Simulation Modelling Practice and Theory, Volume 17, Issue 6, July 2009, Pages 1011-1031

Review: Network Modelling and Simulation Tools

Muhammad Azizur Rahman, Algirdas Pakštas, Frank Zhigang Wang

PII:	S1569-190X(09)00019-7
DOI:	10.1016/j.simpat.2009.02.005
Reference:	SIMPAT 781
To appear in:	*Simulation Modeling Practices and Theory*
Received Date:	19 June 2007
Revised Date:	9 January 2009
Accepted Date:	13 February 2009

Review: Network Modelling and Simulation Tools

Muhammad Azizur Rahman[1], Algirdas Pakštas[1], Frank Zhigang Wang[2]
[1, 2]London Metropolitan University, 66-220 Holloway Road, London N7 8DB England
[3]Cranfield University, Cranfield, Bedfordshire MK43 0AL, England
[1]m.rahman@londonmet.ac.uk, [2]a.pakstas@londonmet.ac.uk, [3]f.wang@Cranfield.ac.uk

Abstract
Computer network technologies have been growing explosively and the study in computer networks is being a challenging task. To make this task easy, different users, researchers and companies have developed different network modelling and simulation (MS) tools. These network MS tools can be used in education and research as well as practical purposes. They vary with their characteristics. This paper reviews some of the most important network MS tools developed recently. This paper also shows a classification of the tools used in communications networks.

Key words:
Modelling, Network, Researcher, Simulation, Tools

I. Introduction

Network is a complex mix of applications, communications protocols and link technologies, traffic flows and routing algorithms. It is immensely complex. Network design process is a challenging task, requiring designers to balance user performance expectations with costs and capacities. One of the obvious approaches to deal with complexity is the use of *modelling and simulation (MS)* techniques. There are currently available various MS tools created by the separate companies and groups of researchers in academia which are intended for use as practical and/or educational tools for network design. There is a need to study the existing network MS tools, their functionality, advantages and function procedure and special features. Nearly every general-purpose network design tool works the same way [1], sometimes having complementary features.

The network design process is a challenging task, requiring designers to balance user performance expectations with costs and capacities. External factors, such as government policies and regulations, the competitive situation, available technological services and products are adding complexity to the design process. Organisational strategies, culture and policies also affect the planning and design process. The amount of human and technical resources in the data communication functions of the organisation can also strongly affect the choice. Electronic communication is so ubiquitous in modern business that it is hard to develop an overall strategic vision that is comprehensive and at the same time detailed enough to be useful. Additionally, these factors can change and make the planning process even more complex. The business role of the proposed network application adds extra complexity to planning process.

Network analysis, architecture and the design process have been considered art, combining an individual's particular rules on evaluating and choosing network technologies together with knowledge about how technologies, services and protocols can be meaningfully combined [2, p2]. During the network design process, technologies for each area of the network are evaluated and chosen, and strategies to connect these technologies across the network are developed. The network design process includes a requirement analysis, flow analysis, logical design, physical design and finally making a decision about addressing and routing [2]. The network design process may have multiple goals that are often hard to formulate, define or compare with each other. In order to achieve a good design, it is often essential to build a network model and apply certain tools to evaluate different scenarios. Network design is also achieving design goals by applying the trade-off, within constraints to parts of the whole network.

Researchers typically conduct the simulation using only one simulation package. Different tools show different results with the same simulation models. There is a wide variation in their results. It is very hard to assume which tool's results are accurate [3]. This divergence and/or inconsistent results suggest that one or all of the output might be wrong/not perfect. Basically, no tool is perfect/accurate as it depends on protocol stack, traffic generation parameters, application, usage profile, package difference, incorrect parameter setting and improper level of details. Insufficient statistical analysis of independent simulation runs and improper data collection techniques can produce ambiguous or inaccurate conclusions. Simulation assumptions imposed by the tools always affect research outcomes. A surveyed was performed on Mobile Ad Hoc Networking and Computing (MobiHoc) from where significant Shortfalls were found [4]. The paper [4] presents the results of the survey with the summarize common simulation pitfallsed studies. Thus, it would be useful, if a model could be analysed using different tools simultaneously.

Unfortunately, errors in simulation models or improper data analysis often produce incorrect or misleading results. Simulation is a powerful method, but sometimes it has pitfalls. Generalization and lack of rigor can lead to inaccuracy, which can result in wrong calculations or inappropriate implementation decisions.

This purpose of this paper is to review these MS tools so that researchers can select the right tools for their experiments. Different tools are analysed in this paper and according to their characteristics they are classified. Different researchers classified the network tools in different ways. But none to them extensively considered all of the tools together. For example network discovery tools are not considered. One [5] of the papers shows the classifications of the tools. A brief review of the some of the tools is also performed in the paper [6].The paper has the following objectives: a) to review recent network Modeling and Simulation tools; b) to help researches to select the right tools for their needs; and c) to identify issues that seem to be open to further research. The information of tools described in this paper is collected from manual, different papers, and website. Some of the tools from different groups (e.g., Brite, Ns-2, Glomosim, Delite, Opnet, Ethereal, THC-RUT, etc.) are practically tested by the authors.

Rest of the paper is organised as follows. Section II describes the classification of MS tools. Section III, Section IV, Section V and Section VI describe the most used and recently developed different kinds of MS tools. Finally, Section VII provides conclusions.

II. Classification of the network MS tools

There are different types of network MS tools available and they are used in many ways. According to the uses and availability (we analysed 100 tools), the network tools can be classified into four groups: analytical tools, simulation tools, topology discovery tools and topology generation tools. The potential candidate tools are considered and selected according to some criteria. The primary criterion is that the tools must have text-formatted input and/or output file(s) so that the information (stored data) of the model can be retrieved. The secondary criterion is that the tools have to have rich networking features (e.g. supported description of protocols and applications, various types of networks (LAN/MAN/WAN); then the usefulness of the tool, i.e., the output of the tools and its analysis capabilities. Finally, the availability of the tools in the market has been considered.

Analytical tools help to design a network model and calculate different factors (e.g. reliability, utilisation) of the model. On the contrary, simulation tools simulate dynamic behaviors (e.g. packet passing, link failure, TCP protocol) of a network model besides modelling. The topology generation tools help to generate small as well as large topologies based on different algorithms. Finally, the topology discovery tools extract the actual network information from an existing system and map them graphically and/or in text format. This type of tool is used for network management purposes. Quantitative network design tools produce more accurate and defensive results than qualitative methods. On the other hand, analytical tools may design and analyse the network models more efficiently and provide solutions more quickly than a simulation tool but do not always achieve the required accuracy. Most often, the dynamic behaviour of the protocol cannot be visualised using the analytical tools but can be visualised using simulation. The topology which is generated or discovered by a specialised tool can be visualised and analysed by simulation and analytical tools.

Both analytical tool and topology discovery tool can be classified into two: educational (free) and commercial tools. Network simulation tools can be divided into three: educational (free), commercial and specialised tool. The Fig. 1 shows the classification of the tools. This paper is focusing on the most recent MS tools.

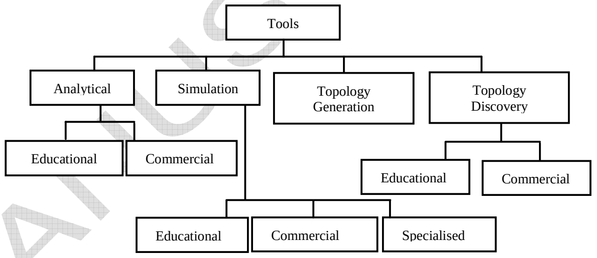

Fig. 1: Classification of the Network Design and Simulation tools

III. Analytical tools for network design

The analytical tool typically formulates the network planning problems as optimisation models where given cost functions are minimised (maximised) under a set of constraints [7]. Normally a single model tries to capture many relevant aspects of the problem. When an optimal solution to the network model is found, the values of the decision variables can be used to decide the optimal action to be taken.

A. Educational Tools

DEsign tool LITE, Delite [8] is an educational and practical wide area network (WAN) design tool, which can produce network designs of limited size using a set of the embedded network design algorithms. Delite can produce graphical displays representing network nodes and links as well as some additional analysis data (delay analysis, reliability analysis, average delay analysis of the link, average number of hops, link utilisation for every separate link between nodes, utilisation of each node, overall network model utilisation, etc.). There are seven files that are handled by the Delite tool for each network design. However, the most important for the users are .gen (original node information, coordinates and available link types) and .net (additionally has table of links between nodes i.e. actual design) files. Links between nodes can be generated using a few design algorithms. Thus, various designs may have different costs, delays, reliability, average number of hops etc. Other files are used for different purposes: noeqip.tbl and param.tbl are predefined (used to supply some network parameters), .cst file describes the costs associated with link types (e.g. T1, T3, D96 etc.), .req file describes capacity of each link type, .inp file contains the names of files related to a particular model. The users can edit all of the files mentioned above as they are in ASCII text format. The algorithms used for designing the topology of a network model supported by Delite are: Prim, Primdijkstra, Tour (Nearest Neighbour), Tour (Farthest Neighbour), Esau-Williams, Sharma, Multispeed Tree Design, Nearest Neighbour-Esau_Williams, Multicenter Esau_William, Mentor, Incrementour. **Contributor:** Robert S. Cahn. **Supports**: different types of network design algorits (mentioned above). **Languages:** C. **Advantages:** 1) Text formated input and output files so user can manually change the input files to see the effects; 2) Population is taking part in designing the models; 3) World map can be included in the model to see the suitability of the model; 4) can include more algorithms in this tool. **Limitation:** 1) limited number of nodes supported as this is a small scale network tool; 2) documentatoin is no rich enough.

Cappuccino [9] is a web based non-commercial network design tool. Using the tool, users can design network model, calculate maximum traffic of each link and a lower bound of the link cost of the network. The switches can be scattered randomly inside the rectangular area (0, 0, 400, 400). The properties of the switches are expressed by switch name, X coordinate, Y coordinate, Alpha (originating capacity for a switch) and Omega (termination capacity for a switch), which can be changed by the users. The Alpha denotes the maximum communication rate, which is permitted by the network to originate from switch to the rest of the network. The Omega denotes the maximum communication rate, which is permitted by the network to terminate to a switch from the rest of the network. Traffic between switches is limited by using set-pair-constraints, which specifies the maximum traffic between two switch sets. While designing a network model, initially 20 switches are scattered in a region. User can add, delete and modify the switches as required. Topology is chosen using predefined algorithms implemented in the tool or manually. Then traffic constraint, which specifies maximum traffic between two sets of switches, is chosen. The cost coefficient is proportional to the link length and a link has two-bandwidth capacity (incoming and outgoing). The tool uses shortest path routing algorithm for traffic. The algorithms for network topology implemented in the tool are: best star, minimum panning tree (Krustral's algorithm), delaunay triangulation and complete graph. Using shortest path routing and taking account the constraints, the tool calculates the maximum traffic and a lower bound cost on every link so that user can see how far away the network is from the optimal. The lower bound cost of the link is calculated based on complete graph. Users can also analyse the traffic of each link, switch and traffic constraints of each using window menu. User can edit the switch information and update the calculation. The tool is very user friendly. It is platform independent. This tool will help network users/designers create their non-blocking network model. **Languages:** Java. **Advantages**: This is web based. **Limitation:** Very limited applications.

B. Commercial Tools

XNetMod [10, 11] (based on NetMod tool), a LAN modelling tool that allows users to analyse the performance of a configuration before implementing it. Three approaches to performance evaluation of this tool are analysis, simulation and measurements. XNetMod can be used networking environment consisting of thousands of computers sites. XNetMod provides customised forms to enable one to specify input parameters for the elements. XNetMod supports two types of topology: ring and bus and two connectors: Router and Bridge. To facilitate the concept of bottom-up design, XNetMod allows users to cerate subnetworks. At present there is only one element type, Subnet in the Subnetwork element group. XNetMod supports analytic and simulation techniques. It can summarise the entire results of an analysis or simulation of just the result for a particular node. **Supports**: FDDI, token ring, token Bus, generic bridge, generic router, subneting. **Languages:** C. **Advantages:** 1) ggraphically display and manipulate network topology; 2) ease of subnetwork definition; 3) three types of analysis of data are supported; 4) flexibility of easily add or delete subnetworks; 5) ability to provide substantial user interaction for either simple parameter change of major reconfiguration of LANs and backbone; 6) hierarchical modelling capability for extremely large models; 7) query and generate reports based on currently displayed model.

TND-Tool (Topological Network Design Tool) [12] is used to find out an effective solution to the network topology design problem. The idea behind developing the TND-Tool are twofold: first providing network designer and operator with a tool to facilitate their work, second offering topology design researchers an environment for testing and representing the output of their network design algorithms. It permits to select between node types, link types and data bases for different kinds of networks. A Network viewer allows to display the current topological network structure during the optimisation process. Users, engineers are able to set up a network interactively by inserting nodes and links, typing in network information, e.g. traffic values, moving, adding and removing some part of the network. According to the tariffs defined, the TND-Tool calculates the cost for an existing network on the screen or a network resulting from an optimisation algorithm. The TND-Tool consists of four major modules: 1) representation module is the graphical user interface of the system providing all important facilities needed by the network engineer; 2) optimisation module enables integration of network optimisation algorithms; 3) cost calculation module is the entity providing the rules for calculation of network cost, e.g. tariff tables and cost functions; 4) data base module provides all the needed data for the functionality of the system. Currently, the TND-Tool is used for designing GSM *(Global System for Mobile communication)* and DCS1800 *(Digital Cellular System*

1800) fixed networks as part of the cellular mobile communication networks. The TND-Tool is not limited to cellular networks and has been designed as a platform for designing all kinds of networks such as Wireless Local Loop, Satellite networks, ATM networks and so on. **Supports**: WAN, Wireless, ATN, Satellite. **Advantages:** 1) an interactive menu and mouse-driven tool; 2) Web based. **Limitation:** 1) Limited only to calculate the cost of design; 2) Limited number of topoloy design algorigm supported.

