Dynamic Simulation Control with Queue Visualization

Saurabh Mittal
Bernard P. Zeigler
Arizona Center of Integrative Modeling and Simulation (ACIMS)
ECE Department, University of Arizona
Tucson, AZ 85721
{saurabh, zeigler}@ece.arizona.edu

Abstract: This paper describes a DEVS-based network modeling and simulation environment with dynamic simulation control and queue visualization. The development of a simulation framework supporting run-time simulation tuning is facilitated by the DEVS modeling and simulation framework with its separation of model, experimental frame and simulator. The rapid feedback cycle supported by "real-time" intervention allows experimentation with parameters and structures and results in effective network configuration that is difficult to achieve when turnaround requires hours or days. An example of a distributed simulation application running on a wide area network is given to illustrate the new capabilities.

Keywords: DEVS, queue, visualization, network, System Entity Structure, simulation, HLA

1. INTRODUCTION

Although a number of commercial and academic simulators are available for complex network studies, none have the capability to tune the simulation while it is in execution. Due to tight coupling between the network model and the simulation engine in such simulators, the capability to introduce changes in parameter values during execution is limited or non-existent. The research described here has the objective of developing a DEVS-based network modeling and simulation environment with dynamic simulation control and queue visualization. The DEVS modeling and simulation framework separates model, experimental frame and simulator. This modularity facilitates the development of a simulation framework supporting run-time simulation tuning. The motivation behind providing "real-time" intervention is to support a rapid feedback cycle that allows experimentation with network parameters and structures. This can result in effective network configuration that is difficult to achieve when turnaround requires hours or days. Furthermore, such instantaneous observation and control enables important transient situations to be recognized and considered.

1.1 Real-Time Control And Visualization Imitations Of Existing Network Simulators

Some of the limitations of existing network simulation packages are as follows:

- Everything has to be programmed prior to simulating the network
- User interfaces are not easily customized and provide largely textual interfaces
- There is no support for changing parameters and component structures during simulation
- Simulation run times tend to be long (a few hours); more importantly, if a run ends in a crash, there is no way to intervene and re-adjust the system
- There is no run-time visualization of the system behavior to aid understanding and to steer the simulation in productive direction

To develop a network modeling and simulation environment that addresses these limitations, we extended the existing Discrete Event System Specification (DEVS) software, DEVSJAVA. In this paper, we shall discuss the layered architecture underlying the network simulation environment. After describing this architecture, we discuss some proposed run control and visualization techniques intended to greatly improve user understanding of, and ability to control, the complex structural and behavioral relationships characteristic of large network behaviors. An example of the use of the architecture will be given; it concerns modeling and simulation of network-based distributed engineering studies performed using the High Level Architecture (HLA) middleware.

2. LAYERED ARCHITECTURE

In order to create a DEVS Simulation environment, we follow a layered approach to define our model architecture. Figure 1 describes the layers under consideration. In this architecture, we have two layers, i.e., the Experimental Frame layer and the Network Modeling layer.

The Experimental Frame layer has the following functionality:

1. *Traffic Generator.* This component generates traffic for the network. The traffic generated or packets created will follow characteristic behavior according to stochastic or statistical distributions.

2. *Parameter Controller/Evaluator.* This component provides the capability to interactively conduct controlled experiments and visualize the effects of various parameters on the system behavior.

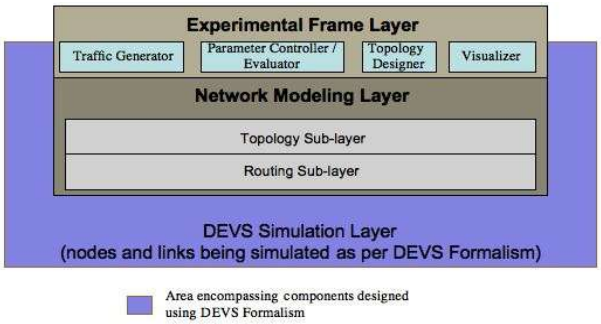

Figure 1. Layered Architecture Approach for DEVS M&S Network Framework

3. *Topology Designer*. This component creates topologies and provides the capability to change the topology at run-timed.

4. *Visualizer*. This component provides frames and panels for visualizing the states of various entities during simulation during execution and the statistics obtained from the simulation.

