be done in the simulation DML file. The simulator provides the following failures:

- Node Failure
- NIC / ONIC Failure
- Link / Optical Link Failure
- Fiber Failure
- Lambda Failure

Only some routing protocol models are provided includes CSPF, BGP in GLASS.

- **R4: Models of optical layer components.**

Grade: +++ (fully supported)

GLASS has a modular architecture consisting of several components, namely, the optical physical layer, the logical layer, and protocols such as routing, restoration, and wavelength assignment algorithms.

- **R5: Models of different teletraffic scenarios.**

Grade: +++ **(**supported)

Some traffic models supported including CBR, FTP, HTTP and VBR. In addition any custom applications can be developed.

- **R6: Models of Quality of Service (QoS) mechanisms.**

Grade: ++ (partially supported)

There is a partial implementation of QoS mechanisms in GLASS. These are:

QoS Protocols: Diffserv Traffic Classifier, Per Hop Behaviors; and Queueing and Scheduling Models: CBQ, RED Round Robin.

- **R7: Models of different network architectures.**

Grade: +++ (fully supported)

GLASS supports different kinds of network architecture.

- **R8: Analysis of typical performance measures.**

Grade: ++ (partially supported)

Reporting of simulation results is not well developed and many useful metrics which include all time series measurements need to be custom coded.

- **R9: Input/output data in XML format**

Grade: +++ (supported)

GLASS provides an XML scenario/topology/model import function. It can also save scenarios in XML format.

2.4.3 GLASS: Credibility of simulation models

- **R10: Credibility of simulation models**

Grade: - (no responsibility; developed on voluntary basis)

The validity of the model library has not been yet fully verified, as there is not too many publications reporting use of GLASS in research projects. Models are being developed on voluntary basis and there are not too many users to make sure that possible bugs have been detected.

2.4.4 GLASS: Credibility of simulation results

- **R11: Quality of sources of randomness**

Grade: ?

No information on pseudo-random number generators can be found in GLASS manual.

- **R12: Sequential (on-line or off-line) analysis of simulation output data.**

Grade: ?

No information on this issue can be found in GLASS manual. It is probably off-line, non-sequential data analysis.

- **R13: Possible cooperation with Akaroa2, an automated sequential controller of simulation in MRIP scenario**

Grade: ?

No information on this issue can be found in GLASS manual.

2.4.5 GLASS: Extendibility

- **R14: Ability of extending/adding new simulation models of protocols and mechanisms in a reasonable time**

Grade: +++ (supported)

GLASS is inherently extendible because of its Java based structure and SSFNet platform. All provided models are available as source code, and can thus form a basis for further model development.

2.4.6 GLASS: Usability

- **R15: Existence of GUI, which can be used as input/output interface.**

Grade: +++ (supported)

There are two visualization tools provided in GLASS: Browser and Topology and Simulation Creator (TSC). The browser is a viewer on top of the simulation framework. It helps examine the actual state information of the physical and the logical topology. The browser allows running the simulation and examining the values of each component after the simulation ends.
The GLASS-TSC allows the user to create a simulation by using a graphical user interface. Here the user can script failure and recovery events as well as configure the nodes, links, protocol stack, and predefine connections. The TSC also allows an algorithm developer to test his/her protocol without running whole simulation. In this case, a user has to create the topology and specify connections between nodes. The TSC also allows interactive simulations to improve debug capabilities.

- **R16: Existence of a good manual and an introductory tutorial, explaining how a given simulator can be used.**

Grade: + (at this stage: poorly supported)

GLASS provides a manual and an introductory tutorial allowing users to learn how to use that simulator in a short time. The manuals are still evolving and have not reached the level of detail which is desirable. Technical questions are typically answered within a week.

2.4.7 GLASS: Cost of licences

- **R17: The cost of a commercial license of a given simulator.**

Grade: +++ (freeware)

GLASS and SSFNet are available free of charge.

2.4.8 GLASS: Strengths and Weaknesses

Strengths.

- Comprehensive GMPLS modelling
- Specially designed for studying IP/MPLS/Optical network and multilayer resiliency
- Growing customer base
- GLASS is Java based and linked with scalable platform of SSFNet

Weaknesses.

- Documentation is currently not adequate
- A relative new/not widely used simulation tool, so the validity of modules has not been yet verified
- Smaller library of pre-built modules and protocols than in other simulators
- Simulation result reporting is not adequate
- Poor post-technical support

2.5 QualNet

2.5.1 Overall Description

QualNet network simulation software has been developed and marketed by Scalable Network Technologies. It provides a comprehensive set of tools with many components for custom network modeling and simulation. Models in source code form provide developers with a solid foundation from which to build new functionality or to modify existing functionalities. QualNet does have a range of wired as well as wireless models but its main strength is in the wireless area.

A user can decide to use the GUI which is a Java based front end. This allows one to construct topologies using building block like network components which can be dragged-and-dropped. The GUI has the following functions.

- Scenario Designer: This is a graphical experiment setup tool. Define geographical distribution, physical connections and the functional parameters of the network nodes, all using intuitive click and drag tools. Define network layer protocols and traffic characteristics down to each node. .
- Animator: This is a graphical simulation visualization tool. As simulations are running, watch traffic flow through the network and create dynamic graphs of critical performance metrics. You can also assign jobs to run in batch mode on a faster server, viewing the animated data later
- Protocol Designer: This is a Finite State Machine tool for custom protocol modeling. Shorten your development time by using an intuitive state-based visual tool to define the events and processes of your protocol model. You can modify ready-made protocol models or generate code from scratch for your own custom protocols and special statistical reporting. Protocol Designer generates API calls automatically to speed up code development as well as simplify integration to existing QualNet models.
- Analyzer: This is a statistical graphing tool. Choose from hundreds of metrics in pre-designed reports, or use Designer to customize your own reports. You can view statistics as they are being generated, as well as compare results from different experiments. All graphs are exportable to spreadsheets.
- Packet Tracer: This is a packet-level visualization tool for viewing the contents of a packet as they go up and down the protocol stack.

One can also build a configuration script which can be used on the command line as a parameter to the command line version.

A network entity in QualNet is made up of 5 layers. These are:

- Application
- Transport
- Network
- MAC
- Physical

Each layer is coded as one or many C source files with header files. Any of the layers may be modified or replaced with custom layers as long as proper interfacing is maintained between the layer above and below. Most of the source code is provided under the basic license option.

2.5.2 QualNet: Modelling capability

- **R1: Models of VoIP protocol stacks.**

Grade: ++ (partially supported)

QualNet model library includes SIP, H225, H232, RTP, RTCP models. There are no user contributed models.

- **R2: Models of MPLS and RSVP-TE signaling protocols**

Grade: +++ (supported)

QualNet offers a fairly detailed MPLS model which allows the user to configure dynamic/static LSPs. The label distribution protocols supported are LDP and RSVP-TE.

- **R3: Models of typical failures and different routing protocols**

Grade: +++ (well supported)

QualNet supports modelling of node/card/link failures and recovery. Entities can be set to fail/recover at a specified time or randomly during simulation.

Many different routing protocol models are provided which include Distributed Bellman Ford, ICMP, RIPv1, RIPv2, Static Routing, AODV, BGP, EIGRP, HSRP, IGMP, IGRP, Mobile IPv4, OSPF, Policy Based Routing, Router Access Lists and Route Maps.

- **R4: Models of optical layer components.**

Grade: -

Not supported in QualNet. Only packet level switching is currently available.

- **R5: Models of different teletraffic scenarios.**

Grade: +++ (fully supported)

Many traffic models supported including VoIP, CBR, FTP, FTP Generic, HTTP, Lookup, SuperApp, tcplib, Traffic Gen, Traffic Trace, VBR and MCBR. In addition, any custom applications can be developed. The GUI based support in this sense is very limited and FSM builder is not fully developed yet. Even so applications can be manually coded using C.

- **R6: Models of Quality of Service (QoS) mechanisms.**

Grade: +++ (supported)

There is a full implementation of QoS mechanisms in QualNet. These are:

QoS Protocols: Diffserv Traffic Classifier, Per Hop Behaviors; Transport Protocols: ECN Queueing & Scheduling Models: CBQ, RED, RIO, Round Robin, SCFQ, WFQ, WRED, WRR; Routing Protocols: QOSPF; etc.

- **R7: Models of different network architectures.**

Grade: +++ (supported)

QualNet supports wired, wireless, centralized and ad-hoc networks.

- **R8: Analysis of typical performance measures.**

Grade: + (weakly supported at this stage)

Reporting of simulation results is not well developed and many useful metrics, which include all time series measurements, need to be custom coded. Some, as those related with TCP, can be imported by using a free third party software.

- **R9: Input/output data in XML format**

Grade: +++ (supported)

QualNet provides an XML scenario/topology/model import function. It can also save scenarios in XML format but this format is not the same as that used in OPNET.

2.5.3 QualNet: Credibility of simulation models

- **R10: Credibility of simulation models**

Grade: +++ (full responsibility)

QualNet is a commercial simulation tool, they have full responsibility of credibility of simulation models.

2.5.4 QualNet: Credibility of simulation results

- **R11: Quality of sources of randomness**

Grade: ? (unknown)

QualNet is a commercial simulation tool; while they have a responsibility regarding of providing high quality generators of randomness, no information about generators they use can be found in documentation about Qualnet available on-line. Further inquiries would be need.

- **R12: Support of sequential, on-line/off-line analysis of simulation output data.**

Grade: ? (further inquiries would be need)

The most likely, it does not support on-line analysis of simulations.

- **R13: Possible cooperation with Akaroa2, an automated sequential controller of simulation in MRIP scenario**

Grade: ++ (the most likely)

The most likely, it can be done as the source code, which is available, runs on top of a C-compiler,

2.5.5 QualNet: Extendibility

- **R14: Ability of extending/adding new simulation models of protocols and mechanisms in a reasonable time**

Grade: +++ **(**supported)

QualNet is inherently extendible because it runs on top of a C-compiler. All provided models are available as source code, and can thus form a basis for further model development. Development will be slower than OPNET due to the poor GUI support but the architecture of the simulator makes it easy to understand and modify.

2.5.6 QualNet: Usability

- **R15: Existence of GUI, which can be used as input/output interface.**

Grade: ++ (partial functionality only)

The GUI allows a user to build a scenario/topology very quickly and simulate it. Once a model which is built (preferably manually) has been integrated into the GUI it can be used just like any of the standard models provided. The output of the GUI is not very useful since no time series data is given. Some statistics display formats are hard to understand.

- **R16: Existence of a good manual and an introductory tutorial, explaining how a given simulator can be used.**

Grade: ++ (partially supported)

QualNet provides a manual and an introductory tutorial allowing users to learn how to use that simulator in a short time. The manuals are still evolving and have not reached the level of detail which is desirable. Response time to technical questions is within 2 working days for commercial customers. Telephone technical support is also available.

2.5.7 QualNet: Cost of licences

- **R17: The cost of a commercial license of a given simulator.**

Grade: + (relatively expensive commercial license is needed)

According to an on-line document, the basic package (Sequential QualNet Developer and Runtime Sequential QualNet Developer**, four libraries**: VoIP, MANAT, QoS and IPv6, and one year email support) would cost about US$55 000, ie. slightly below NZ$100000.

2.5.8 QualNet: Strengths and Weaknesses

Strengths:

- Relatively easy to learn
- Professional technical support
- Growing customer base
- Hardware requirements are moderate

Weaknesses.

- Documentation is not adequate yet
- Model design GUI is not detailed enough to be useful
- Not as widely used as NS2
- Simulation result reporting is not adequate, but can be custom built quite easily.

2.6 J-Sim (Version 1.3)

2.6.1 Overall Description

J-Sim (formerly called JavaSim) has been developed by a team at the Distributed Realtime Computing Laboratory (DRCL) of the Ohio State University. Additional third-party packages are also available. J-Sim is a component-based compositional network simulation environment and has been developed entirely in Java. The basic units of J-Sim are autonomous components so that they can be plugged into a software system, even during execution. On top of this autonomous component architecture, J-Sim has a generalized network model that defines the generic network components. These components can be used as the base classes to accommodate or implement new protocols or algorithms across various layers.

