Published in Proceedings of ICCC'99, Tokyo, Japan, , Sept. 14-16

NetLobars: A Simulation System for Web System Design and Evaluation

C. Edward Chow

Department of Computer Science
University of Colorado at Colorado Springs
1420 Austin Bluffs Parkway
Colorado Springs, CO 80933-7150, USA
TEL: +1-719-262-3110 FAX: +1-719-262-3110
e-mail: chow@cs.uccs.edu
WWW: http://www.cs.uccs.edu/~chow

Jingsha He and Tomohiko Taniguchi

Network Computing Lab
Fujitsu Laboratories of America, Inc.
595 Lawrence Expressway
Sunnyvale, CA 94086-3922, USA
TEL: +1-408-530-4576 FAX: +1-408-530-4515
e-mail: jhe@fla.fujitsu.com
ttaniguc@fla.fujitsu.com

Abstract

To evaluate the web system performance, a simulation-based design system, called NetLobars was built with Java 1.2 based GUI for specifying network topology, web system configuration, and client request patterns. It simulates the shortest path dynamic routing among web servers and clients. Using the discrete event simulation, the system provides detailed analysis of various system/network delays and average end-to-end response time for comparing different web system configurations. A simulation-driven message animation based on Java 2D API is integrated for visualizing the web protocol processing. Methods for identifying the web system bottlenecks are also presented. NetLobars system facilitates web system managers and designers in evaluating web system configurations and design trade-off.

To be published in Proceedings of ICCC'99, Tokyo, Japan, Sept. 14-16.

NetLobars: A Simulation System for Web System Design and Evaluation

C. Edward Chow

Department of Computer Science
University of Colorado at Colorado Springs
1420 Austin Bluffs Parkway
Colorado Springs, CO 80933-7150
Email: chow@cs.uccs.edu
TEL: (719) 262-3110
WWW: www.cs.uccs.edu/~chow

Jingsha He and Tomohiko Taniguchi

Network Computing Lab
Fujitsu Laboratories of America, Inc.
595 Lawrence Expressway
Sunnyvale, CA 94086
jhe@fla.fujitsu.com, ttaniguc@fla.fujitsu.com
(408) 530-4576, (408) 530-4542

Abstract

To evaluate the web system performance, a simulation-based system, called NetLobars was built with Java 1.2 based GUI for specifying network topology and web system configuration, client request patterns. It simulates the shortest path dynamic routing among web servers and clients. Using the discrete event simulation, the system provides detailed analysis of various system/network delays and average end-to-end response time for comparing different web system configurations. A simulation-driven message animation based Java 2D API is integrated for visualizing the web protocol processing. Methods for identifying the web system bottlenecks are also presented.

1. Introduction

Web systems have played an essential role in providing information for wide variety of activities and are becoming important vehicles for carrying out electronic commerce. Besides the revenues of the on-line businesses, our daily life are increasingly depending on the performance of the web systems. Therefore it is a critical issue to improve the web system performance. The first task to improve the web system performance is to identify the web system bottlenecks. *Bottlenecks* are defined as system components where work in progress, information, material are being excessively delayed.

This task is complicated by the facts that today's web systems are getting more complex with additional components such as cache servers and load balancing agents. With the increase of traffic, mirroring and caching have been used to improve the performance of user response time. However, currently there is no system and standard way for guiding the selection of mirror sites. Most techniques resort to listing the server as hyper links items in the web page, or to rely on modified DNS [Katz94] to return the IP address of one of a set of server. Round Robin DNS has been used for selecting servers in LAN cluster but there are performance problems due to name caching in local or intermediate name server [Cola97]. Cisco DistributedDirector deals with geographically separate servers but maintain the centralized name resolver, which becomes the bottleneck [Cisco96]. These techniques do not provide or use server load and network path statistics to guide the dynamic selection of geographically separated replicated servers. Dynamic server selection technique proposed in [Fei97] is the first to use server status push and client network probe to guide name resolver to select web servers with better server performance. However this technique is subjected to the wide swing of congestion due to clients' rushing to the lightly loaded servers.

To accurately identify the web system bottlenecks, it is our intent to take all these additional web components and techniques into consideration.

