EVENT DRIVEN PACKET SIMULATOR

Nikolay Georgiev Chillev,Vassiliy Platonovitch Tchoumatchenko*,Tania Krumova Vassileva*

Department of Computer Science,*Department of Electronics, Technical University of Sofia
Kliment Ohridsky 8,1756, Sofia, Bulgaria, phone: +359 2 965 3731, e-mail: nikic@lark.tu-sofia.bg

The paper describes the architecture, functionality, implementation and user interface of Web based network simulator, appropriate for educational and practical use. It enables efficient configuration, simulation and experimental results visualization of real world networks, based on packet switching. Incorporating online help and check facilities as well as feedback provided to the user, make the reported network simulator a powerful tool in every e-learning environment in the field of network architecture, configuration, security and in automotive or industrial networks like CAN and USB.

Keywords: packet switching network, configuration, simulation, e-Learning

1. INTRODUCTION

Networks based on packet switching are in use for a long time, but the reveal of configuration faults, performance bottlenecks and security vulnerabilities, continues to be a hard task. It seems much more difficult, when assigned to a student who is beginner in network technologies. Network simulators with packet animation provide an economical means of network configuration and understanding. General use networks can be examined "painlessly" and inexpensively by viewing simulator's logs or packets animation. There is a set of existing network simulators. Ns is open source tool, developed in C language[5]. Nam is the network animator tool for Ns. It uses Ns's logs to animate packets. Ns has modular architecture, which facilitates the integration of user filters and plugins. It's accuracy is appropriate for sophisticated performance measurements. The drawbacks of Ns is that it has not standard GUI tool, to configure network topology and most of it's functions are not integrated, but spread in separate executables. Ns can not give feedback on user actions or mistakes, which makes it improper for educational use. Although Linux version of NS is in production for few years, it's Windows port is in beta stage and it's developers does not claim that this port is stable. AdventNet Simulation Toolkit 5 is commercial product, which comprises Simulator module(capable to simulate SNMP, TL1, TFTP, FTP, Telnet, etc. protocols), CISCO IOS Simulator(simulates CISCO routers and switches, supporting IOS application), SNMP Agent Simulator(supports simulation of SNMP device behavior). AdventNet can simulate large number of devices 50000+, it features prominent accuracy and variable model granularity[6]. This toolkit is appropriate for professionals, but is lacking features like topology comparison, useful in educational process. Another drawback is the large cost 995$ per license.

Our goal is to create interactive learning tool, which permit students to capture and edit networks and to learn given network by viewing performance statistics and packet animations. The main idea is to ensure platform independence, high level of interactivity and full control of student actions with appropriate feedback.

2.FEATURES

To address the requirements listed above we have developed web based event driven simulator, comprising the following modules:

2.1 Network editor

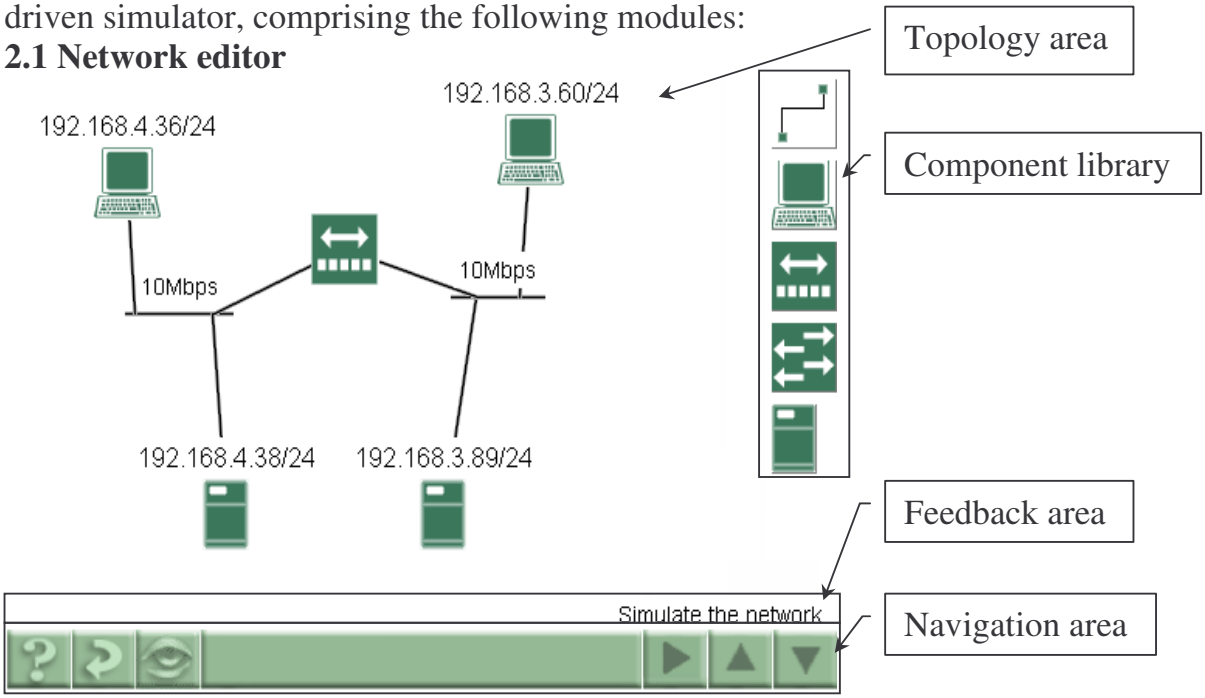
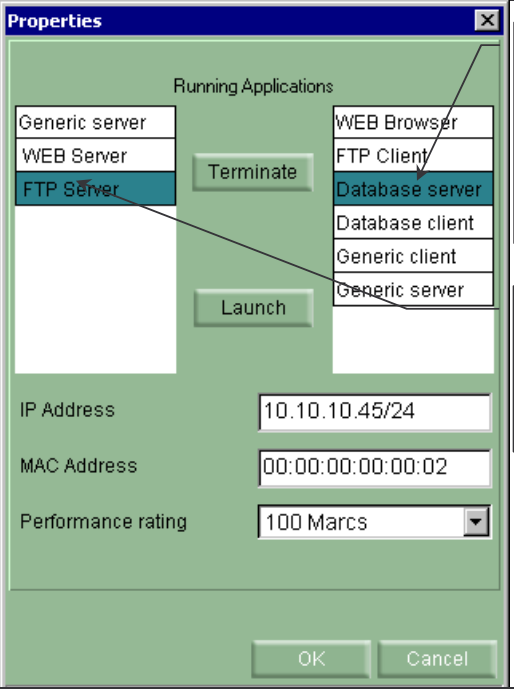


Fig. 1 The layout of topology editor

When a component from the library is selected(by clicking on it), it sticks to the mouse pointer and can be placed on any free area. Multiple components can be selected and dragged or deleted. User receives feedback messages(based on comparison between entered network and assignment) and tasks in "feedback area".



Magazine with applications, which can be started

A list of running applications

Fig.2 Host properties

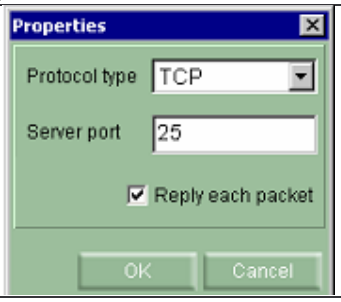


Fig.3 Generic server properties

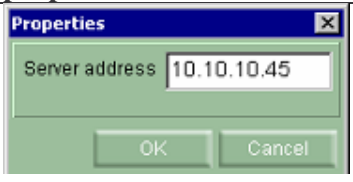


Fig.4 Client settings

With buttons from "navigation area" user can receive context sensitive help, clear the "topology area", see the solution of his assignment, trigger comparation of entered and target networks, simulate entered network or move to previous/next assignment.