NetRule [13] is designed to combine the best features of the mathematical analysis and simulation tool. It is mainly performance prediction tool to design the actual network. NetRule can model and stress test very large networks (1,000 to 10,000 nodes). It has a customised GUI like a simulator. It uses closed-form mathematical analysis that evaluates network load and performance in seconds with detail and accuracy that matches the best practice of mathematical analysis. The tool lets users to create or open network models and libraries, edit them graphically, and evaluate their performance. The diagram can animate message flow, highlight bottlenecks and shows utilisation and delays bars as components. It uses shared objects definitions to reduce modelling time and allow global changes. The NetRule library includes: 1) LANs, including many CSMA/CD, token ring and polling protocols; 2) Frame relay clouds with access ports and virtual circuits; 3) ATM switches and SONET links; 4) Routers, hubs, bridges and Ethernet switches; 5) Point-to-point T1 and other speeds; and 6) FDDI; 7) ISDN. Reports (output) are available for object definitions as well as for evaluation statistics that can be sorted or can be exported to spreadsheet, database, presentation or graphics packages. The most interesting features of the NetRule are period objects, which let one to establish different workloads during different times of day (day and night period) or time of years. Histogram can be produced for performance analysis. Normally the report includes: computer delay/job, computer utilisation, job cost, job delay, link delay/pkt, link utilisation. Another important feature of NetRule is that it can import network topology that has been discovered with a network management tool. **Contributor:** Analytical Engines Inc. **Supports**: CSMA/CD, Frame relay, ISDN, FDDI, ATM, SONET, point to point T1 connection. **Languages:** Java. **Advantages:** 1) supports very large scale network (1,000 to 10,000 nodes); 2) user can import ALL or any part of a network and its parameters to build or modify a NetRule model. E.g .cap files, .csv format file,MIB data; 3) Supports generalisation option for clients, servers, and/or applications (for example, mathematically determining the typical client); 4) it can extract from traffic files and make a new file for all computers which can potentially be servers, all applications (using port numbers), user traffic flow.

XNP (Extensible Network Planner) [14] is a comprehensive network design tool for both extensible and conventional networks. It is implemented using several methods and incorporated them into a previously unpublished java-based software tool, Cappuccino. Its objective is to configure a least-cost network that accommodates any traffic pattern satisfying the given traffic constrains. XNP allows network designers to quickly create, configure and evaluate network designs by providing a convenient graphic-based interface and automated functions. There are five essential building blocks in designing a network in *XNP*. Users can create a trail topology in the blank drawing area of the XNP main window. XNP allows two types of links (unidirectional and bidirectional). Application format (number of processing steps, number of machine instructions and bandwidth required) can be added or removed by user-friendly buttons. XNP provides three ways to specify traffic expectations. XNP process will iterate through all links and processing nodes to compute the capabilities for them. XNP will redraw the topology to show the differences in resources capabilities again and again. The thickness of each links shows its capability relative to other link's capabilities. Users can evaluate the network configuration by computing a lower bound on the cost of the best network configuration. By adjusting options of lower bound and therefore by considering design restrictions for using resource again and again, one can obtain more accurate and useful measure. XNP allows three different ways to view the drawing area-only the trail topology and the design space together or the design space only. XNP also allows to manage multiple designs at a time so than one can try for different trial topologies and compare them. **Supports**: supported algorithms are: 1) Complete network; 2) Delaunay triangulation; 3) Delaunay triangulation with trimmed links; 4) Link complement; 4) Minimum spanning tree; 5) Random link adder; 6) Random node adder; 7) Star network; and 8) Symmetric link adder; 9) Geographical planar projection- longitude and latitude; 10) Grid network – N x M grid network; 11) Torus network- N x M torus network. 12) Localised traffic constraints; 13) Proportioned Pairwise traffic constraints. **Languages:** Java. **Advantages:** 1) Extensible.

NPEST (Network Processing Estimator) [15] is a tool for packet processing cost on a network node. All of the calculations are done with the help of processing instruction per packet. When sending a packet from one node to another, the following delays occur: (1) transmission delay (the time it takes to send a packet into a wire); (2) propagation delay (the time it takes to transmit a packet via a wire); (3) processing delay (the time it takes to handle a packet on the networking components); and (4) queuing delay (the time the packet is buffered before it can be sent). Normally the cost for the delay (2) and (4) are simulated using standard simulator. The delay for (1) and (3) are ignored. From this tool, per packet processing cost, per-byte cost and total processing costs (processing delay) are found and this cost can be simulated.

The NPEST is a framework on top of which packet processing functionality can be implemented and simulated using an actual processor simulator. NPEST allows implementing packet processing functionality as a simple C program with the NPEST framework providing all the necessary packet trace processing and memory management.

Table 1: Summary of network analytical tools

Tools	Usages	Design Algorithm(s)	Target Network Model	Outputs	Stored Data File	Platform
Delite	Educational	Some predefined algorithms	WAN	Graphical Link analysis, node utilisation, overall utilisation in percent, delay analysis	Special formatted Text file	Windows
Cappuccino	Educational	Predefined Design algorithms	WAN	Link cost, capacity etc	Graphical output, no text file	Windows Web based
XNetMod	Commercial	User defined topology.	LAN, Subnetwork	Delay, average packet queue length, packet delay time, utilisation,	Graphical output, No text file	Windows Unix
TND-Tool	Commercial	User Defined topology.	WAN, Wireless, ATN, Satellite	Analysis of Cost of deigned network.	Graphical output. No Text file	Windows
NetRule	Commercial	User defined topology	WAN, LAN, ATM switches, ISDN, FDDI	Computer Delay, Computer Utilisation, Jobs cost, Job delay, Link Delay, Link Utilisation, histogram representation.	Report can be imported to spreadsheet, database	Windows
XNP	Commercial	Predefined design algorithms	WAN	Resource capacities, Cost analysis of links.	XNP formatted text file	Windows UNIX
NPEST	Commercial	Predefined algorithm	WAN	Cost analysis of Packet processing.	Need further investigation	Unix, Linux,, Redhat

The NPEST results can be used integrated and used in network simulations and nodes. One way is to integrate NPEST with a network simulator that uses actual packet traces and simulate the processing of the packets. The other way of using realistic processing cost estimates is to obtain statistics from the NPEST and the using these in network simulations or modules.

The NPEST framework implements the functions that are necessary to read and write packets, control the processor simulator and manage memory. The NPEST consists of the following sections: NPEST framework, NPEST application and NPEST API. The framework provides basic "layer 2" functions that prepare packet for processing, which includes reading the packet from the input trace file, extracting headers and writing back the processed packets to output trace file, manage memory allocated to packets and controls the processor simulator. The application component is the software that requires to be evaluated for processing cost. The API defines the interface between framework and the application. The cost is calculated using consideration on processor clock, processor instruction set, memory access time, flow and packet size, processor cycle.

Several metrics can be derived from NPEST prototype. Such as, instruction count (number of instruction executed), memory access (number of memory references performed), instruction mix (percent of instruction that belong to different categories). The outputs of the NPEST are overall statistics (total instruction, total memory access) and application statistics (total instructions, total memory access, average instruction/packet, average packet memory access/ packet, average lookup memory access/packet), instruction mix in percent (load, store, unconditional branch, conditional branch, instruction computation, FP computation). Using the number of instruction executed, actual time taken can be estimated. The memory access statistics gives an indication if memory could become a bottleneck for a given application.

This tool is helpful for the users interested for node cost analysis and can be used with integration with other network design and simulation tool. **OS:** Unix, Linux, Redhat. **Supported elements**: ARM, **Developed Languages:** C. **Outputs:** Performance analysis, observe dynamic behaviour with instant changes while simulation runs; observe the dynamic reaction of failure of network**.**

Table 1 shows the summary of analytical MS tools.

IV. NETWORK SIMULATION TOOLS

Simulation is the discipline of designing a model of an actual or theoretical or physical system and manipulating the model in such a way that it operates on time or place to compress it, thus enabling one to practice the interaction. Many unimportant details can be abstracted away and simulations can be completely repeatable. In a simulation, a mathematical/logical model is numerically evaluated over the period of interest and performance measures are estimated from model-generated data.

Simulations are complementary to analysis [16], not only by providing a check on the assumptions of the models and on the correctness of the analysis, but by allowing exploration of complicated scenarios that would be either difficult or impossible to analyse. Simulation plays a vital role in helping researchers to develop intuition. Simulation analysis is applicable to systems of almost any level of complexity.

A Educational Tools

ns-2 (Network Simulator) [17, 18, 19, 20], a VINT (Virtual Inter-Network Testbed) project from U.C. Berkeley/LBL/Xerox PARC, is a discrete event simulator targeted at network research, which provides substantial support for simulation of TCP, routing and multicast protocols. The simulator is using a Tcl/Otcl (Tool Command Language/Object Oriented Tcl) as a command and configuration interface. There are four types of files related to ns-2 simulator. Model is described in files .tcl or .ns which have common subset of commands but not exactly compatible between each other's. While simulator runs a model defined in .tcl/.ns file, simulation trace file (.tr) and animation file (.nam) are created during the session. Network Animator (.nam) files are used to visualise the behaviour of the network protocols and traffic of the model. Once created, users can play with the .nam file just like a media player and check the behaviour repeatedly. ns-2 facilitates the three broad themes of network research simulations: 1) selecting a mechanism from several options; 2) exploring complex behaviour; and 3) investigating unforeseen multiple-protocol interaction. **Supports**: TCP family, UDP, CBR, FTP, HTTP, Pareto, Exponential protocols, wires, wireless, unicast, multicast. **Languages:** both C++ and OTcl languages. **Outputs:** dynamic output. **Advantages:** 1) user can design model both manually or writing code; 2) easy configurable and fast simulation by using two different languages (OTcl and C++); 3) many protocol already implemented; 5) dynamic behaviour can be visualised using *nam* editor; 6) Open source code and can be extended for future. **Limitation:** 1) Long time to get used to usign it; 2) does not large scale simuaton (e.g., internet). 3) incomplete API; 4) no real time simulation; 5) no performance analysis is possible 6) badly documented source code; 7) no direct support of mobility and shared wireless radio channels; 8) unable to scale to networks of Internet size due to the computational requiremets of fine-grain packet-level simulation and memory needed to maintain queues at each network link.

Network Workbench [21] is a discrete event network simulator developed for the academic investigation of Internet protocol. It contains a complete protocol stack, abstracted from the Internet stack and a set of exercise that focus on critical protocol algorithms in the Internet stack. Network Workbench has several version, which are: version 0(1994), version 1(1994), version 2(1997), and version 3(1998). This tool supports several network topology, DLC error control, CSMA/CD collision backoff, optimal route computation, reliable transport, multicast, and LAN/MAN integration. It supports FDDI (Fiber Distributed Data Interface), OSPF (Open Shortest-Path-First). Workbench is abstracted from the internet (TCP/IP) stack. It contains five layers (application, transport, network, datalink control, and physical). The Network Workbench tool can be used to study of WAN topology, DLC (datalink control layer) frame formatting, DLC flow and error control, CSMA/CD local area network, network layer routing, reliable transport, multicast networking etc. Summary of statistics of the performance is produced at the end of the simulation. **Contributor:** J. Mark Pullen. **Supports**: network topology, DLC error control, CSMA/CD collision backoff, optimal route computation, reliable transport, multicast, and LAN/MAN integration, OSPF, FDDI, Ethernet, TCP/IP. **Languages:** C++. **Limitation:** Very limited implementation of protocols.

Netsim (M.I.T.'s Network Simulator) [20, 22] is a single process discrete event simulator used for the investigation of many aspects of Local Area Network (LAN). Netsim has three main goals: 1) flexibility of experiment of specifying the network and traffic; 2) simulate the accurate behaviour of Ethernet; and 3) has features that make running sequences of related experiments easier. The experimental data of Netsim simulation is stored in an experiment description file which contains: the layout of the network to be simulated, the traffic generation behaviour of the station on the network, information about repetitive of each run and about the sequence of parameters to be used for a series of experiments, etc. The entries specify the physical characteristics (e.g., length, packet size, data rate, number of attached stations) and traffic generation behaviour of the attached station. There are five types of traffic distributions in Netsim: exponential, uniform, deterministic, continuously queued and user defined discrete distribution (e.g. biomodal). Netsim collects information on many aspects of the operation of the simulated network. For example, average packet delay, histogram of packet queuing delays and collisions per packet transmitted, overall throughput (percentage utilisation), actual data rate, average, queuing delay, variance of queuing delay, observed packet transmission rate, average message size, total observed collision rate and total number of packet, maximum packet size, minimum packet size, maximum data rate etc. **Contributor:** MIT LCS Advanced Network Architecture group. **Supports**: Ethernet link, a point-to-point link, a switch (switches packets between several links), a host (about the same as a switch), Purdue's implementation of TCP, data supplier and consumer from TCP, a simple Poisson traffic source and a packet sink. **Languages:** C. **Advantages:** 1) Netsim is a publicly available ATM simulator originally developed at MIT; 2) its source code is freely available and modifiable; 3) Netsim has a user friendly Graphical Interface (GUI). **Limitation:**1) The GUI however is very primitive and is inadequate for instructional purposes. It is not fault-tolerant to novice users. An incorrect sequence of keys or mouse clicks could cause the simulation to crash; 2) Its uses are limited.