The Network Modeling layer consists of two layers: a *routing* layer that includes the packet routing functionality; this layer provides the routing of messages that are generated in the Experimental Frame layer. It currently implements the Dijkstra routing algorithm [*Ospf*], which is based on a link-state information database. The *topology* layer sits atop the routing layer and contains links between pairs of nodes to specify the topology of the network.

3. NETWORK CONTROL AND VISUALIZATION TECHNIQUES

3.1 System Entity Structure Description

The overall software development is described using the System Entity Structure (SES) [*Kim et.al 1990*] as illustrated in Figure 2. SES is a structural representation scheme that contains the knowledge of decomposition, taxonomy, and coupling of components of a model. Figure 2 portrays an SES of a DEVS Network model specialized for application to the HLA distributed simulation to be discussed. We'll describe the simulator processing and its interaction with users in a moment.

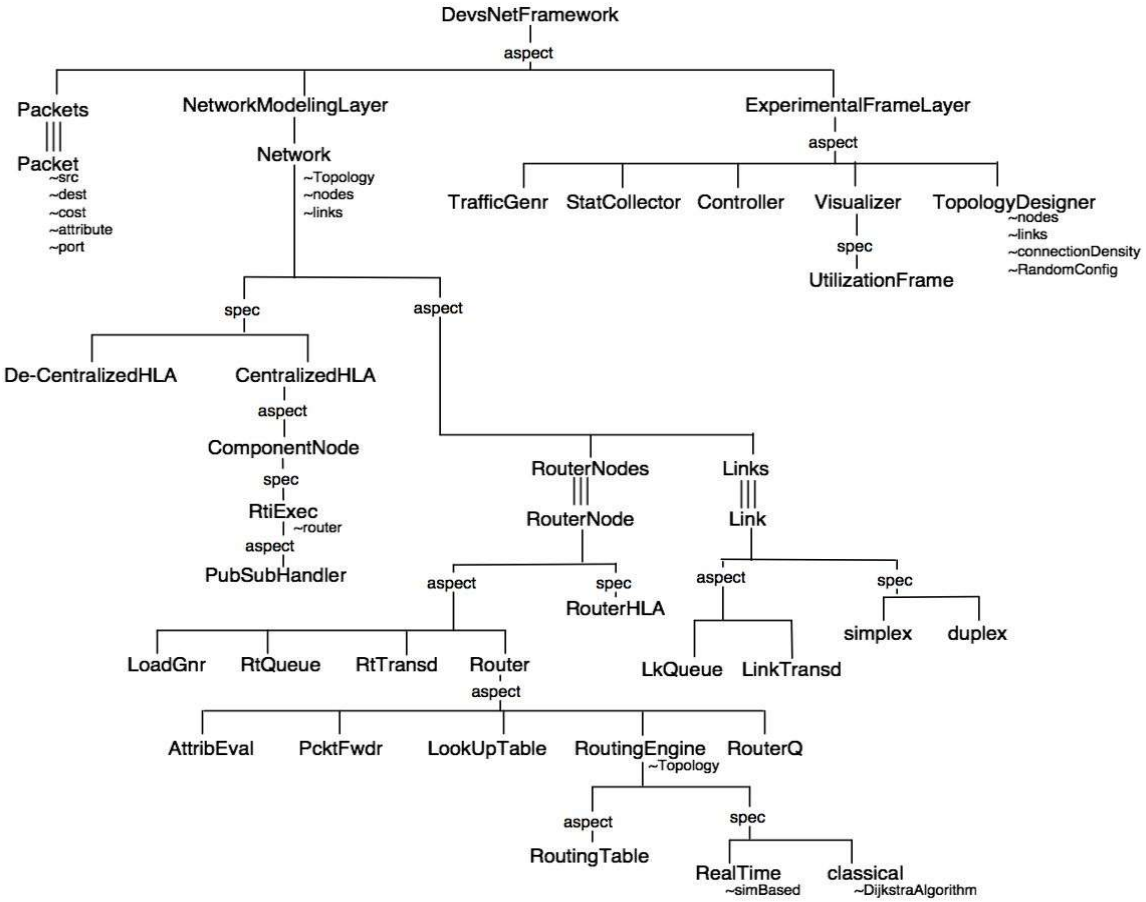

Figure 2. SES Structure of DEVS Net Framework for HLA Network modeling and simulation