With the general network model, J-Sim is able to accommodate other network architectures, such as IETF DiffServ architecture, mobile ad hoc environment and the WDM-based optical network architecture. In addition, J-Sim supports script languages such as Perl, Tcl or Python to configure components at runtime and also it is a dual-language environment to integrate components using Tcl/Java. That facilitates simple and fast prototyping of simulation scenarios and diagnosis. J-Sim also has a GUI and can be installed in Windows or UNIX and the system requirements for running J-Sim is that a Java Virtual Machine (JVM) and Sun's JDK version 1.4 or later is used. There are a number of features of J-Sim, which are described in the following subsections:

- **Loosely coupled, autonomous component programming model:** J-Sim is built on loosely coupled components architecture. Components have the capability to handle data in independent execution context, which enables them to be designed, implemented and tested separate from the rest of the system. Also, since they are loosely coupled or independent, components can be reused in other systems.
- **Dynamic thread execution framework for real-time process-driven simulation**: In J-Sim, execution is implemented by Java threads. The thread scheduler is the Java Virtual Machine (JVM), which schedules the thread execution. With this mechanism, events are executed and simulation runs at real time (parallel simulation engine), thus preserves the real systems behavior and enhances the fidelity of the simulation
- **Implementation of a suite of Internet Integrated/ Differentiated/ Best Effort Services protocols**: J-Sim supports other network architectures such as DiffServ architecture, mobile ad hoc environment and WDM-based optical network architecture.
- **A dual-language environment that allows auto-configuration and on-line monitoring**: J-Sim provides a dual-language environment that creates components. A script language is used to integrate components at run time and to provide dynamic control. This facilitates fast prototyping of customized simulation scenarios, on-line monitoring and data collection.

2.6.2 J-Sim: Modelling capability

- **R1: Models of VoIP protocol stacks.**

Grade: - (not supported)
So far, J-Sim model library does not include any VoIP models.

- **R2: Support of MPLS and RSVP-TE signaling protocol**

Grade: + (weakly supported)

The MPLS model provided by J-Sim is not flexible. Indeed, the user can associate only statically an outgoing interface and an outgoing label with an incoming label and incoming interface. This is not really satisfactory if users plan to study RSVP-TE protocols and dynamic Label Switched Path (with stack of labels and associated operations, etc.).
The label distribution protocol LDP or RSVP-TE is not supported in J-Sim.

- **R3: Models of typical failures and different routing protocols**

Grade: ++ (partially supported)

J-Sim supports commands to enable/disable/display the 'component' flag, in simulation, this feature is useful to set a failure. For example, disable a link to simulate a link failure,

disable a node to simulate a node failure or disable a protocol component to simulate a protocol failure.

Only some routing protocol models are provided which include RIP, OSPFv2, CBT, DVMRP and Multicast shortest path tree in J-Sim.

- **R4: Models of optical layer components.**

Grade: ++ (partially supported)

J-Sim can serve as a base model for other network architectures, e.g., WDM-based optical network architecture by custom coding.

- **R5: Models of different teletraffic scenarios.**

Grade: + (weak support)

Only some traffic models supported, including CBR, FTP, FSP and WWW.

- **R6: Models of Quality of Service (QoS) mechanisms.**

Grade: ++ (partially supported)

J-Sim provides DiffServ architecture (such as marker at edge routes and buffer management at core routers). There are two major components: (1) the tagging/policing mechanisms at the edge routers that appropriately mark or police packets before they enter the network; and (2) the active queue management mechanisms at the core routers that buffer and forward packets and realize the per-hop behavior.

- **R7: Models of different network architectures.**

Grade: +++ (supported)

J-Sim supports hierarchical networks of arbitrary depth. Specifically, an internetwork consists of networks, nodes and links. A network, in turn, contains nodes, links and networks of smaller sizes. The internetworking system itself is a network entity, and the ancestor of all entities contained in it.

- **R8: Analysis of typical performance measures.**

Grade: + (weakly supported)

Reporting of simulation results is not well developed and many useful metrics which include all time series measurements need to be custom coded.

- **R9: Input/output data in XML format**

Grade: +++ (supported)

J-Sim provides an XML scenario/topology/model import function. It can also save scenarios in XML format.

2.6.3 J-Sim: Credibility of simulation models

- **R10: Credibility of simulation models**

Grade: - (nobody is responsible for possible bugs; the validity of models has not been tested yet, as there is a limited use of JSim in research projects**)**

All work is done voluntarily, so nobody is responsible for possible bugs. So far, only a limited number of publications based on simulation results from J-Sim, mostly written by developers, have been published. Thus, the validity of these modules has not been fully confirmed yet.

- **R11: Quality of sources of randomness**

Grade: + (adequate for not-too-large simulations)

No information on pseudo-random number generators can be found in JSim manual.

However, the most likely it uses a PRNG from Java, with the cycle length of 2^{48}. Thus, in larger and stochastically dynamic simulations on fast processors, when larger number of events needs to be simulated, or when dealing with strongly correlated stochastic processes, or when larger samples of output data are needed for producing accurate estimates, this generator can have a too short cycle for avoiding repetition of the randomness. However, this PRNG should be of adequate quality for smaller to medium scale simulations of not-too-strongly correlated processes.

- **R12: Sequential (on-line or off-line) analysis of simulation output data.**

Grade: ? (probably off-line, non-sequential data analysis only)

No information on this issue can be found in J-Sim manual. It is probably off-line, non-sequential data analysis.

- **R13: Possible cooperation with Akaroa2, an automated sequential controller of simulation in MRIP scenario**

Grade: ? (further investigation needeed)

Probably yes, but to fully answer this question, one needs to discuss it with technical staff of JSim.

2.6.4 J-Sim: Extendibility

- **R14: Ability of extending/adding new simulation models of protocols and**

mechanisms in a reasonable time

Grade: +++ (supported)

It is possible to extend J-Sim to new network architecture by subclassing appropriate network modules and redefining their network attributes and methods that manipulate the attributes.

2.6.5 J-Sim: Usability

- **R15: Existence of GUI, which can be used as input/output interface.**

Grade: +++ (a limited support)

J-Sim is equipped with a Swing-based GUI (the J-Sim Graphical Editor). The editor enables building topology in a graphical manner rather than "manually," in Tcl. The configuration data are stored in XML documents. Additionally, equivalent Tcl script for setting up the topology may be generated. Unfortunately, the editor cannot import Tcl scripts; thus, in practice, it is helpful in building and modifying the topology only (and the XML document must be present from the beginning). This limits its functionality.

- **R16: Existence of a good manual and an introductory tutorial, explaining how a given simulator can be used.**

Grade: + **(**weak support)

J-Sim is free available simulator with downloadable source code, examples, tutorials and white papers. The documents are still evolving and have not reached the level of detail which is desirable. Technical questions are answered with a week delay.

2.6.6 J-Sim: Cost of licences

- **R17: The cost of a commercial license of a given simulator.**

Grade: +++ (freeware)

J-Sim is an open-source software, so there is no costs involved for using it. However, all work is done voluntarily, so nobody is responsible for possible bugs.

2.6.7 J-Sim: Strengths and Weaknesses

Strengths.

- Java based and scalable platform
- Open-source software

Weaknesses.

- Documentation is not adequate yet
- Far less pre-built modules and protocols than in other existing simulators
- Simulation result reporting is not adequate
- Poor GUI and post-technical support

2.7 TOTEM (Version 2.0)

2.7.1 Overall Description

The TOTEM project (Toolbox for Traffic Engineering Methods) is funded by the Direction Générale des Technologies of the Walloon, Belgium. It is a three year project, started in November 2003. The objective of the TOTEM project is to develop a toolbox of algorithms for traffic engineering purposes. The simulator basically takes into account the distribution of the traffic, the fault-tolerance requirement and the support of the quality of service. The simulator develops generic algorithms for the optimization of networks of big size which will apply, on one hand, to IP networks, and on the other hand, to networks operated with (G)MPLS.

TOTEM can be used to simulate the traffic routed on a network using SPF, CSPF or other TE routing algorithms. TOTEM can simulate "what-if" scenarios to help understand the effects of metric changes, failures, traffic changes or BGP policy changes. Some of these algorithms will require extensions to the routing (OSPF-TE, ISIS-TE, BGP) or signalling (RSVP-TE) protocols. This toolbox will be available in open source and will be designed such that its elements can easily be integrated in various platforms such as Linux PCs, routers, and open source network simulators like NS and/or J-Sim. The architecture of TOTEM is characterized as follows:

- **Interoperable network representation**: TOTEM uses an interoperable XML format integrating topology, label switched path and BGP configurations. The traffic matrix is also represented using an XML file. These data can be assembled from a variety of sources such as router's configurations, MRT, Netflow, SRLG..
- **Flexible simulation scenario**: the simulation scenario integrates link utilization computation using SPF, LSP creation with or without backup, link or node failure simulation. All these scenarios can be automatically executed by the toolbox.
- **Topology generation**: the BRITE universal topology generator is integrated for generating state of the art realistic topologies.
- **Traffic matrix deduction**: traffic matrix can be inferred from link load using a simple gravity model or directly from Netflow traces. If no link utilization information is available, it can also generate random traffic matrices following classical distributions.
- **Graphical User Interface**: the toolbox provides a graphical user interface which allows the user to visualize a network topology and the load resulting from a traffic matrix and the CSPF algorithm. The GUI uses the JUNG library.

2.7.2 TOTEM: Modelling capability

- **R1: Models of VoIP protocol stacks.**

Grade: - (not supported)

So far, TOTEM model library does not include any VoIP model.

- **R2: Models of MPLS and RSVP-TE signaling protocols**

Grade: ++ (partially supported)

In TOTEM, the DAMOTE (Decentralized Agent for MPLS Online Traffic Engineering) Model provides two main basic functionalities:

- QoS-based routing of DiffServ LSPs (Label Switched Paths) under constraints.
- Local detour (backup) LSP routing for fast restoration

TOTEM also includes a hybrid IP/MPLS optimization method called SAMTE (Scalable Approach for MPLS Traffic Engineering). The idea of SAMTE is to combine both the simplicity and robustness of IGP routing and the flexibility of MPLS. This approach lies between the pure IP metric-based optimization (as IGP-WO) and the full mesh of LSPs. SAMTE uses the simulated annealing meta-heuristic to find a small number of LSPs (given as parameter) to establish in the network. The combination of the set of LSPs computed by SAMTE and the IGP routing for remaining flows optimize a given operational objective.
So far, the label distribution protocol LDP or RSVP-TE is not supported in TOTEM.

- **R3: Models of typical and different routing protocols**

Grade: ++ (partially supported)

TOTEM only supports modelling of link failure and repair link events. Some basic routing protocol models are provided which include OSPF, ISIS and BGP.

- **R4: Models of optical layer components.**

Grade: - (not supported)

- **R5: Models of different teletraffic scenarios.**

Grade: ++ (partially supported)

Only basic traffic models are supported, including synthetic traffic and gravity model. TOTEM can also generate traffic matrix using NetFlow traces.

- **R6: Models of Quality of Service (QoS) mechanisms.**

Grade: ++ (partially supported)

A simplified DiffServ version is implemented into the toolbox, together with basic DiffServ-aware TE and the MAM (Maximum Allocation Model) bandwidth constraints model.

- **R7: Models of different network architectures.**

Grade: +++ (fully supported)

The BRITE universal topology generator is integrated into TOTEM for generating state of the art realistic network topologies.

- **R8: Analysis of typical performance measures.**

Grade: + (weak support)

So far, only the metric of link load is supported.

- **R9: Input/output data in XML format**

Grade: +++ (supported)

TOTEM provides an XML scenario/topology/model import function. It can also save scenarios in XML format with its own style.

2.7.3 TOTEM: Credibility of simulation models

- **R10: Credibility of simulation models**

Grade: + (limited)

TOTEM is a relative young simulation tool. So far, only a limited number of publications, mostly written by developers, have based on results produced by TOTEM. Thus, the validity of these modules has yet to be thoroughly acknowledged..