MaRS (Maryland Routing Simulator) [23, 24] is a discrete-event simulator proving a flexible platform for the evaluation and comparison of network routing algorithms. MaRS allows physical network, routing algorithm and traffic sources. MaRS is structured in two parts: a simulation engine, which manages the event list and user interface and a set of components for modelling the network configuration and handling certain simulation functions. MaRS has been used to evaluate and compare several next-hop routing algorithms: two distance-vector algorithms- a) Loop-free Bellman-Ford Routing Protocol Without Bouncing Effect, a Failsafe Distributed routing Protocol; b) a link-state algorithm (the New Routing Algorithm) for ARPANET. MaRS is limited due to the lack of scrolling capability in the user interface window. Thus, users are limited in what can be displayed to the size of the non-scrolling window. In addition, MaRS constrains itself to a point-to-point network structure and therefore is not readily suitable for simulating networks with broadcast communication. **Contributor:** University of Maryland. **Supports**: network routing protocols (SPF, Merlin_Segall, Bellman_Ford), FTP, Telnet, Workload (Poison, uniform). **Languages:** C. **Limitation:** 1) Its support for transport layer protocol and application source models is very limited; 2) a primitive GUI is provided; 3) not available now.

pdns (Parallel/Distributed ns) [25] simulator consists of extensions to the widely used and publicly available ns network simulator. The extensions allow many existing ns simulations to be run in a distributed environment with minimal changes. The pdns implementation also takes advantages of the large body of existing models found in ns and uses those without modification. Also includes in the pdns extensions is novel packet routing method called NIx-vector that allow routing decisions without the necessity of routing tables, resulting in substantial memory saving. While distributing the ns tool to several processors, for connecting each sub-model, IP address and a network mask is added to physical end point (nodes) for communication in logic as well as physical connection. For routing, each simulated node will start with routine table priori and dynamic routing information in the simulation time using Boarder Gateway Protocol (BGP) to adapt to any change in the simulated network topology. For event time management and event distribution among the parallel simulation sub-models, RTIKIY and LBTS (lower bound time-stamp) etc. time management run-time library are used. For synchronous event communication, multicast group management (MCAST) strategy is used. **Contributor:** The PADS research group at Georgia Institute of Technology. **Advantages:** 1) Very large-scale simulator (hundreds of thousands of nodes); 2) IP Addresses are supported.

The **GTNetS** (Georgian Tech Network Simulator) [19, 26] was developed by George F. Riley using C++ language. GTNetS has a number of basic design goals, which is categorized into seven high-level goals. The GTNetS is a full featured network simulation environment that can be used for experimental networking research on moderate to large-scale topologies. The design is such that it is easy to learn and use. GTNetS is presently fully capable of distributed large-scale simulations of routers, end-systems, LAN's and various end-user applications. Researchers find the tool useful in instance where existing tool can not be achieve the scale needed to work functions being studied or can not achieve the scale needed to produce the desired results.

The object-oriented methodology is used to design the tool for easy extension and modification. All queueing methods use a subclass of queue to define their behavior. The simulator is designed like real networks are designed. In GTNetS, there is a clear distinct between nodes, interfaces, links, and protocols. Node objects represent the basic functionality of a network (either a router or end-user system), and contain one or more interface objects. Each interface objects has an IP address and associated network mask, as well as a link object encapsulating the behavior of the transmission medium. Packets in GTNet*S* consist of a list of protocol data objects (PDUs). This list is created and extended while a packet moves down the protocol stack the various layers. When moving up the stack, each protocol layer removes and processes the corresponding protocol header in a fashion closely modeling a real protocol stack. Each protocol layer communicates with the layers below it by invoking a DataRequest method, specifying the packet and any protocol specific information required by the next lower layer. GTNetS used NIx-Vector routing as the default packet routing mechanism.

Simulation models for a number of different random number generators are provided, including exponential, Pareto, normal, uniform, empirical and constant. GTNetS allows the specification of default object types wherever practical. For nearly every objects (links, queues, protocols, etc), a default value is provided that allows for creation of object without specifying details. It provides a number of stock objects for creating well-known topologies such as star, dumbbell, grid and tree. GTNetS includes simulation models of a number of popular protocols at the application layer, transport layers, network layer and link layer (IEEE 802-3 for wired networks, IEEE 802.11 for wireless). The application layer models included in GTNetS are: Gnutello, GCache, Syn-Flood and UDP storm. The transport protocols are: TCP, TCP Reno, TCP NewReno, TCP Tahoe and TCP SACK layer, UDP. Additionally, the design of TCP model uses a client/server paradigm. Data contents as well as length can be model within this tool. The tool uses IPV4 exclusively for the network layer protocol. DropTail and Random Early Detection (RED) queues are implemented. Nodes with wireless interfaces also have mobility models and support random initial node placements.

The GTNetS simulator consists of a large number of C++ objects, which implement the behavior of a variety of network elements. Building and running a simulator using GTNetS consisting of creating a C++ main program that instantiates the various network elements to describe the network topology and the various applications and protocols used to move simulated data through the topology. The C++ main program is then compiled with any compiler that fully compiles with the C++ standards. After the successfully compiling the main program, it make is linked with the GTNetS object libraries. The resulting executable binary is simply executed as any other application, which results in the simulation of the topology and data flows specified in the main program.

Finally, GTNetS keeps and optionally reports detailed statistics about the simulator's performance. These statistics include the number of objects created, number of simulation events, memory used, just to name a few, which assist the simulator user in identifying resource limitations or performance problems should they occur. The tool has a number of data summation primitives in gathering performance statistics. Histogram(e.g., sequence number of packet vs. time in a TCP connection) can be plotted for data sets. The histogram objects can then be queried and printed. Packet tracing in the simulation can be performed for analysis. The trace file is saved in text format (.txt).

The tool runs both in windows and Unix in C++ standard and freely available. Input file(c++) and output can be saved for later use. The tool can handle up to half-million nodes; up to 455,168 TCP flows and more than 4 billion simulated packet transmission events. With 480,000 nodded topology, simulation can complete within 15 minutes. Any user who has a good understanding of the design and operation of real networks will find that GTNet*S* works similarly. **Contributor:** Riley; **Year:** 2002; **Supported Operating system:** Windows, Unix; **Supported elements**: Ethernet and point to point link, Static and NixVectors routing, IPV4 models, TCP family, UDP, DropTail, RED; **Developed Languages:** C++.

WIPSIM-Development of the Wireless IP Simulator [27] was started in 2000 in the WING-group of CPK at Aalborg University. In July 2001, the framework of the simulator was redesigned to allow for easy addition of protocols and mechanisms by multiple developers. Several reasons to start writing a new simulator instead of using an existing tool: No (affordable) simulator available that could fulfill the needs of the WING-group; Increased learning effect by implementing protocols from scratch; Create tool that could aid in education. WIPSIM is an event-driven simulator, written in ANSI-C++ and is open source. Input and output are handled using text-files: Scenario-file: movement of the nodes and the traffic in the network; Network-file: nodes, interface, protocols and their parameters; Config-file: default parameters for protocols, aliases

The framework of the simulator connects the different layers of the protocol stack. Each part of the framework is implemented as a base-class, which provides the interface for access to other parts of the simulator and to other layers. Each specific protocol implementation can then be derived from this base-class.

At the physical layer, interfaces of nodes are either in-range or out-of-range of each other. Currently, no detailed radio propagation models are used, a frame transmitted at the physical layer either reaches other nodes (if the interface of that node is in-range) or not (if the interface is not in-range). Optionally, a frame error rate can be set. The radio propagation is kept this simple to reduce simulation overhead and because the focus of the simulator is the IP and MAC layer.

Movement of nodes is described in the scenario-file, where event are listed where nodes move in- and out-of-range of each other. Any tool can be used to provide the input for the movement of nodes to the scenario-file based on the desired mobility model. As such, mobility is pre-determined before the simulation starts. This allows for comparison of different protocols under the exact same mobility conditions. Moreover, it reduces the amount of calculations that needs to be done in case a number of simulations are carried out with the same mobility. **Contributor:** WING-group of CPK at Aalborg University; **Supported elements**: MAC, IP layer, cd hoc rouing protocols and connectivily, node mobility, AODV, OLSR, Multipath-aodv, simple shortest path algorithm (e.g. Dijkstra), Diddserv, CBR UDP model, MPEG-4 video source model. **Developed Languages:** C++. **Extensibility:** easily extendible. **Outputs:** calculates average packet delays, throughput, packet delivery ratios etc. **Advantages:** Development knowledge available within CNTK; Simulator + source code freely available; Design of the general framework of the simulator makes it easy to add new protocols/mechanisms; Source code is well documented. **Limitation:** User/developer base is still small, therefore there is some uncertainty about the correctness of the implementation of protocols; the number of implemented protocols is not so large yet; Development and user documentation is still under development

B Commercial Tools

OPNET (Optimised Network Engineering tool) [19, 20, 28, 29] was launched in 1987 as a first commercial available simulation tool for communication networks. It provides a comprehensive development environment for the specification, simulation and performance analysis of communication network. It can simulate all kind of wired networks and a 802.11 compliant MAC layer implementation is also provided. A large number of communication systems from a single LAN to global satellite networks can be supported. The most important features of OPNET are: modelling and simulation cycle (to assist user to go through three phases in design circle-building model, execution of simulation and the analysis of output), hierarchical modelling (describes different aspects of the complete model being simulated), specialised in communication networks (support for existing protocols and allow users to either modify theses existing models or develop new models), automatic simulation generation. An OPNET model consists of three layers: the network models, the node model and the process model. The tool allows the different layers in a protocol stack running on individual nodes to be represented. OPNET contains various tools for data collection. A probe editor allows specifying: 1) which statistics are to be collected from where (Probe editor); 2) their own statistics (analyse tool); 3) animation view and formats (Filter Tool, Animation viewer). OPNET contains objects that are capable of generation vast amount of output data during simulation. It can generate error rate and throughputs, delay queue size. Packet trace may be done. Output can be plotted in graph, such as end-to end delay vs. queue buffer capacity, loss ratio vs. queue buffer capacity. Probability distribution function, cumulative distribution function as well as histogram can be plotted for several data sets. OPNET is extensively used for the study of TCP transport across different types of ATM bearer capabilities and diffserv per hop behaviour. **Contributor:** Mil 3 Inc. **Supports**: routing protocols (OSPF, RIP, EIGRP, BGP, IGRP, DSR, TORA IS-IS, PNNI), Diffserv, MAC, mobility of nodes, ad hoc connectivity, different application models. Node failure models, modelling of power-consumptiion etc. **Languages:** C**,** Java. **Advantages:** 1) extendible; 2) large customer base; 3) professional support; 4) very well documented; 5) ships with a large number of built-in protocols. **Limitation:** 1) relatively high price; 2) complex, takes time to learn; 3) there is restriction to its portability.

COMNET III [30, 31, 32] is a commercial integrated discrete event object-oriented simulator for modelling and performance analysis of computer network. With COMNET III, users can create a variety of network architectures, including LANs, MANs WANs, packet switching, ATM, frame relay and so on. COMNET is entirely driven by graphical user interface. Users are able to graphically display traffic modelling patterns, use of realistic network objects to reflect real networks and apply the network concepts. COMNET III allows the users to model, tune and analyse the performance of various types of networks. It provides an extensive library of nodes, links, protocols and traffic objects. The library of link objects include two classes: point to point links and multi-access links. Multi-access protocols that can be modelled include CSMA/CD, CSMA, ALOHA, Token BUS, Token Ring, FDDI and Polling. It supports subnets. It has two kinds of sources: application sources and traffic sources. There are four types of traffic generators: message sources, session sources, response sources and call sources. COMNET III is able to produce about 100 reports for different model building blocks. Most commonly produced reports include node utilisation and application delays, link delays, channel utilisation, message delays, packet delays, calls blocked, disconnected and preempted, session set up delay, collusion statistics, token ring statistics, buffer statistics etc. **Contributor:** CACI Company. **Supports**: LAN (Ethernet, Token ring, FDDI), MAN, internet, packet switch, circuit switch, ATM, bursty traffic, CSMA, CSMA/CD, ALOHA, Polling, Protocols (TCP/IP, IPX, SNA, DECNet). **Advantages:** 1) a graphical package, allow quickly and easily analyse and predict the performance of network; 2) allows users the flexibility to try an unlimited number of "what if" scenarios; 3) realistic and accurate results. **Limitation:** 1) source code is not available; 2) new modules were very difficult to add; 3) It is restricted to experimenting with the set of networking protocols provided by the package; 4) It is not programmable by the users.