3.2 Visualization Techniques

Visualization is an important part of any research endeavor that deals with large data sets and multiple components [WongBerg 1995]. There have been efforts to develop an end-to-end low-cost solution for analyzing and visualizing large time-varying data on distributed computing architecture [MaCamp 2000]. Due to large computation times, such solutions usually do not provide feedback to the modeler during simulation. As indicated before, our interest is in developing a visualization architecture that is integrated with the study of large scale networks and provides rapid feedback to modeler. An animation architecture for message passing in hierarchical DEVS models has been developed [Yi 2003]. The DEVSJAVA simulation environment has a similar visualization capability, SimView, that aids the design of DEVS modular and hierarchical models. However, this feature is not powerful enough when we are considering the design of a network system that can be viewed from many other perspectives.

To overcome the limitations of available simulators mentioned above, we designed our *Utilization* and *Controller* frames. The graphical representation of these frames will be shown later in Section 5, Figure 7. The features of these designed frames are as follows:

Utilization Frame

- is a modular Graphical User Interface (GUI) construction
- has Link Panel and Router Panel to display their buffer utilizations
- is created during the network setup definition
- can be updated at run time in the event of any link/router going up/down in the network

Controller Frame

- provides customized controls based on network parameters
- is interfaced with DEVS SimView (simulation can be entirely controlled by this frame)
- supports run-time adjustment and tuning of network / simulation parameters
- supports avoidance of simulation crash by intervention of user to modulate the system parameters

The *network topology designer* is intended to support creation of network topologies and their modification during simulation. Not finding any convincing open source topology generator with run-time support for changing network topology, we created our own Graph Modeler, as shown in Figure 3 with the following features:

Topology Designer

- allows specifying following parameters:
  - Number of nodes in the network (2…*n*) where *n* is the upward bound on node number
  - Connection density of the network (range [0,1]) where 1 means fully connected graph
  - Mean distance between the nodes, where distance may be interpreted as a network parameter such as available bandwidth, node-delay, capacity and cost.
- provides the ability to randomize connections with same topology at run time
- is coupled to DEVS engine during the simulation
- ensures that any run-time modification brings about corresponding change in DEVS simulation
- is constructed as a software component that can be interfaced with any simulation engine

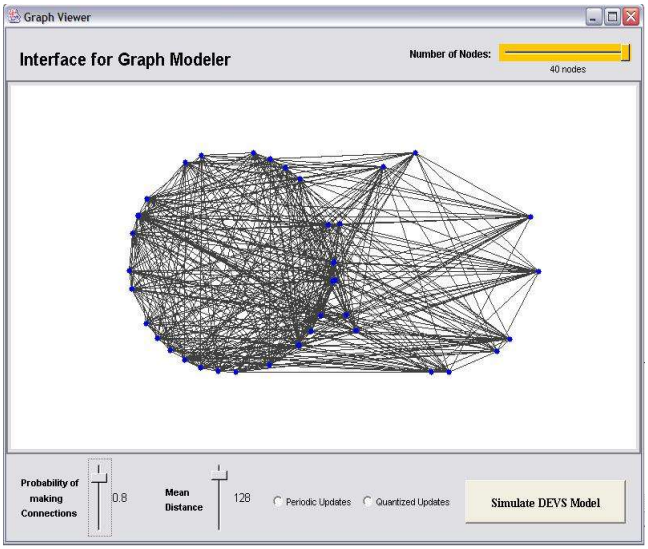

Figure 3: Graph Modeler to create topologies

3.3 Interrupt Handling by Simulation Controller Frame

The Controller Frame is built on top of DEVS Coordinator in DEVSJAVA. We developed interfaces to enable the DEVS engine to take into account the change of Experimental Frame parameters during the simulation run. It generates interrupts, which are handled by the coordinator in DEVSJAVA.