- **R11: Quality of sources of randomness**

Grade: ? (the TOTEM technical staff should be asked about the sources of randomness used in TOTEM)

According to TOTEM's designenrs, its elements should be easily integrated in various platforms, such as Linux PCs. Thus, the most likely it uses a PRNG of the platform used by its user. If this is correct, a care needs to be taken for not to use PRNGs with too short cycle, to avoid repletion of randomness when executing larger simulations.

However, this needs to be confirmed with the TOTEM staff. No information on this subject can be found in the on-line technical documentation of TOTEM.

- **R12: Support of sequential, on-line/off-line analysis of simulation output data.**

Grade: ? (most likely, it is not supported)

No related information can be found in the on-line technical documentation of TOTEM.

- **R13: Possible cooperation with Akaroa2, an automated sequential controller of simulation in MRIP scenario**

Grade: ? (further investigation needed)

Probably yes, as its elements should be easily integrated in various platforms, such as Linux PCs. However, to fully answer this question, one needs to discuss it with technical staff of TOTEM.

2.7.4 TOTEM: Extendibility

- **R14: Support of extending existing model or developing new simulation models of protocols and mechanisms in a reasonable time**

Grade: ++ (partially supported)

All provided models are available as source code, and can thus form a basis for further model development, but the current version of TOTEM comes with far less pre-built modules and protocols than the other simulators.

2.7.5 TOTEM: Usability

- **R15: Existence of GUI, which can be used as input/output interface.**

Grade: ++ (partially supported)

The toolbox is together with a Graphical User Interface. The TOTEM graphical interface allows user to load a scenario and take control on its execution process and see the step by step results graphically. However, the current GUI in TOTEM is still poor, only have some simple options e.g., domain loading and unloading, load traffic matrices, adding LSP, the related configurable parameters are limited.

- **R16: Existence of a good manual and an introductory tutorial, explaining how a given simulator can be used.**

Grade: ++ (partially supported)

Currently, TOTEM provides a manual and an introductory tutorial allowing users to learn how to use that simulator in a short time. The manuals are still evolving and have not reached the level of detail which is desirable. Sometimes, there is no reply to technical questions.

2.7.6 TOTEM: Cost of licences

Grade: +++ (freeware)

- **R17: The cost of a commercial license of a given simulator.**

The toolbox is an open-source software, so there are no limitation on how it is used.

2.7.7 TOTEM: Strengths and Weaknesses

Strengths.

- An open-source software
- Specially designed for studying traffic engineering problems in networks

Weaknesses.

- A relative young simulation tool, so the validity of modules has not been yet widely acknowledged
- Documentation is not adequate yet
- Far less pre-built modules and protocols
- Unsophisticated reporting of simulation results
- Poor GUI
- Slow technical support

2.8 A custom-made simulator

2.8.1 Overall Description

We are consider here a hypothetical custom-made simulator developed just for this NGN project. A simulator would be programmed to exactly conform to the requirements of our NGN simulator. Depending on the developer, it could be written in nearly any language, though C++, Java and C# would be the most likely.

2.8.2 Custom Simulator: Modelling capability

All the key models for NGN would be developed.

R1: Models of VoIP protocol stacks
Grade: +++ (fully supported)

R2: Support of MPLS-TE and RSVP signalling protocol
Grade: +++ (fully supported)

R3: Support of failure modelling and different routing protocols.
Grade: +++ (fully supported)

R4: Support of optical layer protocols

Grade: +++ (fully supported)

R5: Support of different traffic
Grade: +++ (fully supported)

R6: Models of Quality of Service (QoS) mechanisms.
Grade: +++ (fully supported)

R7: Support of different kinds of network structures.
Grade: +++ (fully supported)

R8: Analysis of typical performance measures,
Grade: +++ (fully supported)

R9: Input/output data in XML format
Grade: +++ (fully supported)

3.8.3 Custom-made simulator: Credibility of simulation models

The protocol models could be developed directly from the RFCs and standards, ensuring their accuracy. With a higher budget and longer lead-in time than stated below, actual implementation of protocols could be ported on various sources, such as the Linux Kernel. For performance reasons, it might be beneficial to abstract some details and algorithms used in the protocols. While these may be less accurate, they could be almost as credible. The scope for doing this would need to be earlier clarified.

R10: Support of credibility of simulation models
Grade: +++ (fully supported)
Without a large user base, there will not be much verification of the network models. The developers could take responsibility for the credibility of the models.

2.8.4 Custom-made simulator: Credibility of simulation results

R11: Quality of sources of randomness
Grade: +++ (the highest quality).
The best currently known PRNG would be implemented

R12: Sequential (on-line or off-line) analysis of simulation output data
Grade: +++ (sequential on-line analysis would be supported)
The simulation results could be processed by the simulator itself, or it would be linked with Akaroa2, to unsure that the final results would be with the required level of statistical

precision.

R13: Possible cooperation with Akaroa2, an automated sequential controller of simulation in MRIP scenario

Grade: +++ (fully supported, to take advantage of parallel processing for speeding up sequential simulations).

2.8.5 Custom-made simulator: Extendibility

A custom simulator would be designed for achieving with its extendibility in mind.

R14: Ability of extending/adding new simulation models of protocols and mechanisms in a reasonable time

Grade: +++ (fully supported)

Source code should be part of the package. Getting the developer to add a new module could entail a long delay.

2.8.6 Custom-made simulator: Usability

R15: Existence of GUI, which can be used as input/output interface

Grade: +++ (supported)

The interface to the simulator could be chosen to be what would suit best. This could include: topology editor, event scheduler, statistic gatherers, visualization tools, and script editor. It is most likely that the simulator would be separate from the user interface, to allow for a more powerful and flexible interface.

R16: Existence of a good manual and an introductory tutorial, explaining how a given simulator can be used.

Grade: +++ (supported)

With modern software development practices, an API document should be created, incluidng at least some test cases. The quality of introductory tutorial and manual could be what is required. Post-development technical support might be limited depending on who the developers are and what support they will provide.

2.8.7 Custom-made simulator: Cost and availability

R17: Costs of commercial license

Grade: + (the development could take 9 to 18 months)

A rough indication of cost would be between $100,000 to $300,000. This does not significantly change if using an event simulator (such as from SSFNet or NS2) as its core or if developing a new one. The lead-in time before the simulator would be available would be around 9 to 18 months, with a small amount of ongoing development and updates after that.

This cost could be offset by selling licenses for it to other organizations.

2.8.8 Custom Simulator: Strengths and Weaknesses

A custom-made simulator would strictly conform to the needs set down in the Requirements document for the development of the NGN simulator. Therefore all the items covered would be its strengths, as it would be purposely designed for this NGN research project. Its only weakness would be that it does not exist at the moment, thus its first version would be delivered with a delay of approximately one year.

3. Comparison

A comparative study of the main features of the simulators surveyed in the previous section is contained the table below. It has all eight simulators considered in the previous chapter listed in consecutive columns. The rows list scores given each simulator for fulfilling the basic requirements R1-R17, discussed in Section 1; see also Appendix A.

	OPNET	NS2	OMNET++/ OMSET	GLASS/ SSFNET	QualNet	J-Sim	TOTEM	Custom made
R1	++	++	++	-	++	-	-	+++
R2	+++	++	+	+++	+++	+	++	+++
R3	+++	+++	++	++	+++	++	++	+++
R4	++	+	++	+++	-	++	-	+++
R5	+++	++	++	+++	+++	+	++	+++
R6	+++	+++	++	++	+++	++	++	+++
R7	+++	+++	+++	+++	+++	+++	+++	+++
R8	+++	++	+	++	+	+	+	+++
R9	+++	+	+++	+++	+++	+++	+++	+++
R10	++	++	+++	-	+++	-	+	+++
R11	+	+++	+++	?	?	+	?	+++
R12	+	+++	+++	?	?	?	?	+++
R13	-	+++	+++	?	++	?	?	+++
R14	+++	+++	+++	+++	+++	+++	++	+++
R15	+++	+	+++	+++	++	+++	++	+++
R16	+++	+++	++	+	++	+	++	+++
R17	+	+++	?	+++	+	+++	+++	+

These seventeen requirements were considered in the context of eight simulators. However, when it was obvious that a simulator failed a vital requirement, such as the requirement R1 by GLASS/SSFNET, J-Sim and TOTEM, R4 by QualNet and TOTEM, and R10 by GLASS/SSFNET and J-Sim, we have not approached the technical staff of these simulators for further questions related with their implementations of R11-R13. This could be done later if the missing information was required.

One obvious conclusion from the table is that none of currently available simulators satisfies all requirements. This problem could be solved by designing a custom-made simulator, discussed in Sec. 3.8. As stated there, the custom-made simulator would strictly conform to all requirements. It cost would be comparable to the costs of commercial licenses for currently existed simulators,. Later such new simulator could even bring if it were sold commercially. However since it does not exist at the moment, its first version would be delivered with a delay of approximately one year.

Looking at existing simulators, the table clearly points at three simulators as the best candidates for being used in this research project:

- two commercial simulators: OPNET and OMNET++/OMNSET, and
- one freeware: NS2.

The main difference between OPNET with OMNET++/OMNSET is that the latter be used with sequential on-line control of the final simulation results (since it can cooperate with Akaroa2, automated controller of simulations). This cannot be done with OPNET, since under OPNET's current policy the access to the source code is restricted.

The costs of commercial licenses of OPNET and OMNET++/OMNSET are expected to be comparable. However, OPNET offers richer library of simulation models needed in this research project. Thus, when simulated processes do not have too large variances, i.e. very long simulation experiments are not required to obtain the final results with small statistical errors, OPNET is a good choice. This should be the case in simulations planned in our research project in the nearest future, as we foresee conducting experiments with different enforced failure scenarios. However, if investigations of networks with highly correlated processes were involved, then OMNSET should be seriously considered as the candidate for our NGN simulator. This will be also the natural choice of an NGN simulator when OMNEST becomes equipped in the full library of MPLS and RSVP-TE models.

Among non-commercial simulators, freely distributed under open-source policy, NS2 is clearly the best choice. Its library of NGN-relevant models is comparable with that of OMNET++/OMNSET, and only a bit behind OPNET. However, its main advantage over OPNET is again the fact that NS2 can be used with sequential on-line control of the final simulation results (since it can cooperate with Akaroa2, automated controller of simulations).

J-Sim, GLASS and TOTEM are all java-based simulators. The MPLS model in J-Sim forwards packets according to the configuration of the forwarding table. It does not include any label distribution protocol, so LSPs must be setup statically. There was supposed to be an RSVP-TE implementation added to the simulator but this work has not been finished. Because of the lack of a signaling protocol, we decided not to consider J-Sim further.

GLASS also looks a promising candidate for the NGN project in future. Although the name

implies that it has been developed mainly for GMPLS simulations, it can be used for MPLS simulations as well. The simulator has implemented MPLS forwarding and label distribution with LDP, CR-LDP and RSVP-TE. However, its library does nor include models for VoIP. Additionally, after some basic testing, we found that this simulator was quite unstable. We tried to contact the developers of this simulator, but answers were returning very slowly. Because of these problems, we decided not use GLASS in our NGN project at this stage. But we will keep watching the evolution of this simulator and try to keep in touch with the developers for more detailed information, in case if the updated version is published soon. GLASS could be a good Java-based simulator for GMPLS and multilayer resiliency simulation studies in the future stages of the NGN project.

TOTEM is a relatively young project started in November 2003, so it is not as complete as other simulators. The current version of TOTEM comes with far smaller library of modules and protocols than the other simulators considered here. It is mainly a toolbox of traffic engineering methods used in IP/MPLS networks. It can be used as flow-oriented simulation tool to understand the effect of some actions taken.. The actions can be LSP creation, IGP metric change, link failure, etc. TOTEM usually takes a topology and a traffic matrix as input and simulates network in that scenario.

After such filtering of the results of our survey work, we have decided to apply OPNET and NS2 in our simulation case study reported in the next Section. In addition, exploiting our experience with QualNet, the case of our simulation study based on QualNet will be also reported.

4. A Simulation Case Study

A scenario for the simple simulated case study, set up for studying different simulators, has been already described in [6]. We will assess performance of the three selected simulators on the basis of Checkpoints C1-C8 described below.