REAL (REalistic And Large) [33] is a network simulator originally intended for studying the dynamic behaviour of flow and congestion control schemes in packet-switched data networks. It provides around 30 modules that exactly emulate the actions of several well-known flow control protocols (such as TCP), and 5 research scheduling disciplines (such as Fair Queuing and Hierarchical Round Robin). There are nearly 30 source types, corresponding to 30 or so transport protocol and workload types. The sources can be categorised into one of two types: flow-controlled and non-flow-controlled data sources. REAL has a graphical user interface (GUI) written in Java. The simulator takes as input a network *scenario (*a description of network topology, protocols, workload and control parameters) described in NetLanguage (ASCII). It supports sources or sinks node, gateways (synonymously with routers, bridges and switches). Users have to specify some network parameters, the transport protocol (in particular, the flow control) and the workload at each source. Finally, users must specify control parameters such as the latency and bandwidth of each communication line, the size of trunk board buffers, packet sizes etc. **Contributor:** S. Keshav at Cornell University. **Supports**: TCP/IP, XNS, FTP, Telnet, ill behaved, FIFO, FCFS, Fair Queuing (FQ), DEC congestion avoidance and Hierarchical Round Robin. **Languages:** C, JAVA (GUI only). **Advantages:** 1) the GUI allows users to quickly build simulation scenarios with a point-and-click interface; 2) source code is provided so that interested users can modify the simulator to their own purposes; 3) extendible. **Limitation:** 1) NEST (initial tool) didn't not allow for timers, REAL sends out a timer packet from a source back to itself to return after some specified time, but timers cannot be reset using this method; 2) misses some of the behaviours in the common protocol implementation; 3) do not support common implementation of protocols (direct execution). Instead it uses codes that simulates the major characterstisc of the protols.

SSF (Scalable Simulation Framework) [34, 35] is designed to model very large-scale networks, which are described in generic modelling language called *Domain Modelling language (DML).* SSF can be run in a distributed environment (called DaSSE) on a tightly coupled shared-memory symmetric multiprocessor. DaSSF achieves good parallel performance by using a periodic time-stepped approach. All processors can safely process messages between synchronisation cycles without fear of erroneous results due to unsafe events. SSF supports multicast in-channel (many to one communication) as well as multicast out-channel (one to many) and bus-style channel mapping (many to many). SSFNET is the first collection of SF-based models for simulating Internet protocol and networks. The SSFNET libraries include component models for network elements (hosts, routers, network interface card, local area networks) and network protocols (currently IP, UDP, TCP, BGP and OSPF). SSFNET models are self-configuring, that is each SSFNET class instance can autonomously configure itself. SSF has been demonstrated on networks of several hundred thousand nodes. **Contributor**: Renesys Corporation. **Supports**: TCP, UDP, BGP protocols. **Languages:** Java, C++. **Outputs:** packet-by-packet simulation. **Advantages:** 1) Open souce code; 2) Free for education; 3) Platform independentent; 4) This has open source code and based on OOP, it can be enhanced in future. **Limitation:** Dynamic simulation cannot be visualised using this tool.

TeD (Telecommunication Description Language) [36, 37, 38] is a language for describing telecommunications networks, coupled with an optimistic network simulation engine, based on Georgia Tech Time-warp. The TeD language specification is split into two distinct parts- MetaTeD and "External Language". MetaTeD defines a set of concepts for modelling the dynamic interactions of entities and their compositions. When MetaTeD is appropriately combined with any regular general-purpose programming language (say C++) then complete language is formed. TeD is process-oriented where the number of processes easily exceeds one million, warranting efficient support for large-scale process orientation. It has demonstrated good performance and scalability when modelling ATM cell switches and private network-network interface (PNNI), Internet and wireless network. TeD has been demonstrated on network models consisting of tens of thousands of nodes. Conservative and Optimistic synchronous processes have been implemented in TeD. **Contributor:** Kalyan Perumalla, Andrew Ogielski, Richard Fujimoto at Georgia Tech. **Year:** 1996. **Supports**: TCP/IP, ATM Private Network to Network Interface (PNNI) signaling neworks, Multicasting protocols, wireless networks. **Languages:** C++ and Meta-language (MetaTed). **Outputs:** Simulation of models. **Advantages:** 1) TeD itself is independent of the underlying parallel simulator and can be used with other parallel simulator (e.g. Nops, recently developed at Darmouth); 2) TeD achieves high parallel performance on multiprocessor machine, speeding up the simulation by a factor proportional to N for N-processor machine. **Limitation:**1) Limited protocols are implemented; 2) Not freely available.

USSF (Ultra-Large Simulation Framework) [39, 40] simulator is based on the WARPED simulation engine. WARPED is a parallel discrete event optimistic simulator based on Time Warp. The syntax and semantic of the input topology models for USSF are described in a *Topology Specification Language (TSL).* Using TSL, large topologies can be built from smaller sub-topologies and sub-sub-topologies. The topology is parsed into an *Intermediate Format (TSL-IF)*. The analysed TSL-IF by the help of static analyser and code generator modules is then used to generate an optimal simulatable network topology. USSF is used to model large networks using a network of Dual-CPU Pentium processors connected by an Ethernet network. Decoupling of the data and states are provided for swapping data and states in and out of the main memory based on demand, freeing up the memory for better performance of the simulation. USSF has been demonstrated on network models of hundreds of thousands of nodes. The current implementation of USSF is in C++ language in UNIX system. **Contributor:** Dhananjai Madhava Rao, Philip. A. Wilsey. **Supports**: LAN, MAN, **Languages:** C++. **Advantages:** Very large-scale simulator (hundreds of thousands of nodes).

Dummynet [41, 42] is a simple, accurate and flexible network simulation tool with minimum modification to an existing protocol stack, allowing experiments to be run on a standalone system and can be used to simulate network with arbitrary topologies. Dummynet works by intercepting communications of the protocol layer under test and simulating the effects of finite queues, bandwidths limitations and communication delays and possibly lossy links. The tool allows the use of real traffic generators and protocol implementations while solving the problem of simulating unusual environment. The running of an experiment with this tool is as easy and quick as running the desired set of applications on a workstation and as a consequence no overheads in the communication and experiments can be performed up to the maximum operating speed supported by the system in use.

The basic version of Dymmynet works at the interface between TCP and IP. The implementation takes less than 300 line of kernel code in FreeBSD. Under normal condition, there is no system overhead. The principle operation is to implement typical protocol stack where each layer communicates with the adjacent ones. In Dummynet, to simulate the presence of a network between two communicating peers the following elements are inserted in the flow of data: Router with bounded queues size and a given queueing policy; and communication links (pipes) with given bandwidth and delays.

Losses due to congestion are simulated by bounded size queues. Random packet reordering is also simulated. These show the unreliability of the networks. According to [41], a simplest setting is introduced in Dummynet, which includes one or two routers and one pipe. RED, FIFO with droptail queue policies are implemented in Dummynet. Some filtering rules are used to affect the traffic (e.g. TCP traffic to/from port, all traffic through a given interface).

The applications of Dummynet are debugging, study of new protocol and performance evaluation. Several protocols can be tested using this tool and can see the bugs and can debug the unexpected features of the implementations. This tool makes it easy to simulate unusual or hard to reproduce settings in the study of new protocols. The tool can be used to study the behaviour of existing or new congestion control mechanism in presence of bottleneck links or any asynchronous links. Performances can be evaluated for a model in specific parameter settings.

There are some limitations in Dummynet. It can only approximate the behaviour of a real system with given features. Most of the approximation introduced by the tool derived from granularity and the precision of the operating system's timer, and in many cases have little influence on the experiments. A second problem is that the periodic task might be run late or even misses one or more timer ticks. Events in the tool occur synchronous with the system's timers, which may hide or amplify some real-world phenomenon that occurs due to race condition. The current implementation of dummynet lacks any automated tool to setup an arbitrary network topologies starting from a graphical or textural description. The tool uses low-level user interface like command. More details of the commands are shown in the paper [41].

According to the paper [41], Dummynet tool can be extended to simulate complex network model and graphical output can be visualized. Trace file can be constructed to for the traffic information. Dummynet runs in FreeBSD and file is not saved and therefore the overhead introduced to the tool is almost negligible. **Contributor:** Luigi Rizzo. **Year:** Sept.1997. **Supported Operating system:** FreeBSD. **Supported elements**: TCP/IP, FTP, Telnet, Web borwsers, UDP. **Developed Languages:** C. **Extensibility:**Yes. **Outputs**: Emulates a link with fair queueing, artificial delay, etc., for testing protocols. Simulate the effects of bandwidth limitations, propagation delays, bounded-size queues, packet losses, multipath. **Advantages:** Free. Open source; Great control over operating parameters, simplicity and availability to use real traffic generators, high accuracy; almost no overhead. **Limitation:** Dummynet can only approximate the behaviours of a real system with given features; The periodic task might be run late or even miss one or more timer ticks depending on the overall system load; Events in dummynet occur synchronously with system's timer which might hide or amplify some real world phenomenona which occur because of race conditions.

Ethersim [43] is a simulation tool to model and study the performance of multimedia–oriented integrated service ATM networks with mobile hosts and wireless links. It is a discrete event base simulator core and incorporates models of user applications and transport, network and MAC layer protocols. It provides the capability to specify a cellular wireless ATM network topology and hosts and basestations.

Ethersim consists of five special entities relevant to modelling mobility and wireless communication: an air module, a map, a mover, mobile hosts, and basestations. The air module models the physical air-interface effects (e.g. RF power decay, frequency collusion etc). The map module is used to define a geographical region and the placement of various wireless entities (e.g., basestations, mobile hosts) in it and is constructed using an undirected graph structure whose node represent arbitrary geographical region referred to as room or cell. The mover is a central entity that moves the mobile hosts on the map. Ethersim allows for both random and goal–directed movements of the mobile hosts, and allows synchronous goal-directed movements to model conference room type mobility patterns. Basestation is a switch with radios, which maintains a routing table using which it routes packets coming in on import port to an output port where they are buffered. The mobile host is derived from the wired host and subsumes all its functionality with additional enhancements (e.g., mobility aware, rerouting protocols, MAC sub-module, and mobile capability with speed). The tool is constructed is a modular fashion to allow functionality at different levels of the protocol stack to be modified independently, thereby allowing network protocol designers to study the interaction between policies embedded in the protocol at different layers.

Ethersim has a rich variety of network components such as hosts, links, switches, ATM and TCP/IP protocol modules that allow the modelling of variety of mixed of wired and wireless network scenarios. Ethersim supports wireless and mobility aspects in an integral fashion and uses the network reference model consists of wired and wireless parts. The wired part is composed of switches and wired static hosts with point-to-point wired links connecting the hosts to switches ports or one switch port to another switch ports. Some switches acts as basestations having equipped with radio interfaces. Wireless mobile hosts are derived from the standard wired host by replacing the wired network link interface with a radio. The radio-equipped host can geographically moves with the users who are carrying it. Radio may have multiple channels and radio may be configured to operate on one of the channels. Channels are intrinsically broadcast oriented.

Ethersim supports mobile network protocol (mobile IP and mobile ATM), wireless links including bandwidth, large scale and small scale propagation loss, receiver sensitivity, multiplexing technique (FDM, TDM, frequency hopping, direct sequence spread spectrum etc.), medium access protocols and hand-off protocols (hard vs. soft hand off), flexible pattern of host mobility (direct and random movements of mobile users), convergence and divergence of roaming to/from a meeting location, and different application models and adaptation. While host mobbing from one basestation to another, Ethersim uses a single parameterized Connection Manager module which follows the combination of the rerouting schemes of extension, extension with loop removal, total rebuild, partial rebuild to a fixed anchor switch, partial rebuild to a dynamically selected cross-over switch, multicast to neighbouring basestations.

The main software modules of Ethersim are: wired static host, switch, wired link, basestation, wireless mobile host, air, map, mover, traffic source (statistical and trace driven), protocol modules (transport, connection establishment, and packet scheduling), measurement modules and graphical user interface modules. The tool uses a single parameterized hybrid Connection Manager module to reroute the packets while host moving.

Ethersim allows to accurately measuring the effect of the frequency hop collisions on the average throughput of applications run on the top of TCP. It has various performance measurement and graphical user interface routine to interpret the simulation results. Host mobility affects the achievable user and network performance as measured by the throughput, packet delay, link utilization levels. Ethersim allows studying the delay for rerouting the packet path as host moves, handoff and register to new basestation as policies and parameters used in MAC, network and transport layers are varied. With Ethersim, user can visualise the effects of reroute, throughput due to frequency collusion, effect of time interval length, performance impact on host mobility, rerouting policy on per-packet delay etc. Error recovery and congestion control mechanism can also be studied using the tool. **Contributor:** AT&T Bell Laboratory **Year:** 1998. **Supported Operating system:** Unix. **Supported elements**: Wired and Wireless network, LAN, WAN, hosts, links, switches, ATM and TCP/IP protocol, mobile IP and mobile ATM, multiplexing technique (FDM, TDM), CBR, VBR, ABR, UBR, MAC protocol, noise models(e.g. aussian), propagation loss, rerouting models. **Developed Languages:** Java. **Extensibility:** Yes. **Outputs:** performance (throughput, packet delay, link utilization), collusion probability, packet loss. **Advantages:** Support mixed wired and wirelss network scenerio; Several host movement policies have been implemented (random movement, goal-directed movement, group movement, mixed movement) which may be diverge or converge; Rich in models. Limitation: Multiath interference has not yet included; MAC is still limited.

BONeS (Block Oriented Network Simulator) [44] is a commercial product of Cadence Designer Systems. To use this tool, license is required. BONeS allows the user to investigate event-driven network systems at the packet level by building up systems in the form of input/output modules in a signal flow block diagram and simulate the responses at chosen probes with given starting parameters. BONeS comes with a variety of supported networking protocols for both wireless and terrestrial environments. No limitation of the number of nodes, links and agents.