In the proposed approach, the web system configuration and traffic pattern are first captured. The corresponding computer simulation model is then built and preliminary simulation runs will then be carried out. Based on the simulation log, the detailed server and network delay parameters will be analyzed and bottlenecks will be identified. In cases where there are no clear conclusions, additional simulation runs will be carried out as attempts to identify the bottlenecks. Finally sensitivity analysis will be performed so that the improvement over the identified bottlenecks can be presented.

There are commercial packages such as OPNET Modeler [OPNET] and COMMNET Predictor [COMMNET] for network planning, research and development based on simulation. However, they either provide restricted, predetermined set of network parameters or require extensive programming on new web service features and routing. The NetLobars system is also unique on its Java-base GUI and portability.

For existing systems, the web system configurations can be captured from the design blue prints or from the network management systems. The traffic patterns can be retrieved from the web server access logs and router logs. The NetLobars system provides a Java-based GUI for specifying the network topology, web system configurations, and client traffic patterns. It can be used to build the computer simulation model of the corresponding web systems. It is also integrated with a discrete event simulator for simulating the web system operations, network routing, and web protocol processing. The discrete event simulator takes the client traffic statistics specified by the GUI interface, generates the corresponding server request events, and computes the message processing time, the transmission delay, and propagation delay along the LAN/link segments and network devices. The simulation logs also record the document retrieving and processing times on servers. When the response from the server is received by the client, the simulator computes the end-to-end response time. At the end of the simulation run, the simulator reports the average end-to-end response time and the system throughput in terms of the number of requests processed.

An all nodes to all nodes shortest path algorithm based was implemented. It was used to compute the routing table, which is used by the network components for routing the protocol messages. The routing table can be updated according to certain frequency by running the shortest paths algorithm based on the current delays in the LAN/link segments. Therefore the NetLobars can simulate the routing protocols such as OPSF. This enables us to simulate closer to the real-life environment.

In Section 2, we give an overview of the software architecture of the NetLobars system. Section 3 discusses the message animation features of NetLboars. Section 4 discusses the basic techniques for identifying the web system bottlenecks. Section 5 is the conclusion.

2. Overview of the NetLobars System

Figure 1 shows the software architecture of the NetLobars software system. Users interact with the system through a GUI interface. The GUI then interacts with modules in the system utility block and modules in the component behavior block for realizing the user's instructions.

The Network Layout Tool provided by the GUI enables users to select network components from a menu and to click on the canvas window for its location. They can also connect components, such as servers, switches, routers, and clients with multi-access links and point-to-point links. The Network Layout Tool calls the various component behavior modules to interact with the user for more detailed information, such as client request inter-departure distribution, location, and processing speed. The resulting network configuration can be saved as a network configuration file for future retrieval.

After specifying the network configuration or reloading previous network configuration file, the user can launch network simulation through the simulation control menu. The GUI invokes the discrete event simulator, which dispatches events to component behavior modules. Those component behavior modules in turn produce and insert events in the event queues managed by the discrete event simulator. The simulation log is created for keeping the important network simulation information such as various delays and response times. To ensure the correctness of the simulation data, the percentages of the processing time in each system components are computed as a sanity check.

The simulation starts by initiating the client behavior modules for modeling the web client processes. When the client process is called, it generates the size of the HTTP-request message and a document ID with the corresponding document size distribution, and sends the HTTP-request message with the attached document ID to the network. It also calculates the schedule the next client request event and put in the event queue. The HTTP-request message size is used to calculate the transmission delay over each network segment based on its transmission speed. The propagation delay is also calculated based on the distance of the network segment and the signal propagation speed over the medium. The switching or routing speed of routers and switches are modeled so that the queueing delay on these components can also be estimated. The messages are routed to their destinations according the table generated by the shortest path algorithms.

When the message arrives at the server, the document ID attached to the HTTP-request message is used to compute the HTTP-response size and the document retrieval time. The HTTP-response size is used to calculate the transmission and switching delays on the return path.

For the cache server, the document ID is used to search the cached document directory. The size of the cache server is specified in the configuration time by the user.