C1: Setting network topology
The core network consists of 5 nodes: includes five core routers (two LER and three LSR:) and four user work stations . All routers are IP-based routers. The links between core routers are DS1 (1.544 Mbps), while those between workstations and core routers are DS3 (44.736 Mbps).

C2: Introducing explicit traffic
There are two explicit applications assumed run in the network: (i) the HTTP application between a user and a server (httpUSER and SERVER), and (ii) VoIP application between two users (voiceUSER1 and voiceUSER2).

C3: Introducing background traffic
The specific background traffic pattern is assumed as carrying 500,000 bits/second, 500 packets/second and lasting for 1 hour, i.e., until the end of simulation.

C4: Running OSPF protocol (with load balancing)
Firstly, we run OSPF protocol with load balancing. We use OSPF scenario and its performance measurements such as link utilizations and rerouting convergence times as the baseline for comparison with the MPLS scenario specified in C5

C5: MPLS setup and LSP configuration

Then, we set up each router as operating under MPLS, by setting up FECs, traffic trunk and static LSPs, which require a strict hop-by-hop configuration. The attributes associated with static LSPs involve their types as well as the nodes on the paths, together with corresponding actions at specific node and assigned labels. We setup two static LSPs as below,

- Primary path: Node 0, 2, 1- shortest path
- Secondary path: Node 0, 3, 4, 1- hardly found with traditional routing protocol, e.g., OSPF

C6: Configure egress/ ingress LER nodes

Configure egress and ingress nodes, so as to divided/mapping different traffic into different paths, e.g., the voice application passes through the primary path, which involves Node 0, 2, 1; while HTTP application is running on the secondary path, which consists nodes 0, 3, 4, 1.

C7: Introducing a failure condition

Configure failure events by assuming the link between node 0 and node 2 failing at the time t=1800 seconds during one hour simulation.

C8: Collect results and compare the performance measures

We expect that the following results can be collected from simulation of our network in OSPF and MPLS scenario, upon the link/node failure condition:

- Re-convergence time
- Each link utilization
- Packet loss in failure
- Results for voice packet end-to-end delay and delay variation.

4.1 Simulation in OPNET

In this section, we verify our checkpoints of simulators' performance and show how the operations they are related with are implemented in OPNET.

C1: Setting network topology

In OPNET, it is very convenient to set up a network topology by using GUI. In the reported case study, the nodes and links were placed and configured with their features as shown in Figure 4.1.1. In

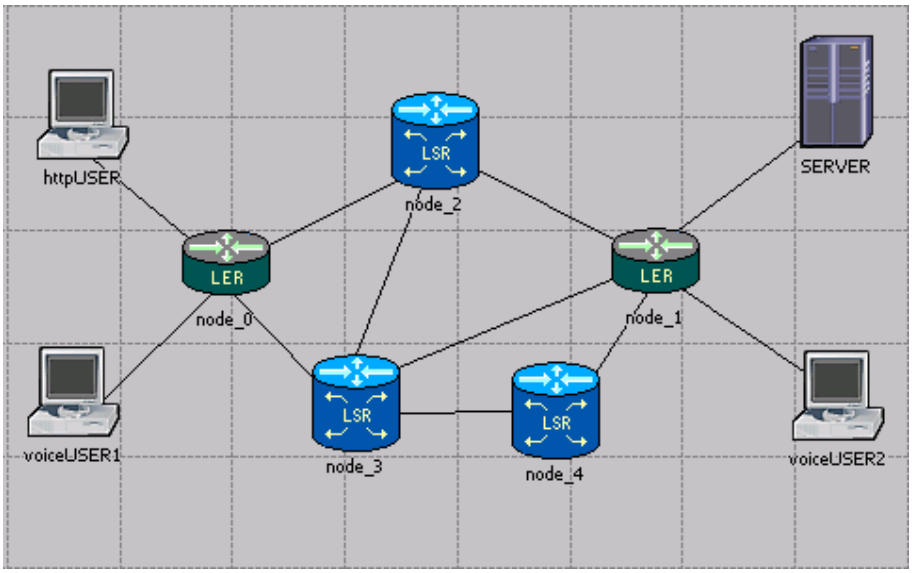

Figure 4.1.1 Network Topology

C2: Introducing explicit traffic

It was also very easy to set up different traffic demand between nodes as shown by Figure 4.1.2.

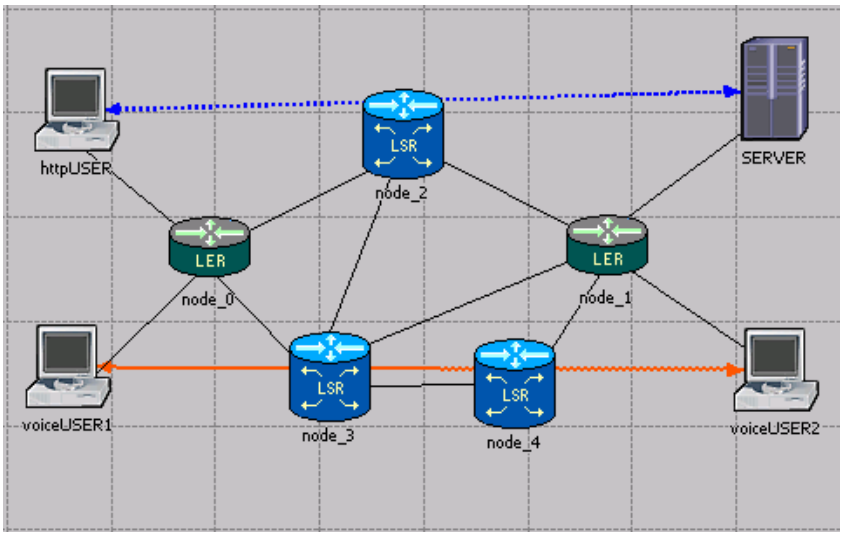

Figure 4.1.2 Traffic demand

C3: Introducing background traffic

The background traffic has been added as shown in Figure 4.1.3.

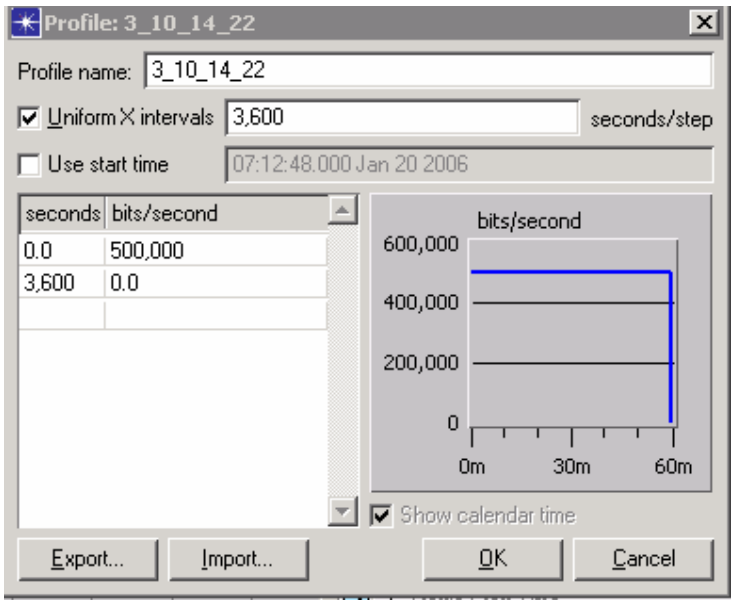

Figure 4.1.3 Background traffic in each demand

C4: Running OSPF protocol (with load balancing)

We have enabled OSPF in the network and displaced the possible paths for each traffic flow, as shown in Figure 4.1.4 and Figure 4.1.5.

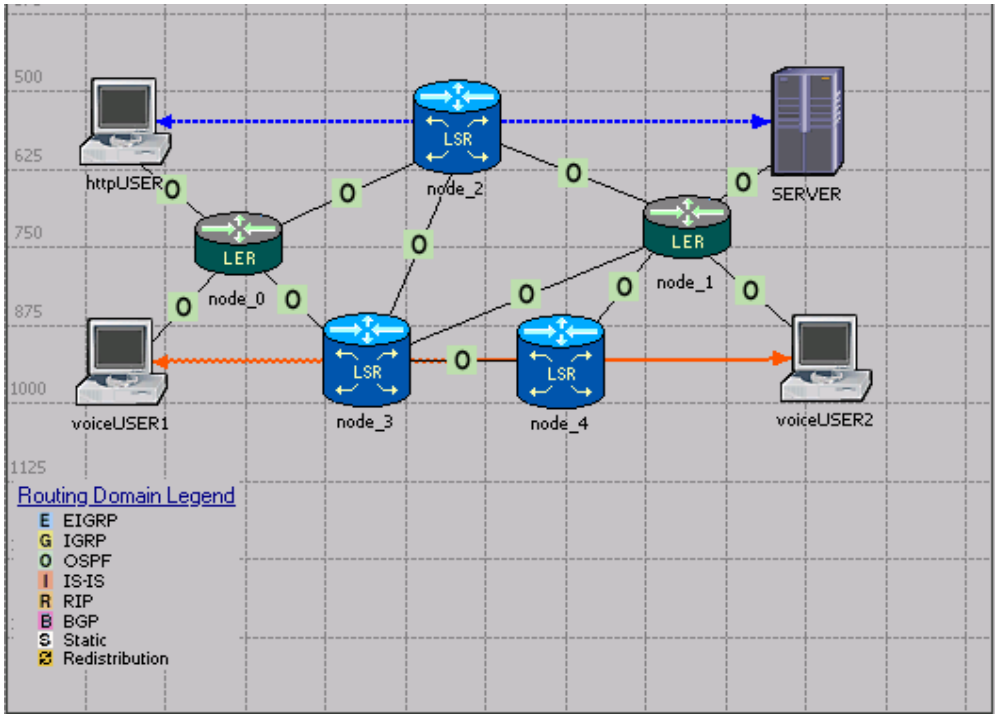

Figure 4.1.4 Configure OSPF

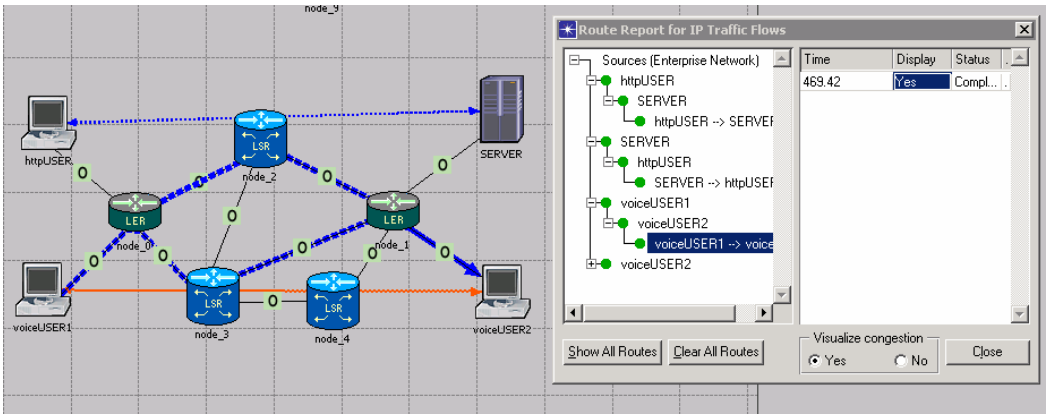

Figure 4.1.5 Display routes for each demand

C5: MPLS setup and LSP configuration

We have done the following configurations to enable MPLS function in OPNET, see Figures 4.1.6, 4.1.7 and 4.1.8.