There is a Motif graphical environment in BONeS simulator for capturing the design or architecture of communication networks and simulating the performance of the captured designs. The user specifies a network design by drawing a hierarchical block diagram with building blocks from the user-extendible BONeS designer model library. Modelling elements at the lowest level are called primitives, and written in C++. These blocks accept data structures as inputs, perform simple operations such as modifying fields within the data structures, and return data structures as outputs. A typical BONeS Designer session consists of four steps: Creating Data Structure; Constructing Block Diagrams; Running Simulations; Evaluating he results.

The data structures, necessary to complete the specification of BONeS designer model using a graphical data structure editor (DSE), are defined hierarchically and can have an arbitrary number of fields which can contain simple entities, such as the sequence number and the time stamps. These data structures are defined to meet the needs of the simulation and do not necessarily duplicate the actual packet structure in the network being simulated. The first step for a model design is to create a data structure.

The network model is also constructed graphically, using the BONeS designer block diagram editor (BDE). Primitives and other blocks are placed on the workstation screen and connected to form protocol functions. These functions are in turn grouped to form nodes, and nodes are then connected by communication links to form a topological network model.

Hierarchical and some form of model aggregation are used to manage the complexity of models of large networks. The user specifies parameters of the individual blocks at some points prior to simulation. Measures characteristics of traffic and communication links can easily be incorporated into BONeS models.

Once the network model definition is complete; BONeS Designer performs a variety of error and consistency checking and automatically translates the graphical model into a C++ program. An event-driven simulation of the network model is then executed, with user-specified values for model parameters. During the simulation, data structures created by some source models flow along connection lines to various processing modules, which may alter the content of the data structures and/or modify their path through the block diagram. Once the simulation has been stated, the current simulation clock time, stop time and whether error and warning messages have occurred. Eventually, data structure arrives at sink modules, where they are taken out of the system. A BONeS simulation continues until there are no more data structures in the block diagram or until the simulation clock reaches a user-specified stop time. These are performed using the simulation manager. The simulation manager also provides the capability to record the sequence of the execution of a model or to aid in debugging block diagram.

During the simulation, BONeS Designer collects data to various points in the network using a variety of probes. The user specifies the location and types of the probe. The BONeS Designer library manager takes care of the storage and retrieval of the simulation models, programs, parameter value, and simulation data. The data collected during the simulation are analysed and displayed graphically using the post processor. The PP has built-in analytical function for performing statistical operations on the data. To display results, the user typically goes through the following steps: selecting simulation; selecting probes; applying a conditional or filter to the data probe (not necessary); specifying the X- and Y-axis expressions. **Contributor:** Cadence Designer Systems. **Year:** 1998. **Supported Operating system:** Unix. **Supported elements**: LAN, CSMA,TCP/IP, Wireless protocol, ATM support. **Developed Languages:** C/C++. **Extensibility:**Extendible. **Outputs:** analyze simulation results, compute statistical and performance measures (throughput and latency) and display results in graphical plots. **Advantages:** a large library of readymade models; Interactive simulation tool for debugging and validating models. **Limitation:** Slow for large simulations;Learning Bones is really hard; Not available now a days.

The **GUTS** is a transfer level simulator to simulate wide-area network application and services. It is high level wide–area network simulator whose goal is to enable simulation or realistic Internet- scale topologies, under a range of realistic workload [45]. The tool is designed explicitly for modelling for simulating networked services at the application level, rather than at the transport level or below. The model does not use network queues. GUTS models a network as a directed graph with nodes that represent either hosts or routers and links that represent direct network connections between two nodes. Each link has two static properties-total capacity and propagation and two dynamic properties-allocated bandwidth and transfers in progress. The network model allocates bandwidth for a transfer only once, using the transient state of the network at the beginning of the transfer, and the allocation remains fixed for the transfer's duration. The services provides by GUTS include content distribution networks (CDNs), replicated Internet services, grid computing, peer-to-peer networks, a flexible workload construction system, supporting several types of client request patterns and object properties.

The tool is written in object-oriented language C++. GUTS runs (execution time) more than 2 orders of magnitude faster than ns [45]. Event rates of GUTS are lower than ns. The tool can handle few hundred of nodes. Grade et al shows in [45] that though the traffic of GUTS are more bursty than Ns-2, the mean traffic is nearly same. **Contributor:** Syam Gadde, Jeff Chase, Amin Vahdat.**Year:** 2002. **Supported elements**: TCP family, congestion control, WAN design, coarse-grain algorithm. **Developed Languages:** C++. **Outputs:** simulation of realistic Internet-scale topology (directed graph) maintaining accurate aggregate performance metrics that enable to compare the relative performance of services over a common network infrastructure. **Advantages:** Runs more than two orders of magnitude faster than ns simulator; Much low event rates (than ns); The Guts network model's asymptotic running time is less than packer- based network simulation model. **Limitation:** Limite network protocols are implemented; No network queueing model is used;

C Specialised Tools

ATM-TN (ATM Traffic and Network simulator) [46] is designed to characterise cell level network behaviour. The simulator incorporates three classes of ATM traffic source models: an aggregate Ethernet model, an MPEG model, World Wide Web transaction model and six classes of ATMswitch architectures including output buffered, shared memory buffered and cross bar switch models. The ATM-TN simulator can be used to characterise arbitrary ATM networks with dynamic multimedia traffic loads. Call set up and tear down via ATM signalling is implemented in addition to the various types of cell traffic streams generated by voice, video and data. The simulator is built on a simple, efficient simulation language called SimKit, which is capable of supporting both fast sequential and parallel execution. Parallel execution is supported using WarpKit, an optimistically synchronised kernel that is aimed at shared memory multiprocessor platforms. The main design principles of the ATM-TN are: 1) accurately mimic ATM network behaviour at the cell level for specific traffic loads; 2) create a modular extensible architecture; and 3) achieve responsible execution times for ATM networks that consist of hundreds of traffic sources. The structure of ATM-TN consists of the components: traffic models, switch models, an ATM Modelling Framework (MF), SimKit, WrapKit, OSS and the Telecom Modelling Framework (TMF). Several distinct components are to construct traffic models: input traffic sources (e.g. FTP, Telnet, Mosaic, JEPG, MPEG), ATM Adaptation Layer (AAL) for converting source data to cell packets, access control mechanisms (e.g. leaky bucket). There are three types of traffic models are used: MPEG, Ethernet and WWW. The basic switch models in ATM-TN simulator have two components: the control module and the switch fabric. Besides dynamic behaviour of ATM network, analytical outputs can be estimated which include: propagation delay, link capacity in bit/sec, error rate, work load and several parameters. **Contributor:** Telesim project led by Brian Unger at University of Calgary. **Supports**: ATM, Data sources (FTP, Telnet, Mosaic, TCP/IP), Video sources (JPEG, MPEG, video- conferencing), WWW model, Qnet model. **Languages:** C++. **Outputs:** Link utilisation, QoS, delay analysis. **Advantages:** 1) Overhead is low; 2) High performance and realistic model; 3) The amount of computation associated with processing such events is very low; 4) ATM-TN has a GUI that provides easy simulation scenario configuration, data set organisation, and control over simulation execution and report generation. **Limitation:** At present, workload is balanced manually.

Glomosim (Global Mobile System simulator) [19, 47, 48, 49] is a scalable simulation environment for wireless mobile network systems. It is designed using the parallel discrete-event simulation capability provided by PARSEC. The protocol stack includes: models for the channel, radio, MAC, network, transport, and higher layers. It supports TCP, IEEE 802.11 CSMA/CA, MAC, UDP, HTTP, FTP, CBR, Fishleye, LARScheme-1, ODMRP, WRP, DSR, MACA, Telnet, AODV, etc. protocols.

There is a visualisation tool VT designed to view and help debugging the protocols. Glomosim currently supports protocols for only wireless network. In the future, it is planned to add functionality to simulate a wired as well as a hybrid network. There are eight files related to a network model designed in Glomosim network simulator. Six input files are used to execute a network model: Configuration file, Nodes file, Mobility file (movement of the nodes and traffic), Routers file, Application file, Ber_bpsk file (Bit Error Rate). Two files are related to the output of the network model.

Table 2: Summary of the small and large scale network simulation tools

Tools	Usages	Target Network Model	Outputs	Stored Data Format	Platform
Ns-2	Educational, Research	WAN	Simulation of several protocols	Text file (ns/tcl format)	Unix
Network Workbench	Educational, research	LAN/MAN	Simulation of network, summary of interlayer statistics	Test file	Windows Unix
NetSim	Educational	LAN (Ethernet)	Simulation, histogram of packet queuing delay, offered load, overall throughput (%), actual data rate, average queuing delay, variance of queuing delay, packet transmission rate, and total number of packets.	Text file	Windows
MaRS	Educational	WAN, link-state and distance vector routing, SPF, ExBF, Segal routing	Simulation of application traffic-ftp, telnet, simple, various meters (binary, bar graph, histogram, line graphs, etc), periodic and event updated statistics, throughput, delay, jitter, dropped packets, routing load etc., no trace file, Hope count, utilisation, delay and hope-normalised delay.	Not known-require further investigation	UNIX, Windows
OPNET	Commercial	LAN Satellite, radio modelling	Simulation of several protocols, delay, utilisation etc. performance analysis.	Text file (C/C++ format)	X window
COMNET III	Commercial	LAN, MAN, WAN	Node utilisations, application delays, link delays and utilisation, message delays, packet delays, calls blocked, disconnected and pre-empted, session setup delays etc.	Graphical input/output. No text file for modelling	Unix
REAL	Commercial	LAN	Statistics such as the number of packets sent by each source of data, number of packet received by sink, the queuing delay at each queuing point, and the number of dropped and retransmitted packets.	Text file	Unix
ATM-TN	Specialised	LAN, MAN, WAN	Simulation of ATM network	Text file in Unix system.	Unix
GloMoSim	Specialised	Wireless network	Simulation of wireless network	Text file (two configuration files)	Windows Unix
Dummynet	Commercial	LAN	Simple tcp /ip protocol, Debugging protocol, study new protocol, Performance evaluation	No output file	FreeBSD
EtherSiM	Commercial	Wired and wireless network, WAN	Performance analysis (throughput, packet delay, link utilization), collusion probability, packet loss.	Further investigation needed	Unix
BoNES	Commercial, research	WAN, Wireless, ATM support	Per flow monitoring, no automatic trace generation, trace, built-in post processing tool.	Further investigation needed	Need further study
GTNets	Educational, research	LAN/WAN	Simulation as well as performance analysis, histogram of response time	Text file (C based source code)	Unix
WIPSIM	Educational	Wireless network	Simulation of all layers of the OSI model, calculation of average packet delays, throughput, packet delivery ratios etc.	Text file	OS independent
QUIPS	Specialized	Differentiated Service	Simulation of behaviour of DS.	Text file	Unix
PlanNet	Commercial	WAN, LAN	Need further study	Need further study	Need further study

All of the statistics (selected simulation events) is compiled together into a file called glomo.stat that is produced at the end of the simulation. Trace file is produced (if 'write trace' option is chosen) which can be played as many times as user wants and this is faster than real time. Basically, Glomo.stat and trace files store same set of data. But trace file can be simulated any time, which is not possible with Glomo.stat. **Supports**: TCP family, UDP, CBR, FTP, HTTP protocols, wireless. **Languages:** Java, C. **Outputs:** Dynamic output in VT tool, trace file. **Advantages:** 1) Easy configurable and fast simulation; 2) Many protocol already implemented; 3) Well documented; 4) Dynamic behaviour can be visualise using VT tool; 5) Large scale simulator; 6) Open source code; 7) can be extended for future. **Limitation:** 1) Currently wired network is not imnplemented; 2) Incomplete API.

QUIPS-II (Queen's University IP simulation-II) [50], a successor of QUIPS, is a discrete event specialised simulator used for the design and performance evaluation of *differentiated service* (Diffserv *or DS)* based on network differentiated services or DS architecture that has recently become a promising method to address QoS (Quality of Service) issues in IP networks. Instead of the peer-flow treatment in Resource reSerVation Protocol (RSVP), diffserv networks provide QoS to each packet in the traffic stream. The simulator implements both the expedited and the assured forwarding per-hop forwarding behaviour in Diffserv network, sharing many features of IETF proposal. By setting up a network model with variable parameters, simulation can be carried out to observe the Diffserv behaviours.

QUIPS II has been designed in a modular fashion using a number of building blocks including network modules, control modules, and a GUI. These building blocks interact with each other. The network modules represent physical network components, including senders, receivers, links and nodes. The control modules are used to get global parameters, read the network topology file, set parameters and collect statistics during simulation. The GUI provides a friendly interactive environment to users for setting parameters and monitoring simulation runs.

QUIPS-II decomposed a DS domain into four kinds of physical (network module) components: senders, receivers, node and link where each kind of component is implemented by a corresponding network module. A sender is a data source that generates a traffic flow which could be one of the three services classes (Premium, Assured, Best-effort) depending on the sender's configuration. When generated, each packet in the traffic is marked in its DS field by the sender. The simulator could simulate two types of traffic: CBR and bursty traffic. A receiver is a destination that consumes the packets sent by it peer sender and each receiver has only one peer sender. When a receiver receives a packet, its delay and others results are measured. A node represents a router in the physical network, which could be either an edge router or an interior router (two types of node). This module is the most important element in the simulator to implement *Diffserv*. Link is modelled as a unidirectional link, which is characterized, by its propagation delay and bandwidth.