If interested in observing or studying the message processing, the user can select the message animation through a menu item and then launch the simulation. In this case, the discrete event simulator will call the Message Animation Module with the source and destination of the message delivery on the current LAN or link segment. Based on the transmission speed and the distance, the message animation module calculates the animation time for the message label to be repeatedly redrawn along the edge of the link.

After a simulation run, the user can select the data analysis from a menu. The GUI will then call the Data Analysis Module to perform network analysis on the simulation log. The analysis results identify the potential bottlenecks in the system and plot the performance improvement ratio of these bottleneck components when their capacities or speeds increase.

The user can also specify the client traffic pattern and request the Resource Allocation Module to plan for the location, capacity, and speed of the network servers, switches, routers, and LAN/Links. The Resource Allocation Module takes the client traffic pattern as input, consults with the modules in the component behavior block, runs efficient resource allocation algorithms (optimal algorithm if possible). The result is a suggested network configuration, which meets certain network design rules or quality of service requirements. The Resource Allocation Module is currently being implemented.

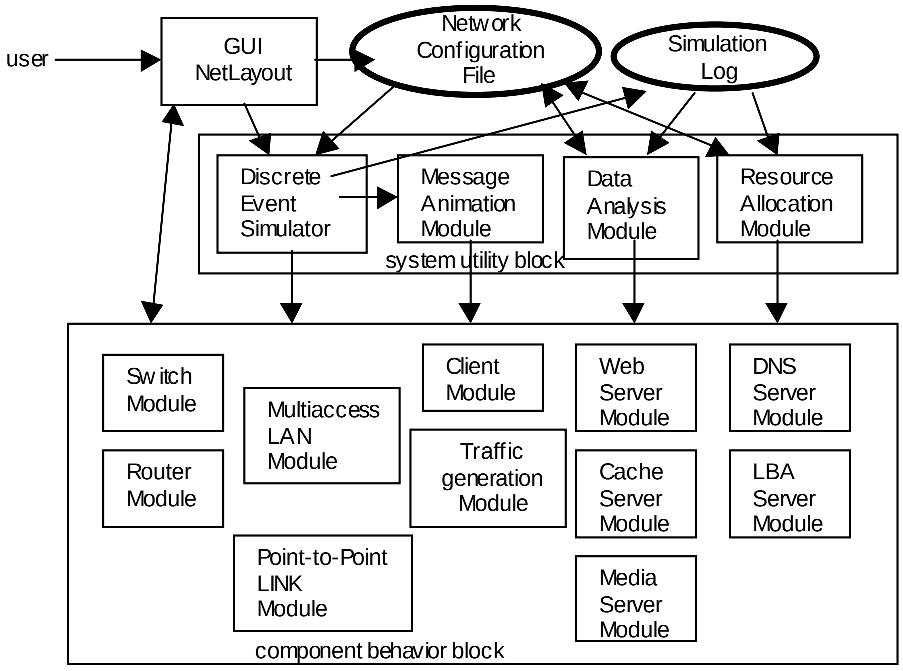
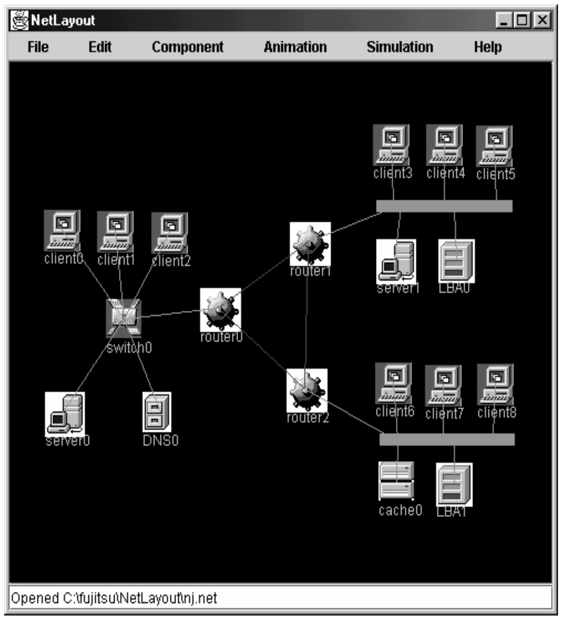


Figure 1.Software Architecture of NetLobars System.