4. EXAMPLE: HLA/RTI SIMULATION

The HLA middleware consists of federates and Run-time Interface (RTI) [*Dahman et.al 1997*]. In a centralized architecture, there is have a single RTI Executive to control the transactions between the different federates. In a decentralized architecture, this functionality is distributed over a number of nodes. The objective of the study was to compare the performance of centralized and decentralized architectures. Accordingly, the Experimental Frame layer is specialized to support both centralized and

decentralized RTI architectures. In a centralized architecture, illustrated in Figure 4, updates are sent by federates to a single RTI Executive that then distributes them to all federates in a publish/subscribe manner. A decentralized architecture is modeled as a peer-to-peer network in which federates exchange state updates among themselves with much of the processing of a central executive replicated at each node.

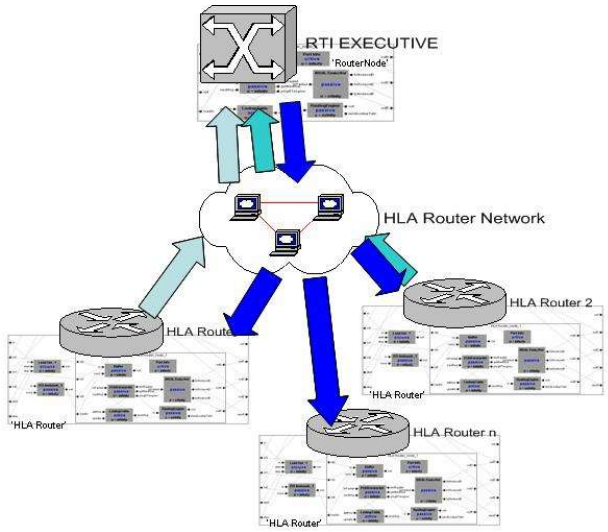

Figure 4. Conceptual Layout of HLA/RTI network architecture

We constructed our "HLA router" and "RTI Executive" according to Figure 4. The details of inner DEVS components can be seen in the SES structure in Figure 2.

The RouterHLA Network consists of DEVS links and DEVS RouterHLAs connected in a particular topology. The RTI Executive Node (RTIExecNode) is one of the nodes in the constructed topology. Traffic emitted by federates is routed to the RTIExecNode in the case of Centralized RTI Architecture. In the non-centralized case, it is routed to another federate, as in a point-to-point (P2P) RTI architecture.

The present study is limited to modeling the RTI as RTI Ambassador, which is not more than a simple M/M/1 queue having a mean delay as a "controllable" parameter.

5. EXPERIMENTAL SETUP

After defining the DEVS models and testing their operations in a standalone manner, we created networks composed of DEVS HLA Router models, DEVS links, and a DEVS RTI Exec in either centralized or non-centralized form.

The network topology can be created using the Topology Designer (refer to Figure 3) where the nodes become DEVS HLA Routers and connections become DEVS links. The node with the highest ID in Graph Modeler defaults to become the RTIExecNode for experiments with centralized architectures. The Topology Designer is an optional component that can be omitted in case the designer wants to create a customized model by explicitly specifying nodes and connections.

A default experiment defines the following parameters for simulating a network (Table 1). The initial values of these parameters are defined at the start of simulation during setup of network layout.

S.No.	Parameter Name	Initial Value	Data Type
1	Router Buffer Length	20 packets	Int
2	Link Buffer Length	10 packets	Int
3	Packets generated per sec	10 pkts/sec	Int
4	Mean RTI Delay	1 sec	double
5	Mean Router Delay	0.1 sec	double
6	Mean Network Delay	10ms	double

Table 1. Simulation-parameters table

The simulation packet is provided with the following fields. A single packet can be translated to *bytes* based on the data or attribute payload.

Source
Destination
Destination Port
nextHop
Packet Type *0=Data, 1=RoutingUpdate*
Attribute 1
Attribute 2
|
Attribute n
Cost Accrued

Figure 5. Simulation packet structure

Once the network is set up, these simulation parameters can be adjusted using the Controller frame. The Controller frame is designed in such a manner that the parameters can be changed at run-time during active simulation so that the individual effect of these parameters can be immediately observed. Detailed analysis of these parameters will be provided in next section.