The control modules are used for controlling (a monitor, a stopper, and some related data files) and monitoring the simulation, which have a monitor, a stopper, and some files (4 types). The monitor is used to collect and display the performance measurements (average delay, average drop rate etc.) of interest periodically. The stopper is used to specify the conditions of stopping a simulation run.

QUIPS-II provides a user-friendly GUI interface. There are five tabs in the GUI: Global parameter tab, PHB tab, Network Topology tab, Result tab and Logs tab. The Global parameter panel is used to set parameters (maximum packets sent, packet size etc), the PHB panel is for displaying the current PHB groups supported by DS domain including the parameters peak rate/target rate of traffic flow etc. The network topology panel shows the network components (sender, receiver, server, nodes and links). The result panel shows the simulation result updated periodically and finally Log panel shows the information generated during simulation run.

QUIPS-II uses the RIQ algorithm for queue management. RIQ is a modified version of RED, which is used to detect upcoming congestion and provides better network utilization. Normally to each traffic flow is 20% for the Premium, 40% for the Assured and the remaining 40% for the best effort. User can change these schemes using this tool. User will be able to realize the concept of QoS, traffic conditioning, different traffic flows. There are three types of PHBs in the Diffser architecture implemented in QUIPS-II which are: Expedited Forwarding (EF), Assured Forwarding (AF) and Best-Effort (BE) which are used to built Premium service (low loss, low latency, low jitter, assured bandwidth0, Assured service (assured minimum throughput) and Best effort (no QoS) services in IP network respectively. User can find this tool useful to see the packet drop and delay behaviour for different services with different workload.
There are four kinds of files in QUIPS II, which are: PHB files (a set of TCAS to specify the DS endpoint with characteristics), network configuration files (configuration of several topologies with parameters setting), record file (various simulation result) and log files (record of the execution logs of simulation runs). These files are useful for later analysis. QUIPS-II runs in UNIX environment. The tool is developed using JAVA programming language. **Contributor:** Queens University. **Supported Operating system:** Unix. **Supported elements**: RSVP, BE, QoS, AF, RIQ, DS, CBR, bursty traffic. **Developed Languages:** JAVA. **Extensibility:** Extensible.**Outputs:** performance measurements (average delay, average drop rate etc.).

Table 2 and Table 3 show the summary of small and large scale simulators, and very large scale simulators respectly.

V. NETWORK TOPOLOGY GENERATION TOOLS

There are not too many works which are focused on the overview of topology generation tools and if they do, it mostly relates to the algorithms on which tools are based (see e.g. [51, 52]). It appears that there are the following historical periods in development of topology generators and their use for Internet research:

- Before 1999 when there was a strong belief that Internet is hierarchical [52] (Waxman algorithm, tools Tiers, Transit-Stub);

Table 3: Summary of the very large-scale network simulation tools

Tool	Usages	Description	Simulation resolutions	Scale of operation	Stored Data Format	Platform
Pdns	Educational, Network research	Extension to the ns simulator. Works in conjunction with Georgia's RTIKIT to process events in a correct timestamp(LBTS) order in the case of distributed simulation. The topology is modelled using (GT-ITM).	Packet level	Hundred of thousands	Text file(tcl)	Unix
TeD	Commercial, Network research	Tele-communication networks, TeD (telecommunication Descriptive Language) is a tool that brings automated parallelisation of network simulation by transforming its models into functionally equivalent GTW (Georgia Time-Warp). C++ classes in ns become entities in TeD but ns packet class is implemented as an event. On the negative side, it acknowledges the cost of 'state-saving' as the most serious among Time Warp overheads.	Packet level	Tens of thousand	TED language (text)	Sun Solaris
SSF	Commercial, Network research	Large-scale networks described in domain modelling language (DML), SSF is a discrete event modelling API designed for very large networks and can execute a million or more concurrent TCP/IP flows. SSFNET models are self-configuring and configuring data is hierarchical structured. SSF architecture has just five generic primary classes.	IP packet	Few hundred thousand	DML(domain modelling language) configuration file	Linux, Sun Solaris, , Unix, Windows
USSF	Commercial, research	To simulate complex model with over 1 million components, USSF is a framework that runs as an application on an underlying parallel kernel and utilising its services. The kernels include the WARPED based on optimistic PDES and NOTIME (an asynchronous PDES kernel). RTEL was developed to reduce the static size of the application modules and in turn its static memory requirements. It is an ideal candidate for simulating large applications that contain LPs of common description.	LP (message passing)	Few hundred thousand	TSL (topology specification language) format(text)	Linux
SWiMNet	Commercial	Wireless, mobile network. Simulation of MH, BS and BSC concepts. Call blocking/dropping, channel utilization, quality of services etc.	Predefined PCS model	Few hundred thousand	Further investigation needed	Linux
GUTS	Commercial Research toward coarser gained network models	A high level wide area network simulator designed to enable simulation or internet-scale topologies under a range of realistic work loads. No network queues are used in the model and model allocates the bandwidth to a particular transfer only once and it remains fixed during the transmission of that transfer-block. The simulator tries to simulate by using bulk-transfers as traffic composed in the backbone is of bursty nature.	No network queues transfer level burst message	Few hundred nodes	Need further study	Need further study

- 1999-2001 after it was discovered [51-53] that the Internet's degree distribution is a power law and most of the work was focused on producing and simulating such topologies (see e.g. [49-51]);

- Since 2001 [42] when attention was shifted again from local properties well represented by degree distributions towards large-scale properties which naturally are better represented by hierarchical generators.

Most of these tools found are discussed briefly in this section.

Waxman [54] is one of the first topology generators which produces random graphs based on the Erdos-Renyi random graph model, but it includes network specific characteristics such as placing the node on a *plane* and using a *probability function* to interconnect two nodes in the Waxman model that is parameterized by the distance that separates them in the plan.

Tiers [51, 55] is a multi-tier network topology generator that implements models trying to imitate the structure of the Internet. The generated model of Tiers is based on a three-level hierarchical structure aimed reproducing the differentiating between WANs, MANs and LANs comprising the Internet. To generate a random topology using Tiers, one specifies a target number of LANs and MANs. Currently Tiers cannot generate more than one WAN per random topology. For each level of hierarchy, one also specifies a fixed number of nodes per network. A *minimum spanning tree* is computed to connect all edges, then other edges are created based on user-specified average inter-level and intra-level redundancy. Edge formation favors close-by nodes, resulting in topologies with large diameters. Tiers is written in C++.

Transit-stub (TS) [56] is a package for generating and analyzing graph models of internetworks. According to the edge count, the Transit-Stub model produces the connected sub-graphs by repeatedly generating graph and checking the graph for connectivity and unconnected graph are cancelled. This method ensures that the resulting sub-graph is from all possible random graphs. Several types of information are related to nodes and edges for the augmentation of the basic topology (e.g. label (string) of node for properties of node, an identifiers of each node for indicating the stub or domain, global identifier for the belonging domain, a domain-local identifier). Each edge has a routing policy (shortest path) weight that can be used to find routes that follow the standard domain-based routing. TS model does not currently support representation of host systems. The TS generation software is written in C language.

GT-ITM [57, 58] is a popular topology generator that produces topologies based on several different models. The GT-ITM topology generator can be used to create flat random graphs and two types of hierarchical graphs, the N-level and transit-stub. The main characteristics of GT-ITM are that it provides the Transit-Stub (TS) model that focuses on reproducing the hierarchical structure of the topology of the Internet. In the TS model, a connected random graph is first generated. Each node in that graph represents an entire *Transit domain*. Each transit domain node is expanded to form another connected random graph, representing the backbone topology of the transit domain. Next, for each node in each transit domain, a number of random graphs are generated representing stub domains that are attached to that node. Finally, some extra connectivity is added, in the form 'back-door' links between pairs of nodes, where a pair of nodes consists of a node from a transit domain and another from a stub domain or one node from each of two different stub domains. GT-ITM also includes five flavours of flat random graphs.

Inet [59] and **PLRG** [60] are two generators aimed at reproducing the connectivity properties of Internet topologies. These generators initially assign nodes degree from a power-law distribution and then proceed to interconnect them using different rules, Inet first determined whether the resulting typologies will be connected, forms a *spanning tree* using nodes if degree greater than two, attaches nodes with degree one to the spanning tree and then match the remaining unfulfilled degrees of all nodes with each other. PLRG works similar to Inet in that it takes as an argument the numbers of the nodes to be generated and exponent value of alpha. This exponent value is the parameter n power law distribution which is used to assign a prior degree to the nodes of the topology.

BRITE [58, 61, 62] is a generator based on the AS power-laws. Furthermore, BRITE also incorporates recent findings on the origin of power-laws and observations of *skewed network* placement and *locality in network connections* on the Internet. By studying a number of existing topology generators, the authors of BRITE claim that the preferential connectivity and incremental growth are the primary reasons for power-laws on the Internet. For completeness, topologies are generated that incorporate both skewed node placement and locality in network connections as well as topologies with just incremental growth and preferential connectivity. To generate a topology on a plane, the plane is first divided into HSxHS squares, then the number of nodes in each square is assigned according to the node placement (NP) which is either a uniform random distribution or a bounded Pareto distribution. The bounded Pareto distribution gives a skewed node placement where a non-negligible number of squares have a large number of nodes in them. Each square is further divided into LSxLS smaller squares and the assigned nodes are then uniformly distributed among the smaller squares. A backbone node is selected from each of the top-level squares populated with nodes and a spanning tree is formed among the backbone nodes. Nodes are then connected one at a time to nodes that are already connected to the backbone. A new node can have preferential connectivity in its choice of neighboring nodes: locality-based, outdegree-based or both. The locality-based preferential connectivity uses a *Waxman probability function* to connect nodes in the topology. In outdegree-based preferential connectivity, the probability of a new node connecting to an existing node is the ratio of the existing node's outdegree over the sum of all outdegrees of nodes in the connected network. Finally, when mixing both locality based and outdegree-based preferential connectivity, the probability of connecting to an existing node under outdegree-based preferential connectivity is weighted by the Waxman probability between the new node and the existing node. Each new node introduces new links.

KOM ScenGen [63] is a topology generator that supports the manual and automatic creation of experimentation scenarios for network research from the topology creation over traffic generation to evaluation. The scenario includes all parameters needed for the simulation and experiment, e.g. topology, link and node properties, traffic mix, parameters, measurement points etc. In ScenGen, in the first step, a topology is created manually or automatically. Then the properties of the links and nodes (e.g. capacity, queuing algorithm) are set manually or automatically. Also the traffic parameters for the scenario have to be set. Next the network load which is the traffic of all nodes is created. This step can be followed by a plausibility check where several things critical for the scenario can be checked for plausibility. An example would be estimating the capacity necessary for the generated traffic and comparing it with the available capacity. If much more bandwidth is needed than offered, the operator might want to change the scenario parameters. After the plausibility check the scenario is exported to ns-2 for simulation and/or to a collection of scripts and configuration files that are used to setup the scenario in a

Table 4: Network Topology Generation Tools.

Tools	Scale of topology	OS	Generated Topology	Implemented Languages	Type of Output
Waxman	Large	Unix	Random graph	C	Text file
Tiers	Large	Unix	Three level-hierarchy model for LAN, MAN, WAN	C++	Text file
Transit-Stub	Large	Unix	Random graph	C	Text file
GT-ITM	Very Large	Unix	Transit-stub model	C	Text file
Inet	Very large	Unix	Internet, Spanning tree	C	Text file
PLRG	Large	Unix	Spanning tree	C	Text File
BRITE	Very Large	Unix	Seven types of model	C++, Java	Text file
KOM ScenGen	Large	Windows, Unix	Network Scenario (topology and traffic)	Java	Text File

testbed. The next step is to manually adapt the ns-2 files or the scripts and configuration files for specific needs. After that the simulation or experiment can be conducted and in the last step be evaluated. In this tool, there are several traffic models, sink models, load generators, traffic generators. There is a converter with ScenGen which can import the topologies generated by the tools Tiers, BRITE, GT-ITM, Inet, and NLANR. Currently, there are two export modules available in ScenGen: one for ns-2 and other for ScenGen's own testbed.

Another set of topologies for which special generators are not required are regular topologies such as the mesh, star, tree, ring, lattice, etc. These topologies have the advantages that they are very simple and are generally used for simplicity or to simulate specific scenarios such as LANs or other shared communication media.

Table 4 shows the summary of network topology generation tools.

VI. Network Discovery Tools

Currently, networks are monitored, maintained and diagnosed using discovery tools that rely on network protocols like *Internet Control Message Protocol (ICMP*) and *Simple Network Management Protocol (SNMP)*. These tools support network discovery and provide the means to remotely query and control network devices, such as routers and hosts. These discovery tools have proved to be effective in determining configuration problems and in helping the security analysis [64].

The network discovery tools are commonly used for network management because they may help to discover the nodes, links, topology, bandwidth, utilisation, operational state of links, bottlenecks and problems within cabling or routine information distributed among LANs, within the domain and in the network backbone, type of services, deployed services, traffic, infrastructure, etc. These types of information help the users/planners to map, control, maintain, monitor, and secure the network.

There are many network discovery tools available. Some of the leading discovery tools are briefly discussed in this section.