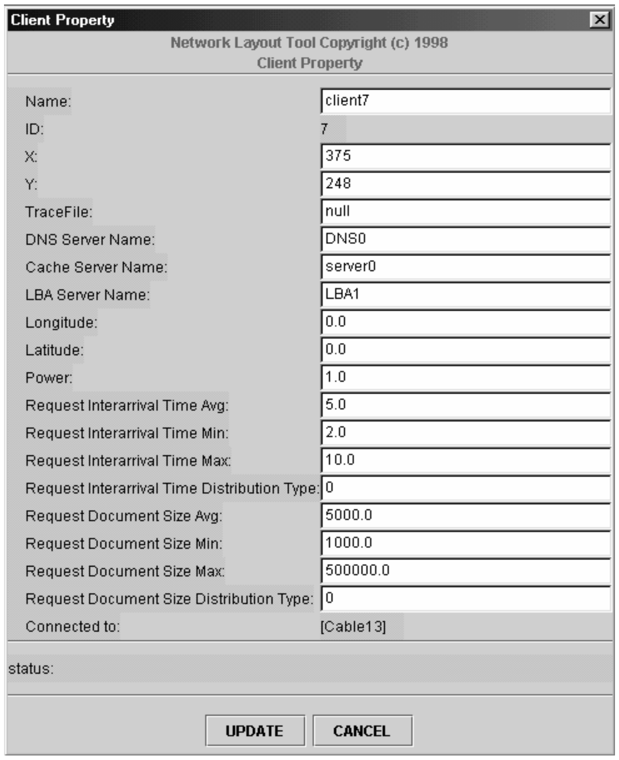


Opened C:\fujitsu\NetLayout\nj.net

Figure 2. NetLayout Tool: a Front End GUI

The current NetLobars implementation is version 0.3. It is implemented with the newest JDK 1.2 or Java 2 platform. Figure 2 show the NetLayout tool, which is the front end GUI for the NetLobars system. From the component menu, users can select one of the following web components: web server, cache server, load balancing agent, and web client, and the following network components: cable (a generic connector), multi-access link (for medium such as Ethernet), switch, router, and wide area point-to-point link. Once selected, the user can click on the canvas window and leaves an iconic symbol of the system component.

By hitting the control-shift-p, the user can open the property dialog box for specifying the system components. Figure 3 shows the property dialog box of a web client. It include the name, the X and Y screen locations, the longitude and latitude, the related servers used by the client, its processing power, and the statistics of the request inter-arrival time and the request document size.



Network Layout Tool Copyright (c) 1998
Client Property

Name:	client7
ID:	7
X:	375
Y:	248
TraceFile:	null
DNS Server Name:	DNS0
Cache Server Name:	server0
LBA Server Name:	LBA1
Longitude:	0.0
Latitude:	0.0
Power:	1.0
Request Interarrival Time Avg:	5.0
Request Interarrival Time Min:	2.0
Request Interarrival Time Max:	10.0
Request Interarrival Time Distribution Type:	0
Request Document Size Avg:	5000.0
Request Document Size Min:	1000.0
Request Document Size Max:	500000.0
Request Document Size Distribution Type:	0
Connected to:	[Cable13]

status:

Figure 3. Web Client Property Dialog box

There is a one-to-one correspondence between the modules in the software architecture and the Java classes implemented in NetLobars version 0.3. For example, the Client.java implements the I/O and event processing of client module and the ClientProperty.java implements the dialog window interface for collecting user's input on the specific client's parameters. Net class keeps the lists of network components, implements shortest path algorithm, and initializes the discrete event simulation. The PTPLink and MALink classes maintain the list of network components connected by them. The components can be connected by more than one of these links. The NetLayoutFrame class is responsible for the layout and processing of menus, canvas and status windows. The Resource Allocation Module is being implemented.

3. Message Animation

Protocol message animation is useful for designing and studying message exchanges of protocols implemented in network systems. In NetLobars system, the message animation is driven by the discrete event simulator. When a message is transmitted between two system components and the animation flag is turned on, the Message Animation Module is called from the discrete event simulator with information on the distance, transmission speed, the message type and size, and the locations of the sending and receiving components.