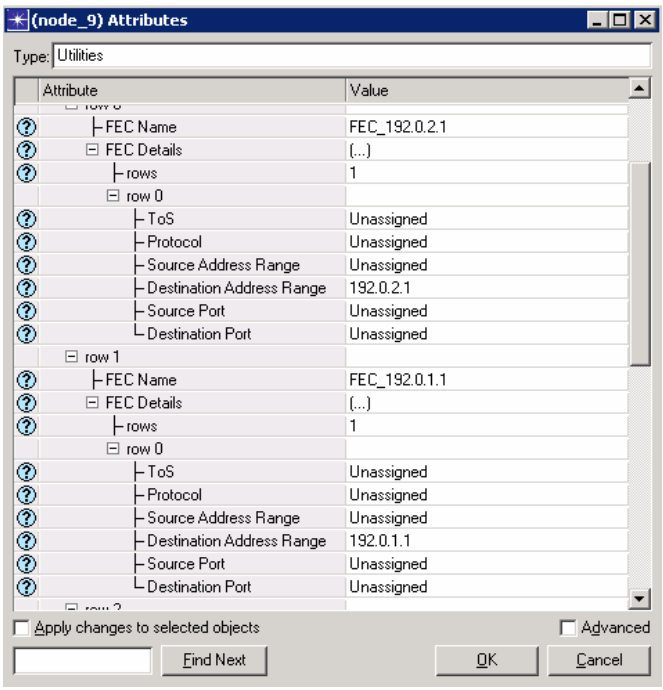

Figure 4.1.6 Configure FECs

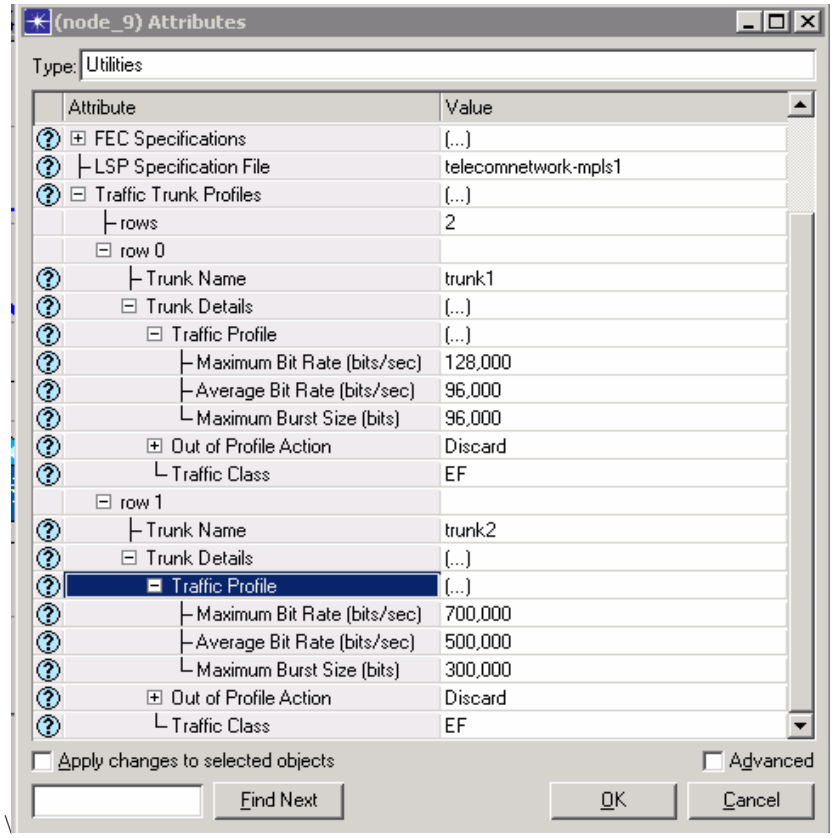

Figure 4.1.7 Configure Traffic Trunks

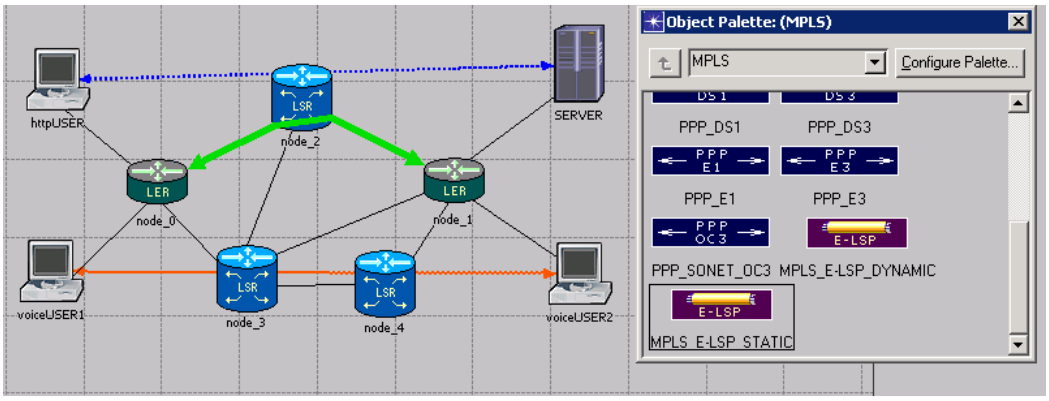

Figure 4.1.8 Configure Static LSPs

C6: Configure egress/ ingress LER nodes

We configure LER nodes and display the routes for demand as shown in Figure 4.1.9 and Figure 4.1.10

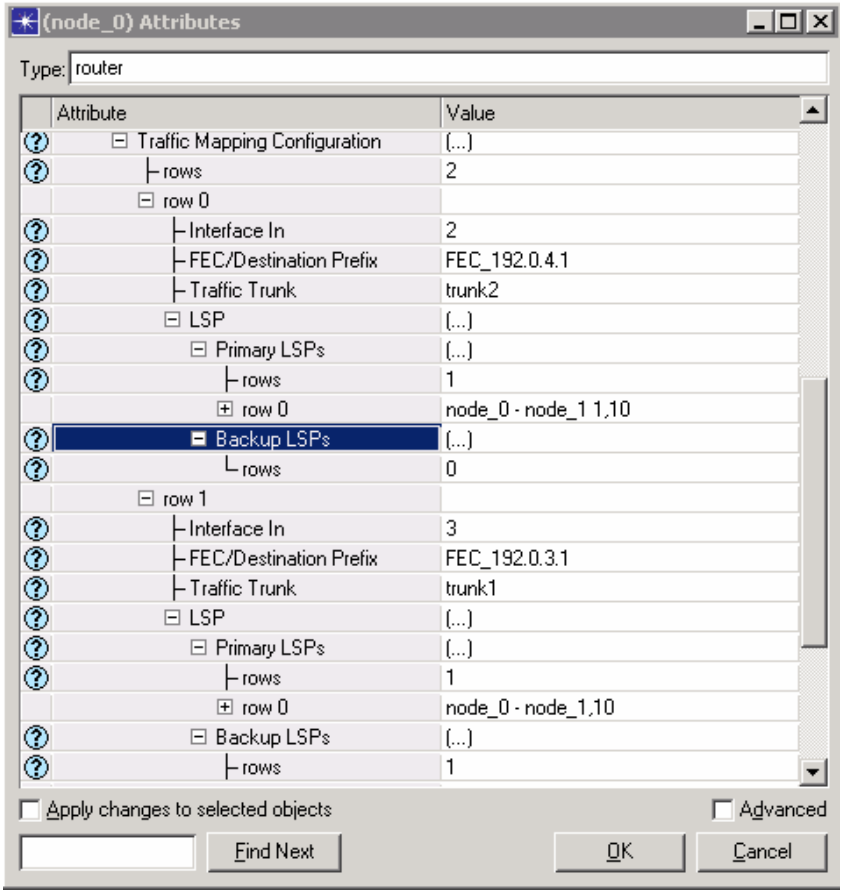

Figure 4.1.9 Configure LERs

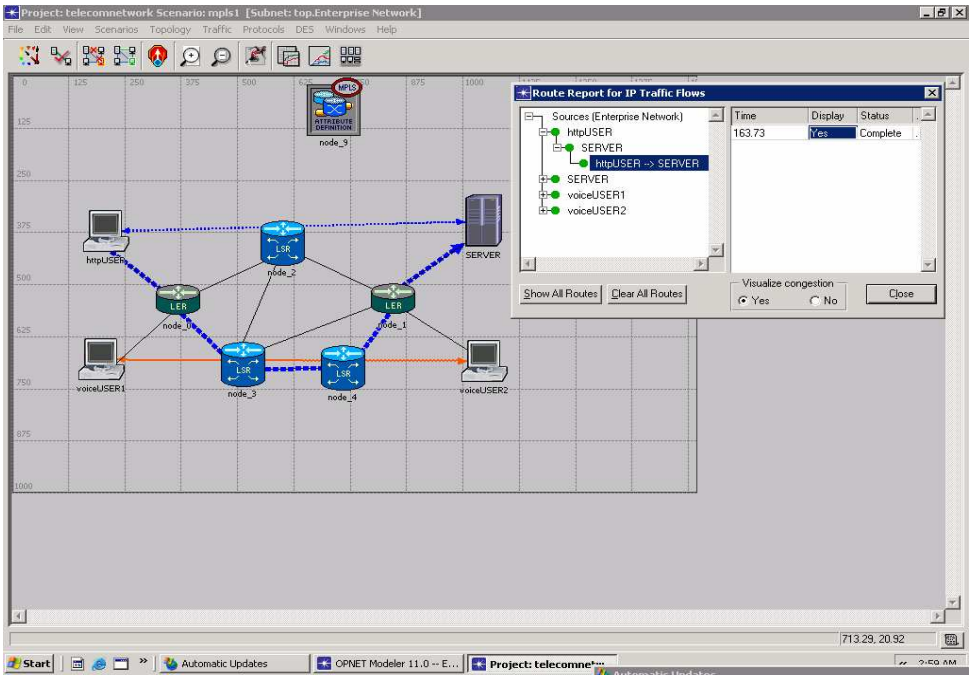

Figure 4.1.10 Display routes for demand

C7: Introducing a failure condition

In OPNET, we can configure failure event as shown in Figure 4.1.11.

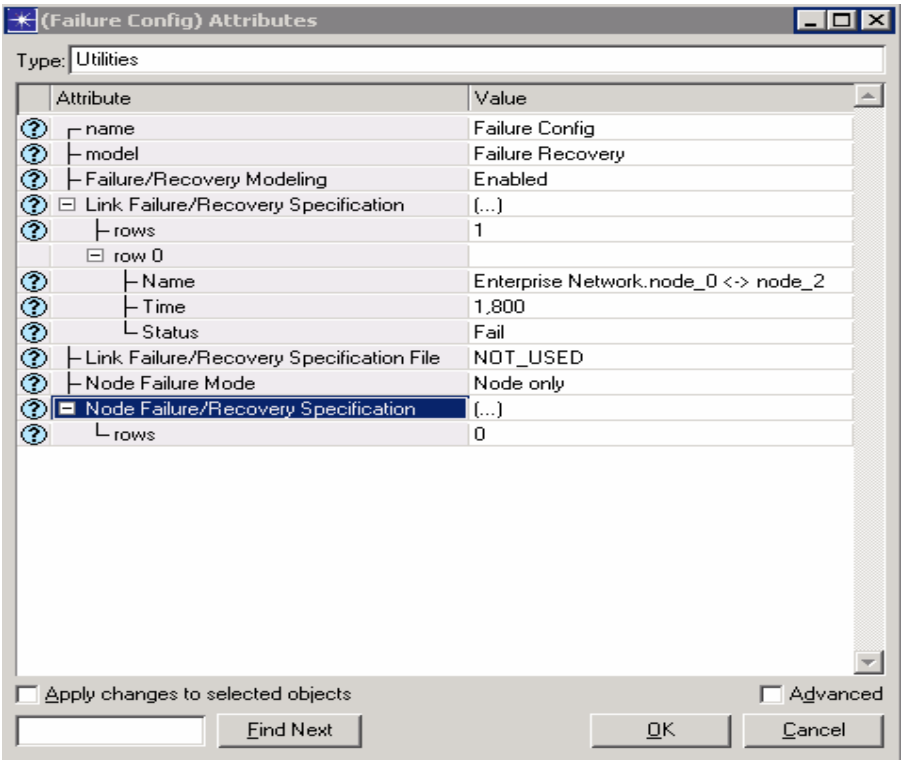

Figure 4.1.11 Configure Failure Event

C8: **Collect results and compare performance**

OPNET provide powerful GUI for collecting and comparing performance measurement as shown in Figure 4.1.12 and Figure 4.1.13