A Educational Network Discovery Tools

Fremont [65] is a network discovery tool that uses a combination of non-SNMP protocols and techniques to discover the network: watching ARP (Address Resolution protocol) packets; sending ICMP ping and netmask requests and using traceroute; watching RIP (Routing Information Protocol) packets between routers; and reading DNS reverse-lookup information and using similar naming-conversion heuristics as Scotty to locate multi-homed machines. The use of so many techniques to discover the network has the advantage of increasing the accuracy of the discovery. On the contrary, it relies on non-standard heuristics, not to mention the amount of work required to properly implement and coordinate the multitude of the protocols.

Scotty [66] is built upon a custom Tcl-based API. The network discovery tool itself is called tkined and .tkined uses ICMP (ping, traceroute, netmask request) and DNS heuristic to discover a network. The advantage of this method is that the protocols are generally well supported. The disadvantages are that accuracy can suffer, since the heuristics are based on common practice in use, not on well-defined standards.

NetMap-there are different discovery tools with the same name "NetMap". As described in [67], NetMap is an attempt to solve the problem of mapping out the interconnections of networks and machines. For maintaining network, an up-to-date map is needed of the network that shows the topology and any hardware attached. NetMap relies on a comprehensive network model that is not limited to a specific network level. NetMap uses only Internet Control Message Protocol (ICMP) and SNMPv1 for the system information. ICMP was chosen because NetMap focuses only on IP-layer detection and the features of ICMP that are virtually universally supported. NetMap can discover any network to which it has IP connectivity due to its non-reliance on protocols such as ARP. The output of the NetMap is in text format.

When all discovering is completed, Netmap will print out the network table it has constructed as well as some statistics, such as the number of machines found, the number with valid agents and so on. If a machine is found in the ping phase but does not support SNMP, it will not be added to the network map. As SNMP does not provide much the accuracy (about 50%), NetMap is not very much used in practice. NetMap is implemented in C++ on Solaris 2.5.1 platform. It uses SNMP API library to facilitate SNMP access.

Another tool also called NetMap is described in [68] with a comparison with other network discovery tools.

Big Brother [69] is a loosely-coupled distributed set of tools for monitoring and displaying the current status of an entire network and notifying network administrator if something should be done. Big Brother consists of local clients that test system conditions and the availability of *network services* and send these status reports to one or more display servers where these reports appear as little dots on a web page, or pager servers that notify administrators about system problems. The most important features of this tool are: simple testing of the network connectivity via ping; discovery of the availability of ftp, http, smtp, pop3, dns, telnet, imap, nntp, and ssh servers; local system clients monitor disk space, CPU usage, messages, and can check that important processes are up and running; support for multiple DISPLAY and PAGER servers for high-availability; warning and alarm levels are all easily redefinable; Web display can be easily modified; support for custom external tests; Integration with other systems like MRTG; many custom tests available to test things like Oracle databases; notify via e-mail, numeric pager, alphanumeric pager, or custom pager; notify based on machine name, test type, time of day, test result; delay notifications until a problem has existed for a predetermined amount of time; require notifications be acknowledged; disable repeated pages from the web display; automatic escalation should a problem exist for longer than a predetermined amount of time.

B Commercial Network Discovery Tools

LANsurveyor [70] makes it easy to map, manage, and report on entire network. LANsurveyor provides four essential functions in one cost-effective application: automatic network maps, asset management reports, network monitor, and remote administration and distribution of software. Once network nodes are discovered, LANsurveyor compiles the information into a cohesive, easy to view network map with lines representing network connectivity and each node represented with an icon. LANsurveyor generates a map of the entire network automatically using several different methods, including ICMP (ping), NetBIOS, and SNMP. Maps can be printed or exported for display or editing in any editor. LANsurveyor allows administrators to create reports that include more than 100 different pieces of information.

NetView [71] allows IP discovery, visualization, automatic update, event notification and so on. NetView performs TCP/IP discovery and displays network topologies for administrators. Also, the software manages events and SNMP traps and performs network monitoring by identifying network failure root causes and gathering trending and analytical data. It too seems use SNMP for gathering its system information. NetView uses a Web-based interface, so the application's data is easily accessible from any Web browser. NetView maintains a device inventory, easing network administrators' asset management tasks. NetView is available for a variety of platforms, including AIX, Linux, Solaris, and Windows NT/2000.

Nessus [72] is a completely new security auditing tool which aims to be an up-to-date and easy to use tool. It is a network scanner that can check for *vulnerabilities* by attempting to exploit them. This makes it more accurate, but also more heavy-handed, than other scanning tools that assume *well-known port numbers*. Nessus supports port scanning, and attacking, based on IP addresses or host name(s). It can also search through network DNS information and attack related hosts at the bequest. Tests are implemented as plug-ins, which are grouped into families, for example dealing with distributed denial of service tools. Individual plug-ins or families can be installed or not to give good control of what vulnerabilities are scanned for. Plug-ins are frequently updated to cover new vulnerabilities. Key features of Nessus are multihost. testing, multithreading, plugin support, easy-to-write plugins, easy-to-use reporting system etc. The plug-in architecture of Nessus allows users to customize it for their systems and networks. As with any scanner, Nessus is only as good as the *signature database* it relies upon. Fortunately, Nessus is frequently updated. It features full reporting, host scanning, and real-time vulnerability searches. It has a client/server architecture, the server currently runs on Linux, FreeBSD, NetBSD and Solaris, clients are available for Linux, Windows and there is a Java client. There could be false positives and false negatives, even in a tool as powerful and as frequently updated as Nessus.

Nmap [73] is a popular much more fully-featured host scanning tool that can be used to determine the topology of a network. Nmap has been available for many years and is probably the most often used tool when gathering information. It features advanced techniques such as TCP-IP fingerprinting, a method by which the returned TCP-IP packets are examined and the host OS is deduced based on various quirks present in all TCP-IP stacks. Nmap also supports a number of scanning methods from normal TCP scans (simply trying to open a connection as normal) to stealth scanning and half-open SYN scans (great for crashing unstable TCP-IP stacks). Administrators can use Nmap on a network to find host systems and open ports on those systems. Nmap is a competent first step in vulnerability assessment. User can map out all the hosts within his network and even pass an option that will allow it to attempt to identify the operating system running on a particular host. Nmap is a good foundation for establishing a policy of using secure services and stopping unused services. Nmap can be run from a shell prompt or using a graphical frontend. A shell prompt accepts the nmap command followed by the <hostname> or <IP address> of the machine user wants to scan. Nmap tests the most common network communication ports for listening or waiting services. This knowledge can be helpful to an administrator who wants to close down unnecessary services.

Table 5: Network Discovery Tools.

Tool	Usages	Description	Node Discovery	Topology Discovery	Node Management	Output
Fremont	Educational	Topology discovery tool	Yes	Yes	No	Text file
Scotty	Educational	Network management Tool	Yes	Yes	Yes	Text file
NetMap	Educational	Port scanning, network analysis tool	Yes	Yes	No	Text file
Big Brother	Educational, commercial	Network monitoring tool	No	No	Yes	Text file
LANsurveyor	Commercial	Topology discovery, Network mapping.	Yes	Yes	Yes	Text file
NetView	Commercial	Topology discovery tool	Yes	Yes	Yes	Text file
Nessus	Commercial	Vulnerability scanning tool	Yes	No	No	No text file
Nmap	Commercial	Port scanning	Yes	No	No	No text file
Open View	Commercial	Network management tool	Yes	Yes	Yes	Database
Intermapper	Commercial	Port scanning, topology discovery, network analysis tool	Yes	Yes	Yes	Text file

OpenView [74] is a Hewlet-Packard's package that covers a wide range of network and system management tasks. The tool specially covers network discovery. Based on configuration choices, it uses SNMP to gather information about network hardware, but how it determines the existence of the machines in the first place is unclear. HP OpenView is quite mature and has many partner developers. Thus, it is very comprehensive package and naturally, it is also quite expensive. In addition, due to its size, it uses a significant amount of resources and its operation has impact on the networks.

InterMapper [75] is a networking monitoring and alerting software that shows the potential network problems before end-users and customers suffer downtime or poor performance. SNMP probes discover and query elements across the distributed network - whether it spans several rooms, a building, an office park, or distributed locations. Synthetic transactions test critical applications and alert use to email, web, or directory server problems.

Table 5 shows the summary of the network discovery tools.

VII. INTEGRATION OF THE TOOLS

It would be useful, if a model could be analysed using different tools simultaneously. One tool can be complemented by other tools for their distinctive features. In this respect, there is a need for integrating analytical, simulation, topology discovery and generation tools altogether. In the integrated environment, the user/planner would have the facility to test his/her network model in different views and could consider the outputs from different angles. Understanding the strengths and weaknesses of a specific model in different design environments makes it possible to improve the design process.

If all of the network MS tools are integrated and used together then the complementary features can be applied in a network model and justified it [76]. For the proof of this concept, four tools have been integrated (e.g., Ns-2, Delite, Glomosim and Brite). As different tools are designed by different developers, it is natural that they have different purposes, nature and characteristics as well as incompatibilities. Delite is a rich tool with its topology design algorithms. It has analysis capability, such as node reliability, utilisation etc. There is a limitation of the Delite, which is that it does not support any model to be designed with more than 100 nodes. The ns-2 is a simulation tool with rich network features. Most probably, this is the most popular tool now-a-days. It supports most of the TCP/IP protocol suit. ns-2 uses nam editor as a companion tool to edit a model as well as to visualise the dynamic behaviours. User can design a model manually or write down code (tcl script) to design a network model. Anyway, ns-2 has no analysis capabilities and topology design algorithms. Brite is topology generation tool, which is used to create synthetic topology. It supports eight types of different topology models. Glomosim is a wireless simulation tool, which is becoming popular day by day. It has also rich networking features. One can design a network model using one tool and transform this model in other format and then can perform experiment and see its features. For an instance, one can design topology using Brite tool and transform it into Ns-2 format, do experiments of different protocols with the topology, then analyse its reliability, utilisation transforming into Delite model (format).

VIII. CONCLUSIONS

In this paper, a survey of the research work on network MS tools with a classification is presented. We have tried to review all the recent research work very briefly that have been brought to our attention, identified which are the main contributions that have been made, and what are the issues that seem to be open to further research.

Topology of a network or a group of networks (e.g. Internet) has a strong bearing on many management and performance issues. Investigation of the topological characteristics of computer network and its practical uses is much more accurate when right tools are applied. In this paper, a review of the research work on network topology generation and discovery tools is presented.

Network discovery tools are pretty diverse and can do tasks from purely topology discovery to network mapping, to port scanning and even vulnerability scanning. Discovering physical IP network connectivity is not easy task and despite the critical role of topology information in enhancing the manageability of modern IP networks, none of the network management platforms currently available on the market can offer a general-purpose tool for automatic discovery of physical IP network connectivity [77].

Our overall conclusion from this survey is that parallel, very large-scale network simulators are relatively new and still rapidly evolving field where Internet is trying to map. Existing research has certainly made significant contribution in this field. However rapid advances in simulation technology enable to execute highly interactive and dynamic, parallel simulations/animations on the network technology.

References:

[1] Bragg A.W. "Which network Design Tool Is Right for You?" *IT Professional*, September/October, 2000, p23-31.

[2] James, D. McCabe. *Network Analysis, Architecture and Design*. Morgan Kaufman Publishers. San Francisco, Inc. Francisco, Ca, 2003.

[3] Todd R. Andel, Alec Yasinsac. On the Credibility of Manet Simulations. *Computer Magazine, IEEE Computer Society*, July 2006, pp48-54.

[4] Stuart Kurkowski, Tracy Camp, and Michael Colagross. MANET simulation studies: the incredibles. ACM SIGMOBILE MCCR Vol. 9, Issue 4, pp. 50-61, Oct. 2005.