The Message Animation Module is implemented with the new JDK 2D API [Java98]. JDK 2D API greatly simplifies the coding of animation of messages on the canvas window. It enables us to rotate the message label at an arbitrary angle and allows the animation to follow exactly the transmission line in the display window.

Figure 4 shows the screendump of an animation step of a HTTP-request message, which was sent by client7 to server0 and being relayed from router2 to route0.

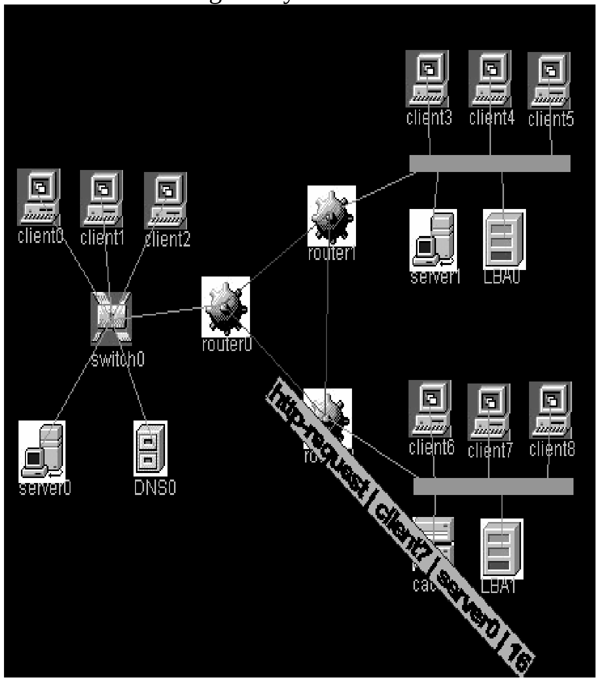


Figure 4. HTTP-request animation.

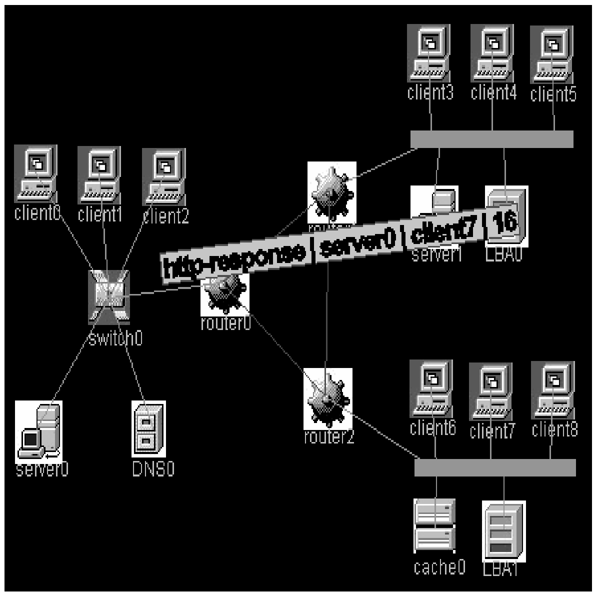


Figure 5. HTTP-response animation

Figure 5 shows the screendump of an animation step of the corresponding HTTP-response message on the way back to client7. It was being relayed from switch0 to router0. To make it easy to distinguish message types, we use cyan color for HTTP-request message label background and pink color for that of HTTP-response message label. The last field of the message label is the session ID generated by the simulator.

4. Identifying web system bottlenecks

Bottlenecks are defined as system components where work in progress, information, material are being excessively delayed. Since the "excessively delay" is a relative term, we must obtain data for the "normal delay" and define the threshold over which a delay can be called "excessive". Putting the web system in context, there are two basic ways we can identify the system bottlenecks. One indirect way is to use the utilization number. The direct way is to use the percentage of the system component delay in the end-to-end response time.