2.2 Components library

Component library is flexible and can be configured with different components.Components in the library are PC Host, Hub, Switch, Server, Stateless & stateful firewalls(Fig.1), they are connected with network links. Each component is visualized with different image and has specific internal structure, modelled with common OSI layers. Each host has several properties, which can be configured by right clicking on it(Fig.2). MAC address has default value, configured by the editor. In order to enable network traffic trough this host, user must enter valid IP address and network mask. From "Performance rating" list user can select relative speed, which is useful when exploring how host speed impacts network performance.

Hosts can serve multiple applications (both servers and clients). Applications are started with Launch button and stopped with Terminate button. Most popular application like Web Server/Web Browser, FTP Server/FTP Client and Database Server/Database Client are implemented. User can simulate custom protocols, by manually tuning Generic Server and Generic Client applications. Generic servers can be configured by right clicking on them Fig.3). Protocol type can be either TCP or UDP. The port on which server is listening is also configurable, a check whether this port is not busy is performed. If "Reply each packet" checkbox is selected, server will reply each received packet. Popular clients are configured which server to inquire(Fig.4). Generic clients receive the address of server to inquire, protocol type and server port (Fig.5). Client port can be arbitrary number. The number of sent packets may affect simulation results. More packets mean more reliable statistical results, although sometimes only one packet is sufficient to reveal the problem (for example: path availability test, some security tests).

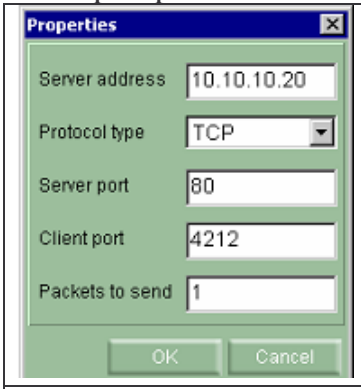


Fig.5 Generic client properties

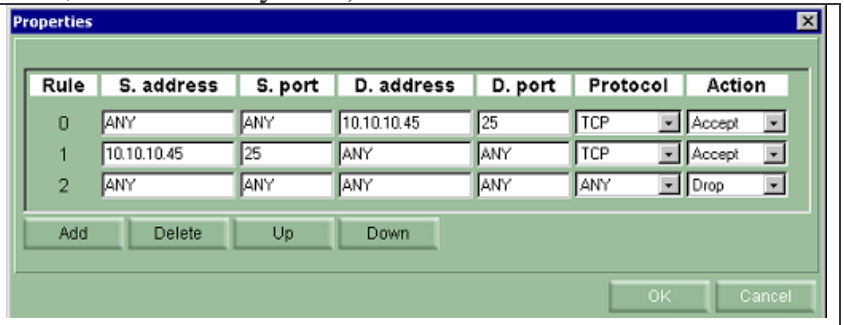


Fig.6 Firewall forwarding rules

Each firewall can be assigned a list of forwarding rules (Fig. 6). Forward rule consists of Source address, source port, destination address, destination port, protocol and action. If packet properties match rule properties, selected action will be performed. The properties of newly added rule get default values "ANY" and the action is "Drop". Source and destination addresses can get either concrete value like

10.10.10.45 or subnet address and mask like 10.10.10.0/24, which will match all packets from 10.10.10.0 subnet. Two firewall types are supported:
Stateless firewall is simple packet filter, which must have rules for both directions – request and reply. When stateful firewall receive request packet, which must be passed it memorizes source and destination addresses. When response packet is received, firewall inspects it's source and destination addresses, sees that request has already been passed and passes response.

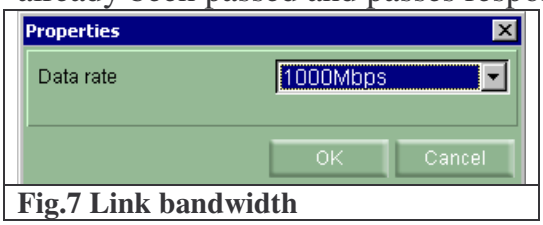


Fig.7 Link bandwidth

Network components are connected with links, which bandwidth is configurable(Fig. 7). Both bus and star topologies can be built. All parts of a bus have the same bandwidth. When a part of the bus is deleted, the whole bus is deleted. When a component is deleted, all links connected to it are also deleted.