[5] Rahman M.A., A. Pakstas and F.Z. Wang. "Network Modelling and Simulation Tools", *Proc. of the 8th EPSRC Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and*

Broadcasting (EPSRC PGNet 2007), Liverpool John Moores University, June 28-29, 2007, Liverpool, UK.
[6] Humayun Akhtar, "An Overview of Some Network Modeling, Simulation & Performance Analysis Tools." *Proceedings of the 2nd IEEE Symposium on Computers and Communications (ISCC '*97), 1997 p344-348.
[7] Jukka K. Nurminen. *Models and Algorithms for Network Planning Tools - Practical Experiences.* System Analysis Laboratory Research Reports, Helsinki University of Technology, May 2003. (http://lib.hut.fi/Diss/2003/isbn9512265745/article6.pdf)
[8] Cahn R.S. "*Wide Area Network Design: Concepts and Tools for Optimization"*. Morgan Kaufman Publishers, San Francisco, 1989.
[9] Ma H. (Bob). "Network Design Tool. Computer Science and Engineering", *Washington University in St. Louis,* MO., 1989. (http://www.cs.wustl.edu/~javagrp/ network-design- tool.html).
[10] Bachmann D.W., M.E. Segal, M.D. Srinivasan, T. J. Teorey. "NetMod: A Design Tool for Large–Scale Heterogeneous Campus *Network". Center for Information Technology Integration (CITI)*, University of Michigan. (IEEE J. on Selected Areas in Communications. January 1991, Vol 9, No 1, 1990, p15-24)
[11] Deboo K. "XNetMod: A Design Tool for Large-Scale Networks". CITI *Technical Report TR-93-6*. University of Michigan, 1993. (http://www.citi.umich.edu/techreports/reports/citi-tr-93-6.pdf).
[12] Shahbaz M. "Network Design Tool: TND-Tool". ComNets *Annul Report.* Communication Networks, Aachen University of Technology, Germany, 1996. (http://www.comnets.rwth-aachen.de/report96/node129. html).
[13] Post. S.D. "Network Planning with a Performance-Prediction Tool". *International Journal of Network Management*, Vol 9, Issue 3, 1999, p167-173.
[14] Choi S. "Resource Configuration and Network Design in extensible Networks". *PhD Thesis*. Department of Computer Science and Engineering, Washington University, 2003. (http://www.arl.wustl.edu/~syc1/web/paper/ thesis.pdf)
[15] Ning Weng, Ramaswary Ramaswary, Tilman Wolf. Considering Processing Cost in Network Simulators. *Proc. of Workshop on Models, Methods and Tools for Reproducible Network Research (MoMeTools) in conjunction with ACM SIGCOMM*, Karlsruhe, Germany, 25 & 27 August 2003, p47-56.
[16] Vern Paxson, Sally Floyd. Why We Don't Know How To Simulate the Internet. *Proceedings of the 29th Winter Simulation Conference*, December 1997, p1037-1044.
[17] Bresla L., D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Ya Xu and H. Yu. "Advances in Network Simulation". The VINT Project. *IEEE Computer*, N5, 2000, p59-67.
[18] Fall K. and K. Varadhan. 2003. *The ns Manual.* The VINT Project, 13 December 2003. (http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf)
[19] Network Simulators, by Sally Floyd. (http://www.icir.org/models/simulators.html)
[20] Network Simulation (Wikipedia) (http://en.wikipedia.org/wiki/Network_simulation)
[21] Pullen J.M. "The Network Workbench: Network Simulation Software for Academic Investigation of Internet Concepts". *Computer Networks.* Elsevier Science Ltd, March 2000, Vol 3, Issue 3, 2000, p365-378.
[22] Barnett III B.L. "An Ethernet Simulator for Undergraduate Networking". *ACM SIGCSE Bulletin*, Vol 25, Issue 1, 1993, p145-150. (Proceedings of the 24th SIGCSE Technical Symposium on Computer Science Education. Indianapolis, Indiana, USA.)
[23] Griffin D. "Traffic Engineering for Quality of Services in the Internet, at Large Scale". *Technical Report, CEC Deliverable Number: 201/UCL/bl*, TEQUILA Consortium, 2000.
[24] Alaettinoglu C., Dussa-Zieger, I. Matta, A.U. Shankar and O. Gudmundsson. "Introducing MaRS, a Routing Testbed", *ACM SIGCOMM*, pp 95-96.
[25] Riley G.F. and M.H. Ammar. "Simulation Large Networks: How Big is Big Enough?". *Proceedings of 1ST International Conference on Grand Challenge for Modeling and Simulation*, San Antonio, TX, 2002, p39-45.
[26] G. F. Riley. The Georgia Tech Network Simulator. *Proc. of the ACM SIGCMM 2003 Workshops*, Germany, August 2003, p5-12.
[27] Wireless IP Simulator ©Copyright 2005 -OSTG Open Source. Technology Group. (http://sourceforge.net/projects/wipsim/, 2005)
[28] Chang X. 1999. "Network Simulations with OPNET". *Proceedings of the 1999 Winter Simulation Conference*, Vol 1, 1999, p307-314.
[29] OPNET Technologies, Inc. 2004. "*OPNET Modeller"*. (http://www.opnet.com/ products/modeler/home.html)
[30] Ahjua S.P. "Comnet III: A Network Simulation Laboratory Environment For A Course In Communications Networks". *Frontiers in Education Conference (IEEE/ASEE conference)*, Tempe, AZ, 1998. (http://fie.engrng.pitt.edu/fie98/papers/1205.pdf.)
[31] CACI Products Company. "COMNET III User's Manual". January 1995.
[32] Goble J.G. and R. Mills. COMNET III: Object-Oriented Network Performance Prediction. *Proceedings of the 1994 Winter Simulation Conference*, 1994, p443-445.
[33] Keshav S. "REAL 5.0 *User Manual"*. Cornell University, Ithaca NY. August 1997. (http://www.cs.cornell.edu/skeshav/real/user.html)
[34] Cowie J.H., D.M. Nicol and A.T. Ogieliski. "Modeling the Global Internte". *Computing in Science and Engineering*. Vol 1, Issue 1, 1999, p42-50.
[35] Cowie J.H., D.M. Nicol and A.T. Ogieliski. "Toward Realistic Million–Node Internet Simulations". *International Conference on Parallel and Distributed Processing Techniques and Applications* (PDPTA'99), Las Vegas, Nevada, 1999, p2129-2135.
[36]Bhatt S. and B. R. Fujimoto. "Parallel Simulation Techniques for Large-Scale Networks". *IEEE Communication Magazine*, 1998, N8, 1998, p42-49.
[37] Perumalla K., R. Fujimoto and A. Ogielski. "TeD - A Language for Modeling Telecommunication Networks". *ACM SIGMETRICS Performance Evaluation Review*, Vol 25, Issue 4, 1998, p4–11.

[38] Perumalla K.S. and R.M. Fujimoto. "Efficient Large-scale Process-oriented Parallel Simulations". *Proceedings of the 30th Winter Simulation Conference*, 1998, p459-466.

[39] Wilsey P.A. and D.M. Rao. "Simulation of Ultra-Large Communication Network". *Proceedings of 7th International Symposium on Modeling. Analysis and Simulation of Computer and Telecommunication Systems*, MASCOT'99, 1999, .p112-119.

[40] Wilsey P. A. and D. M. Rao. "An Ultra-Scale Simulation Framework". *Journal of Parallel and Distributed Computing*, January 2000, Vol 10, No 1, 2000, p18-38.

[41] Luigi Rizza. Dummynet: A Simple Approach to the Evaluation of Network Protocols. *ACM SIGCOMM, Computer Communication Review*. January 1997, Vol 27, issue 1, p31-41.

[42] Luigi Rizzo. Dummynet Tool. (Site Visited 12.9.2004). (http://info.iet.unipi.it/ ~luigi/ip_dummynet/).

[43] M. Srivastava, Partho Minshra, Prathima Agrawal, G. Nguyen. Ethersim: A Simulator for Application–Level Performance Modeling of Wireless and Mobile ATM Networks. *Computer Networks and ISDN Systems*, 1998, Vol 29, p2067-2090.

[44] David Griffin. Traffic Engineering for Quality of Services in the Internet, at Large Scale. Technical Report, *CEC Deliverable Number: 201/UCL/bl, TEQUILA Consortium*, May 2000.

[45] Syam Gadde, Jeff Chase, Amin M. Vahdat. Coarse-Grained Network Simulation for Wide-Area Distributed Systems. *Communication Network and Distributed Systems Modeling and Simulation Conference (CNDS2002)*, 27-31 January 2002. (http://www.cs.ucsd.edu/~vahdat/papers/ cnds02.pdf)

[46] Unger B. W., P. Gburzynski and C. Williamson. "A High ATM Traffic and Network Simulator". *Proceeding of the 1995 Winter Simulation Conference*, Ottawa, Ontario, 1995, p996-1003.

[47] Cavin D., Y. Sasson and A. Schiper. "On the Accuracy of MANET Simulators". *POMC'02*, Toulouse, France, 2002, p38-43.

[48] University of California, Los Angeles. 2001. "*GloMoSim Manual (ver. 1.2)*". 07 February 2001. (http://pcl.cs.ucla.edu/projects/glomosim/ GloMoSimManual.html)

[47-49] Zeng X., R. Bagrodia and M. Gerla. "GloMoSim: A Library for Parallel Simulation of Large-scale Wireless Network". *Proceedings of the 12th workshop on Parallel and Distributed Simulation*, Banff, Alberta, Canada, 1998, p154-161.

[50] H. T. Mouftah, Zesong Di. QUIPS-II: A Simulation Tool For The Design And Performance Eveluation Of Diffserv-Based Network, *Computer Communications*, 1 July, 2002, Vol 25, Issue 11-12, p1125-1131.

[51] Giuseppe Di Fatta, Giuseppe Lo Presti, Giuseppe Lo Re, "Computer Network Topologies: Models and Generation Tools", *Technical Report No. 5/2001*, University of Palermo, Italy, July 2001.

[52] H. Tangmunarunkit, R.Govindan, S.Jamin, S.Shenker, and W.Willinger, "Network Topology Generators: Degree-Based vs. Structural", *Proc. of the ACM SIGCOMM, 2002.*

[53] C.Faloutsos, P.Faloutsos, and M.Faloutsos, "On Power-Law Relationships of the Internet Topology", *Proc. of the ACM SIGCOMM, 1999*.

[54] B. M. Waxman, "Routing of Multipoint Connections", *IEEE Journal of Selected Areas in Communication*, Vol. 6, No. 9, December 1988, p1617–1622.

[55] M. Doar, "A Better Model for Generating Test Networks", *Proc. of IEEE Global Telecommunications Conference (GLOBECOM)*, London, November 1996.

[56] K. Calvert, M. Doar, and E. Zegura, "Modelling Internet Topology", *IEEE Communications Magazine*, June 1997.

[57] K. Calvert, M. Doar, E. Zegura, "Modeling Internet Topology", *IEEE Transactions on Communications*, December 1997.

[58] Topology Modeling, by Sally Floyd. (http://www.icir.org/models/topologies.html)

[59] Cheng Jin, Qian Chen, Sugih Jamin, "Inet: Internet Topology Generator", *Technical Report Research Report CSE-TR-433-00*, University of Michigan at Ann Arbor, 2000.

[60] William Aiello, Fan Chung, Linyuan Lu, "A Random Graph Model for Massive Graphs", *Proc. of the 32nd Annual Symposium on Theory of Computing*, 2000.

[61] A. Medina, A. Lakhina, I. Matta, J. Byers, "Brite: Universal topology Generator from a User's Perspective", *Technical Report, BUCS-TR-2001-003*, April 12, 2001, Boston University. (http://www.cs.bu.edu/brite/ publications/usermanual.pdf)

[62] Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John Byers, "BRITE: An Approach to Universal Topology Generation", *Proc. of MASCOTS 2001*, Cincinnati, OH, August 2001.

[63] Oliver Heckmann, Krishna Pandit, Jens Schmitt, Ralf Steinmetz, "KOM ScenGen The Swiss Army Knife for Simulation and Emulation Experiments", *First International Workshop on Multimedia Interactive Protocols and Systems (MIPS)*, Napoli, Italy, 18-21 November 2003, p91-106.

[64] Giovanni Vigna, Fredrik Valeur, Jingyu. Zhou, Richard A. Kemmerer. Composable Tool For Network Discovery and Security Analysis. 18th *Annual Computer Security Application Conference*, San Diego California, 09–13 December, 2002, p14-24. (http://www.acsac.org/2002/papers/108.pdf).

[65] Davis C. M. Wood, Sean S. Coleman, Michael F. Schwartz, "Fremont: A System for Discovering Network Characteristics and Problems." *Proc. of the USENIX Winter Conference*, San Diego, California. 25-29 January 1993, p335-348.

[66] J. Schonwalder and H. Langendorfer. Tcl Extensions for Network Management Applications. *In Proc. 3rd Tcl/Tk Workshop*, Toronto, Canada, 6-8 July 1995, p279-288.

[67] Nelson Tang, Binary Sugla, NetMap: A Network Discovery Tool. Report for Network & Service Management Research Lab, *Bell Laboratories, Lucent Technologies*, 21 September 1998. (http://www.cs.ucla.edu/~tang/papers/lucent_discovery.pdf)

[68] Giovanni Vigna, Fredrik Valeur, Jingyu. Zhou, Richard A. Kemmerer, "Composable Tool For Network Discovery and Security Analysis". *18th Annual Computer Security Application Conference*, San Diego

California, 09–13 December, 2002, p14-24. (http://www.acsac.org/2002/papers/108.pdf).

[69] Big Brother Professional Edition. Quest Software Inc., 8001 Irvine Center Drive, Irvine, CA 92618. Big Brother Homepage. (http://www.bb4.org/, 2006)

[70] LANSurveyor, Neon Software, Inc., 244 Lafayette Circle, Lafayette, CA 94549. (http://www.neon.com/, 2006)

[71] IBM Tivoli NetView discovery tool. IBM Tivoli Netview Software. Tivoli Systems Inc. (http://www-306.ibm.com/software/tivoli/ products/ netview/, 2006)

[72] Renaud Deraison. Nessus Security Scanner. The "Nessus" Project. Nessus Org. Nessus Homepage. (http://nessus.org/, 2006).

[73] Fyodor. Nmap– Security Scanner, INSECURE.ORG. (http://www.insecure.org/nmap/index.html, 2006).

[74] Hewlett-Packard Company. HP OpenView Software. (http://www.openview.hp.com, 2006)

[75] Intermapper. Dartware, LLC, 10 Buck Road, PO Box 130, Hanover, NH 03755-0130 USA. (http://www.intermapper.com/, 2006)

[76] Rahman M.A., A. Pakstas and F.Z. Wang. "An Approach to Integration of Network Design and Simulation tools", *Proc. of the 8th International IEEE Conference on Telecommunications (ConTEL 2005)*, Zagreb, Croatia, June 15-17, 2005, p173-180.

[77] Y. Breitbart, M.Garofalakis, B.Jai, C.Martin, R.Rastogi, A.Silbershatz, "Topology Discovery in Heterogenous IP Networks: The NetInventory System", *IEEE/ACM Transactions on Networking*, 2004, Vol.12, No 3, p401-414.