The former is to start by defining or collecting the capacity or peak rate of each system component. We then measure the utilization, a 0 to 1 number, of those capacities. The utilization is defined to be the ratio of the used capacity over the total capacity. For example, if the average used bandwdith of a 10Mbps line is 3Mbps, the utilization is 0.3. If a web server capable of 10000 connections per seconds is measured to serve 8000 connections per seconds, the utilization is 0.8. From queueing theory, we know there is strong relation between the utilization and the delay. It is generally true that the higher the utilization, the longer the delay. As a practical approach, we can keep track the utilization numbers of the system components during the simulation run. The heavily utilized system components tend to be the system bottlenecks.

In real system, it is very difficult to compute the percentages of various system component delays over the end-to-end response time. In simulation however, we can accumulate these delays as part of the simulation record along the simulation of web access processing. When the HTTP response is received by the client, the simulation record attached to the simulated HTTP response message will have all the delays of the web access processing. The component with the highest percentage will be the system bottleneck for this particular session.

Since there are many clients and many requests from each client, the system bottleneck on one session may not be the bottleneck on other sessions. One simple approach will be to provide a ranking system where the component with the highest delay in a session is given a high number, says 100, the second one with 99, and so on. Each component in the simulation model maintains a counter, which accumulates the ranking number value. Since the components used by different sessions may not overlap, the components such as the web server will have more overlap and result in higher counter value. To avoid the bias, we add a second variable, frequency, to the ranking structure variable, with the counter variable as the first field of the ranking structure variable. The final frequency variable will be incremented by 1, each time it is accessed. The final ranking value will be the counter value divided by the frequency. At the end of the simulation runs, the final ranking numbers will be calculated among all the system components and sorted. The components with higher-ranking numbers will be the system bottlenecks.

5. Conclusion

We have presented the design of a simulation-based web system planning tool, called NetLobars, for web system design and evaluation. Techniques for evaluating the web system bottlenecks are also presented.

We have found it very useful in providing quantitative numbers for comparing different web system configurations. As an example, for the web system presented in Section 2, with all clients retrieving document from web server0. the average end-to-end user response time is 2.30 seconds. With the addition of two cache servers in the two remote subnets, the average end-to-end user response time becomes 1.38 seconds. It improves by 66% for the same assumed client traffic pattern. This type of quantitative data help web system managers in their cost-performance trade-off and system upgrade decisions.

The NetLobars system is also useful in system evolution case. It can simulate the system performance when the client traffic is increased by certain percentage. For the same example in Section 2 with two cache servers in the remote subnets, after increasing the client traffic five times, says after 5 years, the simulator estimates average the end-to-end response time will be degraded from 1.38 to 32.73 seconds.

The animation feature provides designers with some insight on how the web protocol works and serves as a very effective learning and training tool for the emerging web system techniques. It is also instrumental in the design of a distributed load balancing protocol.

We are currently implementing the Resource Allocation Module for suggesting optimal or efficient network resource placement and allocation.

6. References

[Cisco96] Cisco Distributed Director, http://www.cisco.com/warp/public/751/distdir/dd_pa.htm

[Cola97] Colajanni, M.; Yu, P.S.; Dias, D.M., Scheduling algorithms for distributed Web servers. *Proceedings of the 17th International Conference on Distributed Computing Systems,* p. xvii+596, 169-76, 1997.

[Dias96] D. M. Dias, W. Kish, R. Mukherjee, and R. Tewari, " A Scalable and Highly Available Web Server," Proc. 41st IEEE computer Society Intl. Conf. (COMPCON) , pp.85-92, Feb. 1996.

[Fei97] Z. Fei, et al, "A Novel Server Selection Technique for Improving the Response Time of a Replicated Service," Georgia Institute of Technology, Tech Report, GIT-CC-97-24. Http://www.cc.gatech.edu/tech_reports/

[Java98] Sun Microsystems, "JDK 1.2", Http://java.sun.com/ products/jdk/1.2/index.html

[Katz94] E.D. Katz, M. Butler, and R. McCrath, "A Scalable HTTP Server: the NCSA Prototype," Comp. Net. And ISDN Systems, Vol. 27, 1994, pp. 155-164.

[Opnet] Modeler for Network Research and Development, http://www.mil3.com/products/modeler/home.html.

[COMNET] COMMENT III and COMMNET Predictor, http://www.caciasl.com/comnet.html.