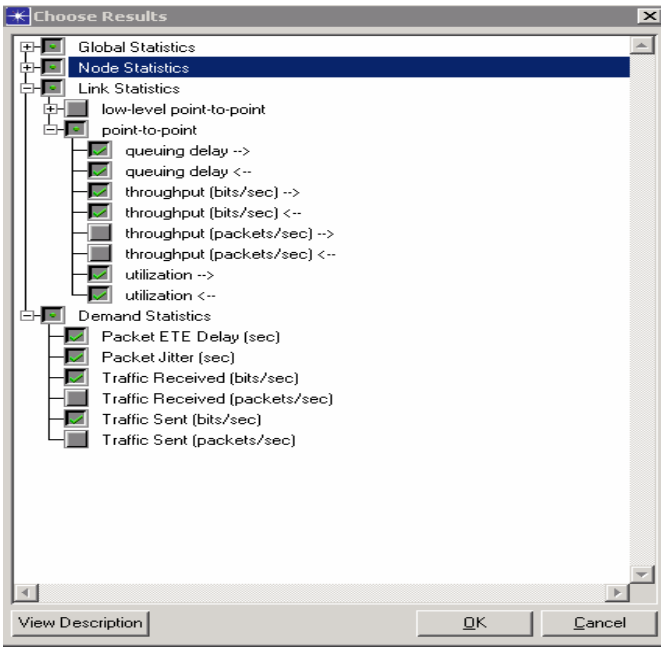

Figure 4.1.12 Collect performance measurement

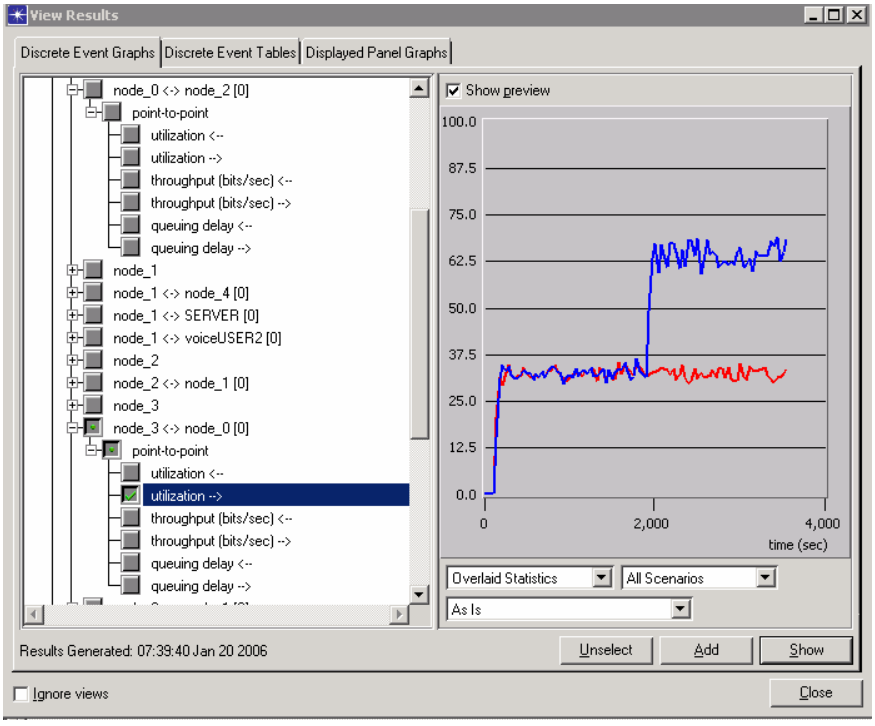

Figure 4.1.13 Compare Performance Measurement

4.2 Simulation in NS2

In this section, we verify the checkpoints as mentioned above and show how they are implemented in NS2.

C1: Setting network topology

The OTcl code to set up the required network topology is shown in Listing 4.2.1 and shown as seen in NAM in Figure 4.2.1.

.

```
set ns_ [new Simulator]
set node(0) [$ns_ node];
set node(1) [$ns_ node];
set node(2) [$ns_ node];
set node(3) [$ns_ node];
set node(4) [$ns_ node];
set node(5) [$ns_ node];
set node(6) [$ns_ node];
set node(7) [$ns_ node];
set node(8) [$ns_ node];

# Set some aliases
set voip1 $node(0)
set voip2 $node(1)
set httpuser $node(2)
set server $node(3)

# labels and colours for nam
$node(4) label "LER_0"
$node(5) label "LER_1"
$node(6) label "LSR_2"
$node(7) label "LSR_3"
$node(8) label "LSR_4"
$voip1 color red
$voip1 label "VoIPUser1"
$voip2 color red
$voip2 label "VoIPUser2"
$httpuser color blue
$httpuser label "HttpUser"
$server color blue
$server label "Server"

# Create links
```

```
$ns_ duplex-link $node(0) $node(4) 44.736Mb 1ms DropTail
$ns_ duplex-link $node(1) $node(5) 44.763Mb 1ms DropTail
$ns_ duplex-link $node(2) $node(4) 44.763Mb 1ms DropTail
$ns_ duplex-link $node(3) $node(5) 44.763Mb 1ms DropTail
$ns_ duplex-link $node(4) $node(6) 1.544Mb 1ms DropTail
$ns_ duplex-link $node(4) $node(7) 1.544Mb 1ms DropTail
$ns_ duplex-link $node(5) $node(6) 1.544Mb 1ms DropTail
$ns_ duplex-link $node(5) $node(7) 1.544Mb 1ms DropTail
$ns_ duplex-link $node(5) $node(8) 1.544Mb 1ms DropTail
$ns_ duplex-link $node(6) $node(7) 1.544Mb 1ms DropTail
$ns_ duplex-link $node(7) $node(8) 1.544Mb 1ms DropTail
```

Listing 4.2.1

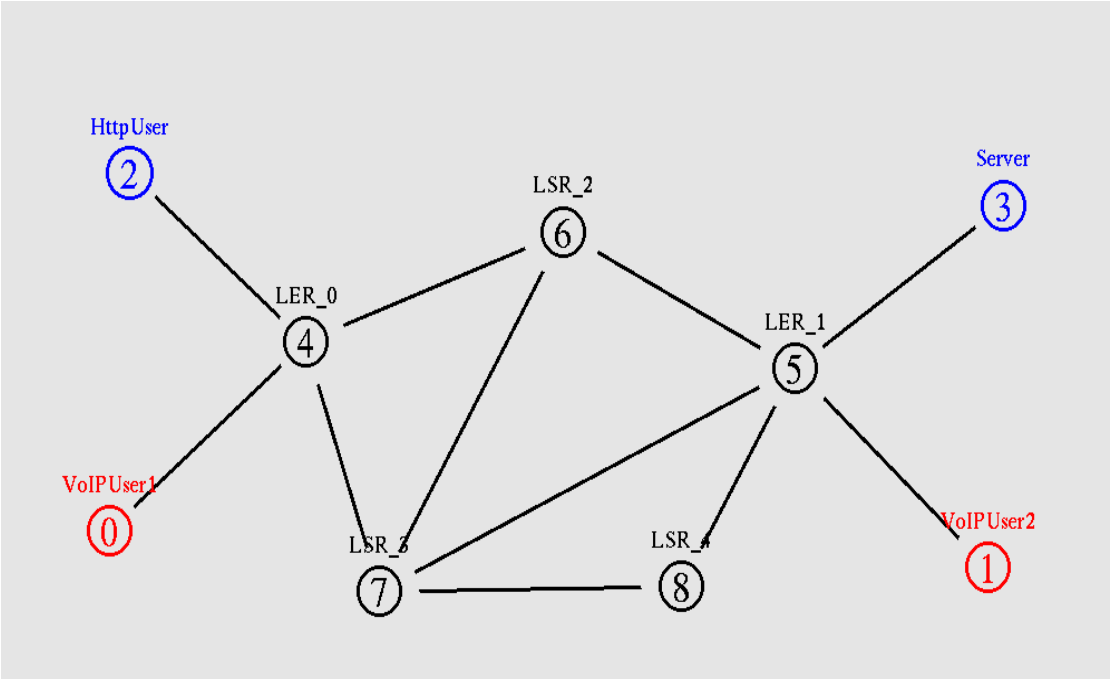

Figure 4.2.1 Network Topology

C2: Introducing explicit traffic

It is very convenient to set up different traffic demand between nodes, see Listing 4.2.2 below

```
#VoIP traffic
#sink attach to the node $voip2
        set vsink0_ [new Agent/Null]
        $ns_ attach-agent $voip2 $vsink0_
```

```
#source
        set vudp [new Agent/UDP]
        $ns_ attach-agent $voip1 $vudp
#traffic
        set vtraffic [new Application/Traffic/CBR]
        $vtraffic set packetSize_ 100
        $vtraffic set burst_time_ 0
        $vtraffic set idle_time_ 0
        $vtraffic set rate_ 64k
        $vtraffic attach-agent $vudp
#connect
        $ns_ connect $vudp $vsink0_
        set vsrc0 $vtraffic
        $ns_ at 1.0 "$vsrc0 start"
```

Listing 4.2.2 Traffic demand for VoIP

C3: Introducing background traffic

The traffic is set up in the previous section, see the traffic comment and the vtraffic variable.

C4: Running OSPF protocol (with load balancing)

To use an OSPF-like routing protocol, the LS (link state) RtProto Object is set up, as in Listing 4.2.4.

```
$ns_ rtproto LS
```

Listing 4.2.4 Configure OSPF

C5: MPLS setup and LSP configuration

Each node has to be configured with MPLS, and the static RSVP-TE paths and defaults need to be set up. All the link Queue types where changed to the Classifier Based Queues (CBQ) for RSVP-TE and MPLS to work correctly. A number of other small changes were made to make it run which should not be necessary in future (such as turning off LS routing and turning on static routing).

The MPLS and RSVP-TE parts of the changes are shown in Listing 4.2.3

```
$ns_ node-config -MPLS ON

set node(0) [$ns_ node];
$ns_ add-to-mpls-list $node(0)
...
#MPLS Settings
```

```
# RSVPTE Settings
$ns_ PATH-color "purple"
$ns_ PATHERR-color "red"
$ns_ PATHTEAR-color "red"
$ns_ RESV-color "grey"
$ns_ RESVERR-color "red"
$ns_ RESVCONF-color "red"
$ns_ color 1002 green
Agent/RSVP set noisy_ 255
Agent/RSVP set refresh_ 30
Agent/RSVP set lifetime_factor_ 3
Agent/RSVP set ip6_ 0
Agent/RSVP set nam_ 1

# One RSVP-TE Agent on each MPLS Node
$ns_ configure-rsvpte-on-all-mpls-nodes

$ns_ cfg-cbq-for-SBTS 10 DropTail 990000.000000 0.010000
auto 0
$ns_ cfg-cbq-for-HBTS 10 DropTail 0.000000 0.000000 auto
0.0000
$ns_ cfg-cbq-for-STS 10 DropTail 0.000000 0.000000 auto
0.000000
$ns_ cfg-cbq-for-RTS 2040 DropTail 0.000000 0.990000 auto
0.0000
$ns_ bind-rsvpte-to-SBTS

set LSR(4) [eval $node(4) get-module "MPLS"]
set ses(1) [$LSR(4) session $node(5) 1]

set LSR(5) [eval $node(5) get-module "MPLS"]
set ses(2) [$LSR(5) session $node(4) 1]

$ns_ at 0.500000 "$LSR(4) PATH-resv-er $ses(1) 1000.000000
50 \
  50   $node(5) 1001 5 5 6_5"
$ns_ at 0.600000 "$LSR(5) PATH-resv-er $ses(2) 1000.000000
50 \
  50 $node(4) 1002 5 5 8_7_4"
```

Listing 4.2.3 Setting up MPLS and RSVP-TE

The code in Listing 4.2.4 was put in each flow of network traffic to identify the flow

and what class of service it used. The value 7 was changed each time to uniquely identify each flow.

```
$vudp set fid_ 7
$ns_ bind-flowid-to-SBTS 7
```

Listing 4.2.4 Per-traffic flow changes

C6: Configure egress/ ingress LER nodes

Some bugs were found in the NS2 patch which caused the packets not to get routed into the MPLS paths, but the simulation set up for it should look like Listing 4.2.5. It is hoped that the bugs will be fixed in the next version of the patch due in the next few months.

```
$ns_ at 0.800000 "$LSR(4) bind-flow-erlsp 5 7 1001"
$ns_ at 0.900000 "$LSR(5) bind-flow-erlsp 4 8 1002"
```

Listing 4.2.5 Configure LER ingress and egress

C7: Introducing a failure condition

In NS2, we can configure failure event as shown in Listing 4.2.6. The restoration event is similar with a different time and “up” instead of “down”.

```
$ns_ rtmodel-at 2.0015 down $node(4) $node(6)
```

Listing 4.2.6 Configure Failure Event

C8: Collect results and compare performance

The simulation file creates two trace files, a NAM file and a tr file. Both these files are ASCII text and can easily be parsed to find the information or statistics or run through animation and graphing programs (nam and xgraph or tracegraph) respectfully.