2.3 Comparison module, simulation engine

When the student has to complete an assignment, a comparison between the entered and target networks is performed. At first the presence of all required components is checked. After that a check for extra components is performed. When components are checked, connections between the components are verified. If this check is passed, the list of started applications is checked. After all checks complete, user will be allowed to simulate network.

Simulation starts from client applications, which generate requests to the server applications. Simulation engine tracks packets, passing through the network and using harvested results it feeds reports and gives scenario to the animator. Applet has "free mode" in which network is directly simulated, which allows student to experiment with arbitrary networks.

2.4 Animation

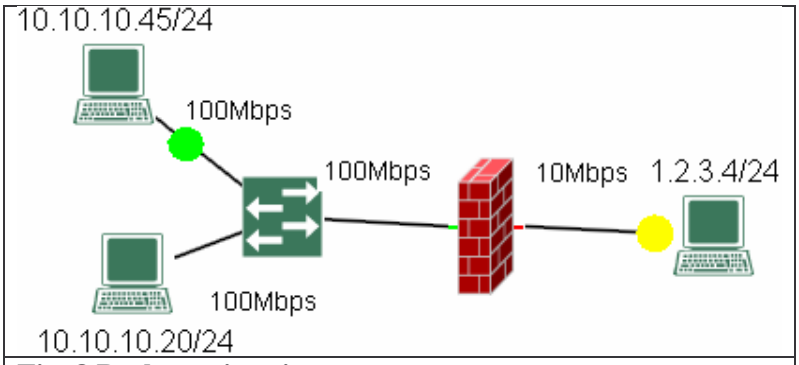


Fig. 8 Packet animation

Using data, collected during simulation, an animation of packet's lifecycle is shown(Fig. 8). If the protocol is TCP ACK packets are also animated. Packet's travel time trough network link is proportional to it's size and link's bandwidth. This is tangible for the user. Animation provides user with visual way to inspect network configuration, performance and security.

2.5 Statistics visualization

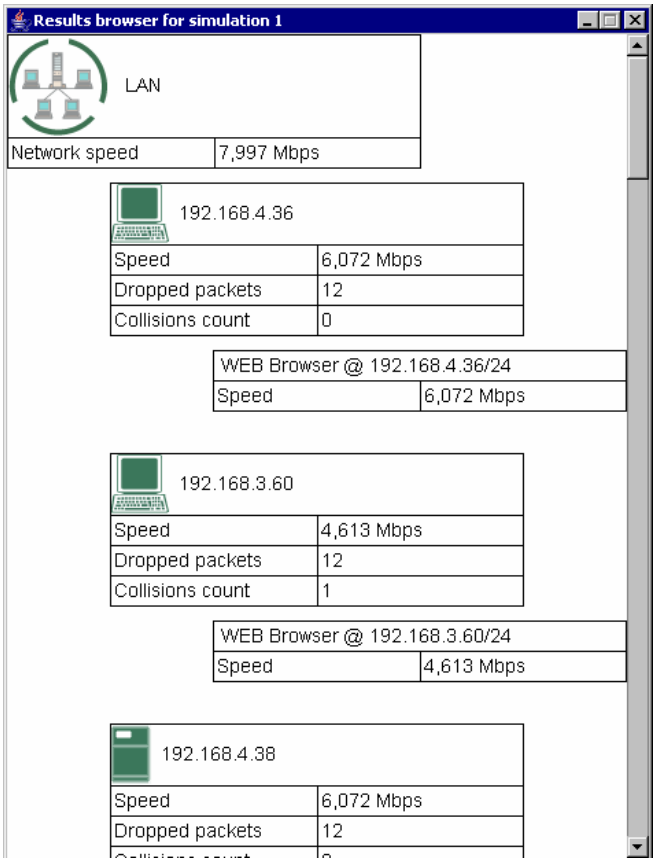


Fig.9 Statistics visualization

After the simulation completes, a performance statistics for each application, host and the whole network (calculated as sum of the subnet's speeds) is shown (Fig.9). In addition for each host is given information about the number of collisions (occur in common bus networks) and dropped packets (when host receive packet, which is not intended for him, it is dropped). The speed of a host is calculated as a sum of hosted application's speeds. Using this statistics, user can profile or debug the network.