The simulation file can itself calculate statistics and values, and display them in anyway it wishes, as it is a full general purpose programming language. NS2 can be used with the Akaroa2 simulation controller which runs the simulation enough times to ensure credible results with low statistical error.

4.3 Simulation in QualNet

The simple 5 switch scenario is simulated and results for checking points C1 to C8 are given below.

C1 : Setting network topology

In QualNet, it is very convenient to set up a topology i.e., nodes and links and configuring their features as shown in Figure 4.3.1.

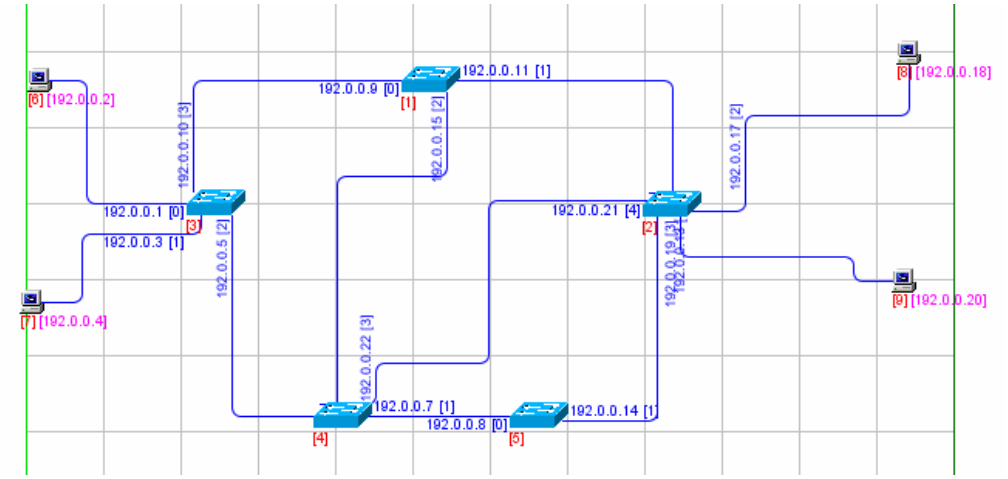

Figure 4.3.1 Network Topology

C2 : Introducing explicit traffic

It is possible to setup traffic flows between arbitrary nodes in the topology as shown below. Figure 4.3.2 below

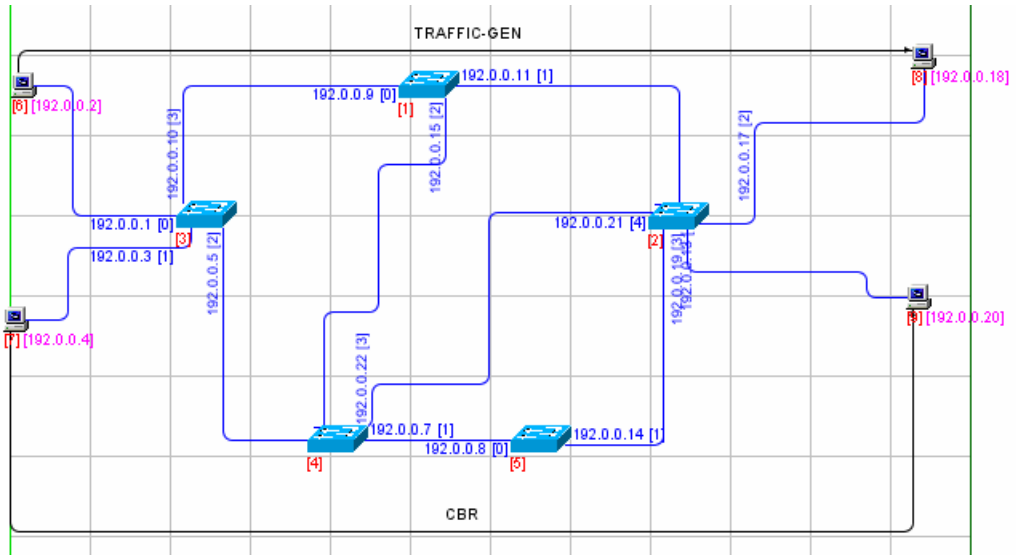

Figure 4.3.2 Traffic demand

C3 : Introducing background traffic

Background traffic is added per interface, see Figure 4.3.3 below. This has the effect of reducing the bandwidth of the link/s which is available to the explicit applications.

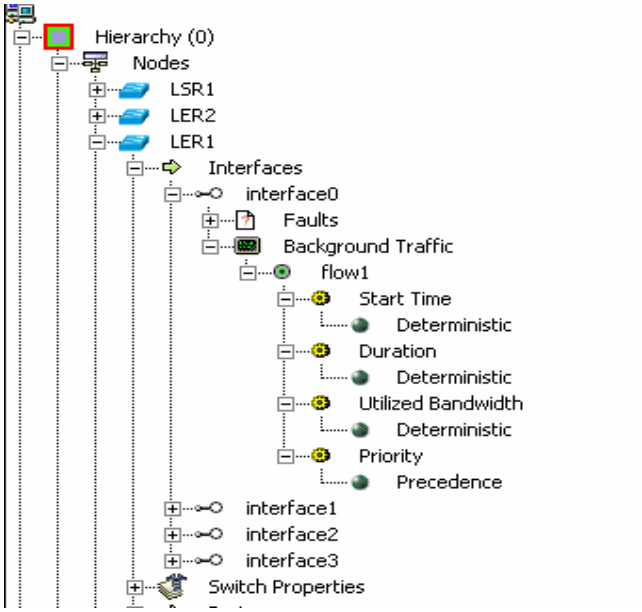

Figure 4.3.3 Background traffic

C4: Running OSPF protocol (with load balancing)

We enable OSPFv2 as the routing protocol in the network, see Figure 4.3.4

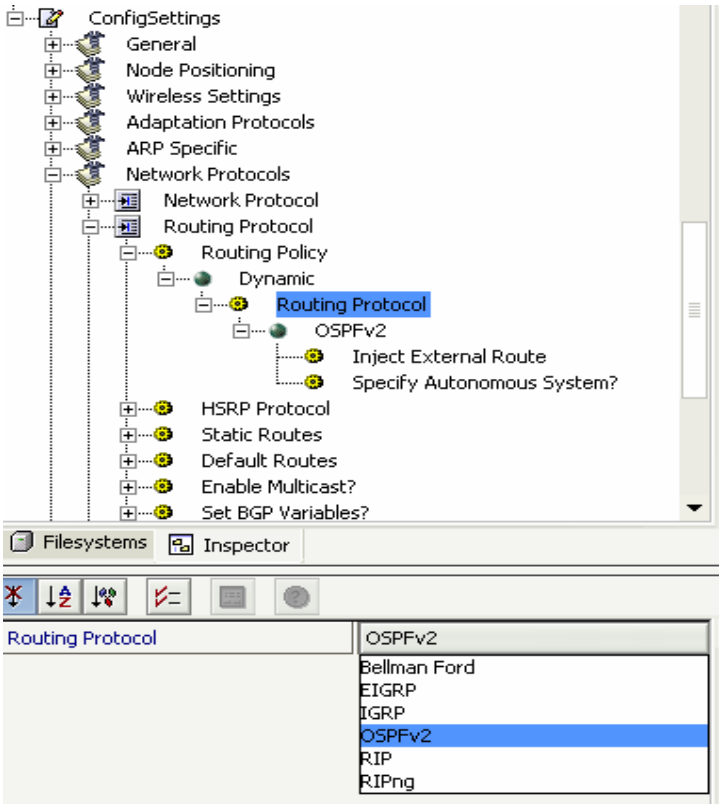

Figure 4.3.4 Configure OSPF

C5: MPLS setup and LSP configuration

We do the following configurations to enable MPLS function; see Figure 4.3.5. If static paths are required a file containing the static routes must be specified.

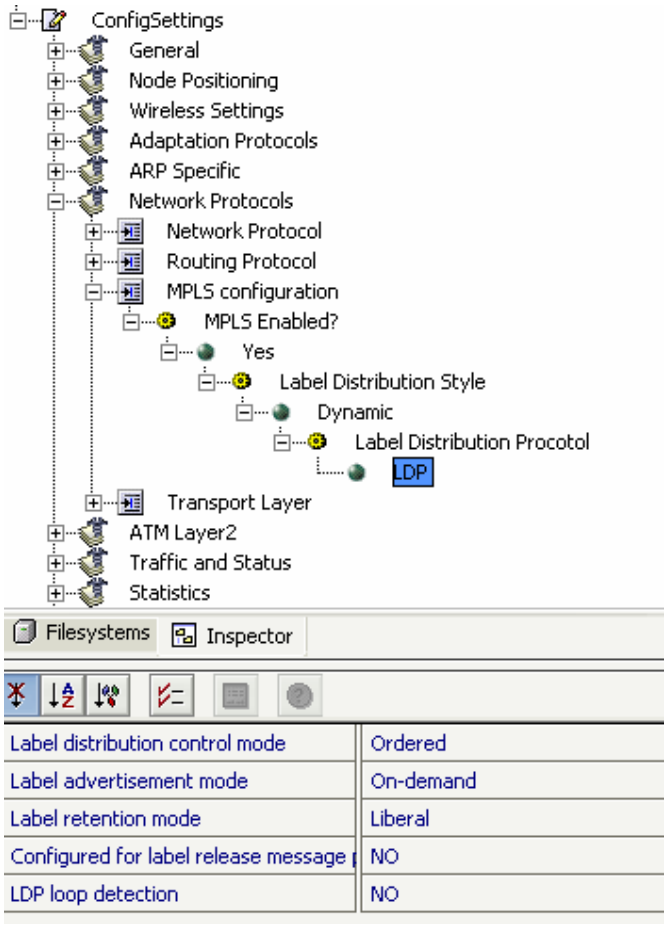

Figure 4.3.5 Configure MPLS

C6: Configure egress/ ingress LER nodes

No specific settings are available for nodes which are edge routers.

C7: Introducing a failure condition

We can configure link failures as shown in Figure 4.3.6. Alternatively an interface failure can be scheduled using the same procedure. In the case of failure the action taken to find an alternate path is routing protocol dependant and no explicit settings are available.

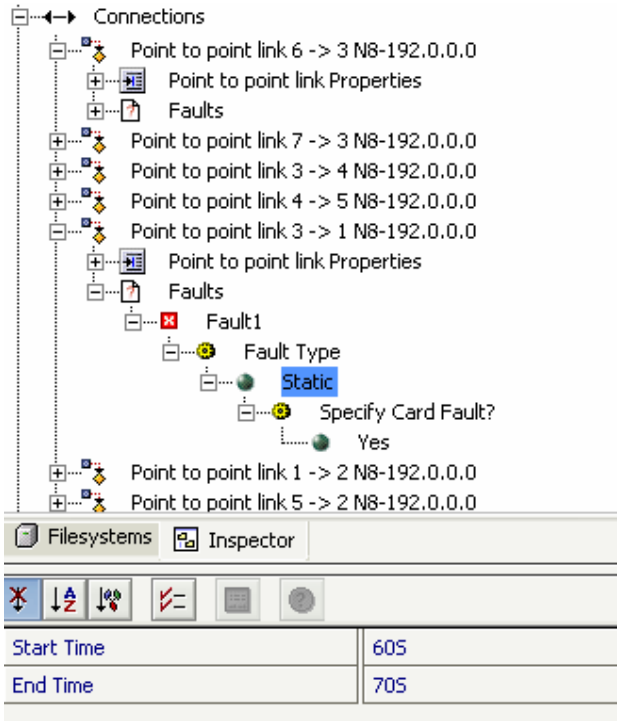

Figure 4.3.6 Configure Failure Event

C8: Collect results and compare performance

QualNet provides a certain amount of support for displaying results. It should be pointed out that none of the provided graphs are time based. If time based outputs are required trace files need to be produced which can then be processed in an external program like Matlab. The available built-in statistics and flavour of graph are given in Figures 4.3.7 and 4.3.8 respectively.

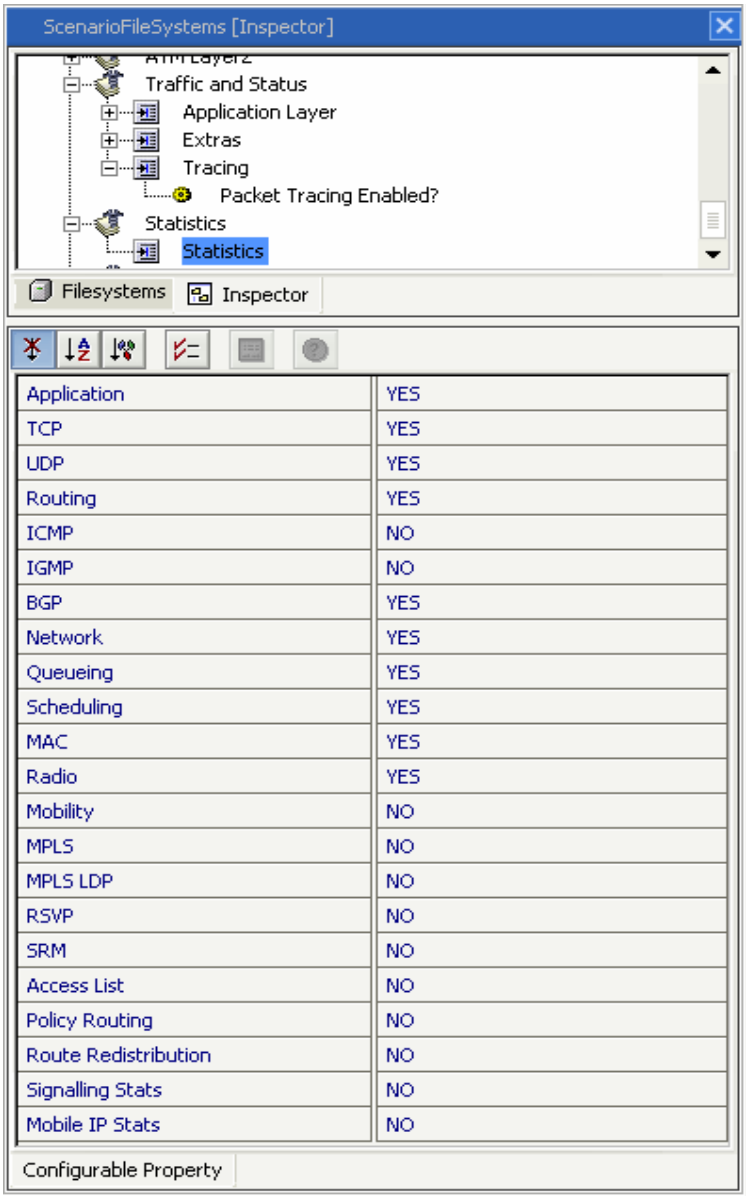

Figure 4.3.7 Available built-in statistics

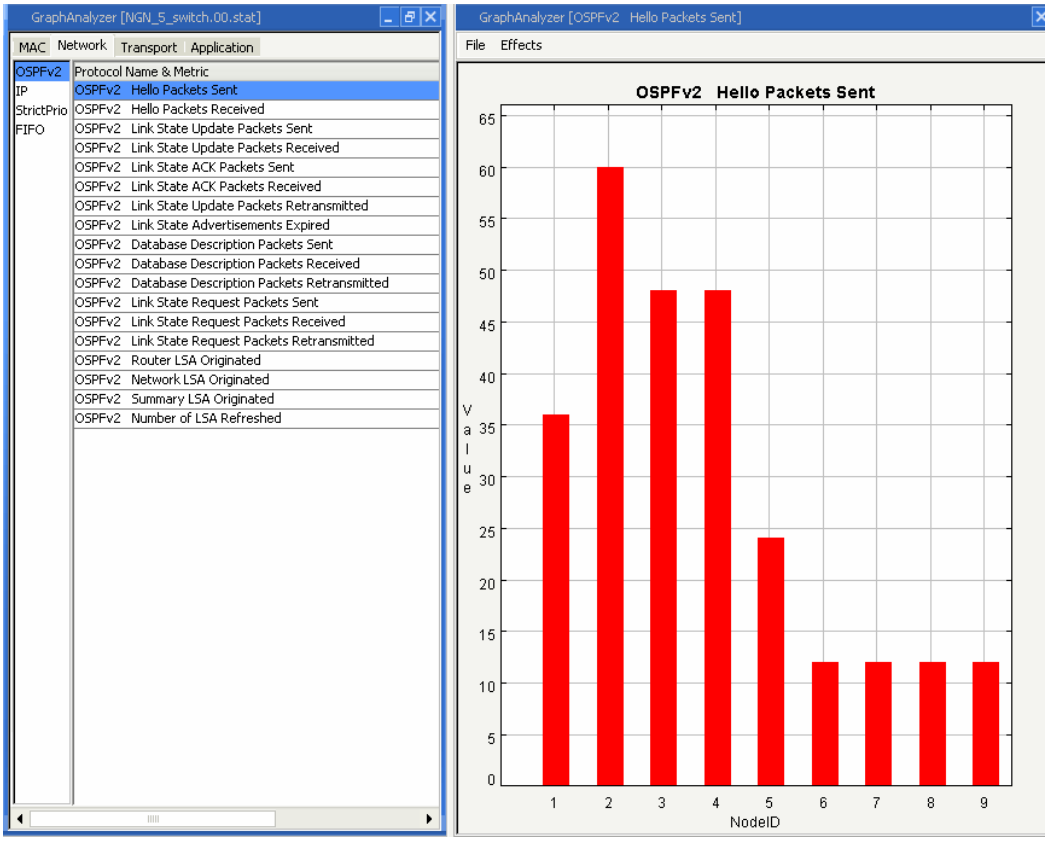

Figure 4.3.8 Sample graph

5. Conclusions

The results reported in the previous section show that simulation studies of service availability and resilience in NGN by using various simulators are feasible and can be done relatively easy. Unfortunately, when applying NS2 we have detected some bugs in a software patch related with C6, see page 65. Thus, until these bugs are fixed, we cannot produce NS2 results for our simulation case study. Because of that, at this stage we cannot validate mutually compare simulation results produced by OPNET and NS2 as we had planned. We still intend to so, if the bugs are fixed before this research project's deadlines.

Nevertheless, we have been able to practically justify our choice of OPNET as the tool for the next stage of our research project. It was easy to use OPNET and there was no problem obtaining the required results.

As mentioned, one of the other two simulators, NS2 with Akaroa2 and OMNSET with Akaroa2 (the commercial version of OMNET++), should be considered as an option if our studies of NGN required simulations of strongly correlated processes. Already now the NS2's library is satisfactorily well equipped for our NGN research, although we have detected a software bug in it (the bug should be relatively easy to fix). In the case of OMNET++/ONMSET, some further software development (in particular, addition of missing MPLS and RSVP-TE components) would be required.

6. References:

1. A. W. Law and W. D. Kelton. *Simulation Modeling and Analysis.* Edition 3, McGraw-Hill, 2000.

2. K. Pawlikowski, H.-D. J. Jeong and J.-S. R. Lee. On Credibility of Simulation Studies of Telecommunication Networks. *IEEE Communications Magazine*, Jan. 2002, vol. 40, no. 1, pp. 132-139.

3. M. Schoo, K. Pawlikowski and D. McNickle. A Survey and Empirical Comparison of Modern Pseudo-Random Number Generators for Distributed Stochastic Simulation. Technical Report TR-CSSE -3/05, University of Canterbury, 2005; see www.cosc.canterbury.ac.nz/research/reports/TechReps

4. G. Ewing, K. Pawlikowski and D. McNickle. "Akaroa2: Exploiting Network Computing by Distributing Stochastic Simulation". Proceedings of European Simulation Multiconference (ESM'99, Warsaw, June 1999), Int. Society for Computer Simulation, 1999, pp.175-81

5. K. Pawlikowski and D. McNickle. Speeding Up Stochastic Discrete-Event Simulation. Proceedings of European Simulation Symposium (ESS'01, Marseille, France, October 2001), ISCS Press, 2001, 132-8.

6. "A Case Study on Simulation Scenario". Canterbury Reports, Canterbury 03-Simulation Case Study.pdf , https://secure.wand.net.nz/ngn, ,Dec. 21, 2005

Appendix A List of Requirements

R1	Models of VoIP protocol stacks.
Comment	VoIP Protocols Stacks includes SIP, SIGTRAN, H323, H.248/MEGACO, MGCP, NCS, RTP/RTCP

R2	Models of MPLS-TE and RSVP signaling protocols.
Comment	Support of MPLS modeling with Traffic Engineering and complete model of protection and restoration behavior, at least MPLS layer, further to integrate optical layer schemes. In addition, support of RSVP signaling protocol.

R3	Models of typical failures and different routing protocols.
Comment	Support for link and node failures modelling, as well as for different routing protocols (including IP/MPLS and optical layer routing, as well as 'traditional' routing protocols such as OSPF, for comparative studies), with ability of fast modifications of routing protocols.

R4	Models of optical layer components.
Comment	Support specification of services and infrastructure at the SONET/SDH, wavelength, and fiber levels.

R5	Models of different teletraffic scenarios.
Comment	Models of different teletraffic scenarios such as HTTP, VoIP, etc.; ability of supporting simulation based on traces of real teletraffic.

R6	Models of Quality of Service (QoS) mechanisms.
Comment	Support of DiffServ scheme (scheduling/marking)

R7	Models of different network architectures
Comment	Support of different kinds of networks, including various freely configurable, interconnected, hierarchical and/or non-hierarchical networks.

R8	Analysis of typical performance measures.
Comment	Analysis of typical performance measures, such as packet loss, mean packet delay, jitter, mean rerouting time, etc.

R9	**Input/output data in XML format**
Comment	Specify the type and format of input/output data, and the computing platform(s) and programming language(s) supported by a given simulator (it has been decided that XML format of input/output data would be preferred).

R10	**Credibility of the simulation models**
Comment	Has anybody assumed responsibility for validation and verification of simulation models used in a given simulator?

R11	**Quality of sources of randomness**
Comment	Does the pseudorandom number generator (PRNG) used in a given simulator represent a satisfactory source of primary randomness for running adequately long simulations of networks and for producing accurate final estimates of the performance measures considered

R12	**Sequential (on-line or off-line) analysis of simulation output data**
Comment	Does a given simulator support sequential, on-line analysis of simulation output data ? If not, can it support off-line sequential analysis of simulation output data ?

R13	**Possible cooperation with Akaroa2, an automated sequential controller of simulation in MRIP scenario**
Comment	If a given simulator does not support sequential on-line analysis of simulation output data, could it be linked with Akaroa2, a sequential controller of simulation in MRIP scenario ?

R14	**Ability of extending/adding the new simulation models of protocols and mechanisms in a reasonable time**
Comment	Does a given simulator allow extending its library of simulation models by new simulation models of protocols and other networks mechanisms in a reasonable time**?**

R15	**Existence of GUI, which can be used in input/output interface**
Comment	Is a given simulator equipped in a GUI, which can be used as the input/output interface?

R16	**Existence of a good manual and an introductory tutorial, explaining how a given simulator can be used.**
Comment	A given simulator is user-friendly if its users have an access to a good manual and an introductory tutorial explaining how the simulator can be used. The response to the technical questions from customers should be within a reasonable time, e.g., 3 working day.

R17	**Cost of licenses**
Comment	The cost of a commercial license of a given simulator and its availability.