3. IMPLEMENTATION

Java is Object Oriented and platform independent language, that makes it proper for simulator implementation. With the signed version of the simulator user can save topology to disk and later load it and simulate it. Although Java has advanced APIs for user interfaces (Swing,SWT) and parsing XML files (JDOM), older libraries like AWT are used for compatibility reasons [1]. AWT components are not customizable and can not be internationalized. But AWT provides framework for custom components. So all components are released as custom components, extending base Component class. Different designs can be applied to the applet. The layout is flexible, so the applet can fit into random layout and size. Configuration is organized in XML files, because of their structure and readability. Microstar's Aelfred, open source parser, is used for XML parsing.

Programmatically all components and links are presented as JavaBeans. To simplify development process, remove repeating code snippets and decrease bug rate we developed Java1.1 compatible framework, which edits and validates bean properties and provides bean persistence. The configuration of editor dialogs is built at applet startup and depicted in XML. In fact this framework is a lightweight implementation of some software development strategies like Aspect Oriented Programming[7], Inversion of Control and Injection[8]

The comparison of the assignment and the drawn schematic is in fact a topological comparison of the network graphs[2]. We have used the idea of "graph polymorphism" algorithm. Each component has a weight. It is formed as a hash function of it's type and properties. The developed algorithm finds equal subgraphs in

the two networks and then tries to append more components to them. At first subgraphs consist of single components. The algorithm appends new vertices to the two subgraphs only if they have same weights and links between them. Two graphs are equal, only if all components are appended to equivalent subgraphs. If the algorithm can not make a step ahead, it returns one step behind and tries combinations with another components[3]. In fact this is a backtracking algorithm, which complexity in the worst case is O(N!), where N is the number of components in the topology, but the implemented restrictions limit the number of combinations and avoid the "combinatorial explosion"[4].

Network is simulated by an implementation of event driven algorithm[5]. Hubs are modeled as electrical power amplifiers, so all components, attached to hub are assumed as attached to a common bus. Each other network component is modeled, assuming 7 layer OSI model. The most interesting layers are implemented, including physical layer (drives network links), MAC(decides whether received frame is for current host), LLC(resolves media conflicts), IP layer which adds IP addresses, Transport layer(performs TCP protocol handling (SYN and ACK packets are also simulated), handles UDP datagrams), Application layer (besides popular applications, generic server and client are implemented, allowing the user to simulate specific software). When a client application sends a request packet to server application, it is encapsulated in an event object. This object is processed by event scheduler, which estimates the delay of recipient layer and transmits it, when recipient is free. In some simple, but frequent cases, event scheduler is skipped and layers communicate directly. This increases simulator performance, at the cost of insignificant accuracy loss.

4. Conclusion

The developed event driven simulator allows network topology editing, comparison of the drawn topology and the assignment, prepared by the teacher and the simulation of TCP/IP based networks. Implemented as Java applet it is platform independent. Also it can be internationalized and has low hardware requirements. These features make it widely accessible and proper for integration in commercial courses. We are planning to extend it to be able to simulate automotive or industrial networks like CAN and USB

5. REFERENCES

[1] Eckel B., *Thinking in Java, 1998*
[2] Sendov B., *Dynamic programming, Prosveta 1985*
[3] Manev K., *Discrete mathematics, Prosveta 1997*
[4] Nakov P., *Introduction to computer algorithms, Top team books 1998*
[5] http://www.isi.edu/nsnam/ns/
[6] http://www.adventnet.com/products/simulator/index.html
[7] http://en.wikipedia.org/wiki/Aspect-oriented_programming
[8] http://www.springframework.org/