The default start of experiment executes the following tasks:

a) Initializes DEVS routers and DEVS links
b) Creates publish/subscribe class information for each of the routers
c) Executes the link-state routing algorithm (Dijkstra) inside the Routing Engine to implement the Net-

work layer functionality in the layered architecture described earlier.

d) Creates Controller, Utilization Frame, and other Analyzer plots for network-state visualization and analysis
e) Gets the network ready to start generating packets

Once the initial setup has been done, the simulation is started that executes the following steps repeatedly:

1. The user sets the traffic injection parameter (packets/sec) through the Controller Frame where it is then relayed to the Load Generator component inside every RouterHLA to create and generate packets (refer to SES for component structure).
2. The packets are then fed into the RTI Emulator component that incorporates the RTI delay parameters.
3. The packets then enter into the Networking Layer component 'router' for routing where they are put on to a specific outport based on the network topology and Lookup Table in router.
4. The packets travel through the network links and are then received by the routing module of destination the next hop RouterHLA. If the node is the intended destination, then the packet is absorbed there else it is forwarded based on the Lookup Table.
5. Changes in the network parameters (described above) are injected at any time during the simulation execution to study their effect on network queue utilization or network latency.
6. The real-time network state and /or component's state is visualized as the simulation proceeds in time.

Experiments Illustrating User "Real-time" Intervention

The topology under experiment is shown in Figure 6. Experiment 1 is the P2P de-centralized HLA network and experiment 2 is the Centralized HLA network with RTI Executive node.

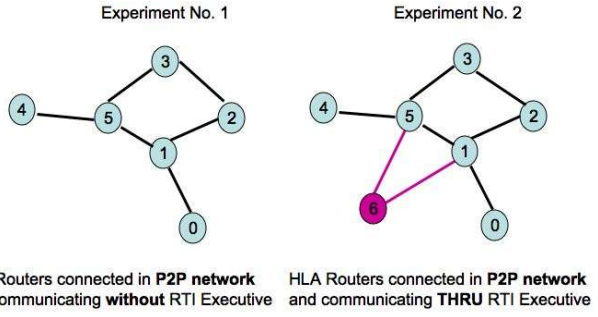

Figure 6. Topology for Experiments

Figure 7 shows a snapshot of Experiment 1. Once the simulation achieved steady state after initialization, the RTI Delay was reduced to 0.01, the load was increased from 10 packets/sec to 25 packets/sec. The experimental network responded with huge surges in the link queue lengths and the link capacities had to be increased from 10 packets to 20 packets to accommodate such surge. Notice the change in the Controller frame in Figure 7. We witness an increase in throughput of the network. This situation is easily explained since we are sending more packets into the network and most of the packets are to be found in the link queues. Since the RTI Delay is relatively small (0.01), there are hardly any packets in the RTI.

The Controller Frame allows us to modify the capacities of the system (e.g., link capacities in this experiment when load is increased) when the network exceeds beyond its normal operating conditions. This then helps us modify the parameters at run time so that we can quickly arrive at a set of parameter values for a stable steady state solution of network under simulation study.

Figure 8 shows a snapshot of Experiment 2. In this particular experiment, each HLA Router publishes a message of a particular class to the RTI Exec and sends packets to the RTI Exec. The RTI Exec multicasts them to other HLA Routers. Those routers that have subscribed to this particular class of message are sent this message. This experiment contains additional statistical plots than Experiment 1.The first row (figure 8) belongs to links between HLA Routers, and the second row of plots belongs to RTI Executive interfaced Links. Since we anticipated a high volume of traffic in links coming out of RTI Exec, we set their capacities to be threefold, with throughput from the usual 20 packets/sec to 100 packets/sec. We conjecture that links coming in/out of RTI Exec should be capable of handling traffic at least four-fold, hence the increased default values.

As the simulation started with a load being generated at 10 packets/sec (default value) and an RTI mean delay of 1.0 sec (default). The simulation started with queue lengths slowly building up, along with the link throughputs. Within 7 seconds the queue lengths as well as throughputs have exploded. The link capacity was increased from 10 packets/sec to 60 packets/sec to accommodate the traffic, but it was not sufficient to control the throughput.

Without restarting, we lowered the rate of packet generation, i.e., the Load generated per second was reduced from 10 packets/sec to 4 packets/sec at 7.21 seconds. There was no change made in the RTI mean delay parameter, which continued to stay at 1.0 seconds. We see that the system takes some time to respond to this adjustment Figure 8. After some time, i.e., at clock 11.941, we observed that the throughput of the system (both RTI links and router links) has been lowered. The *response time* for this change was around 5.6 seconds.

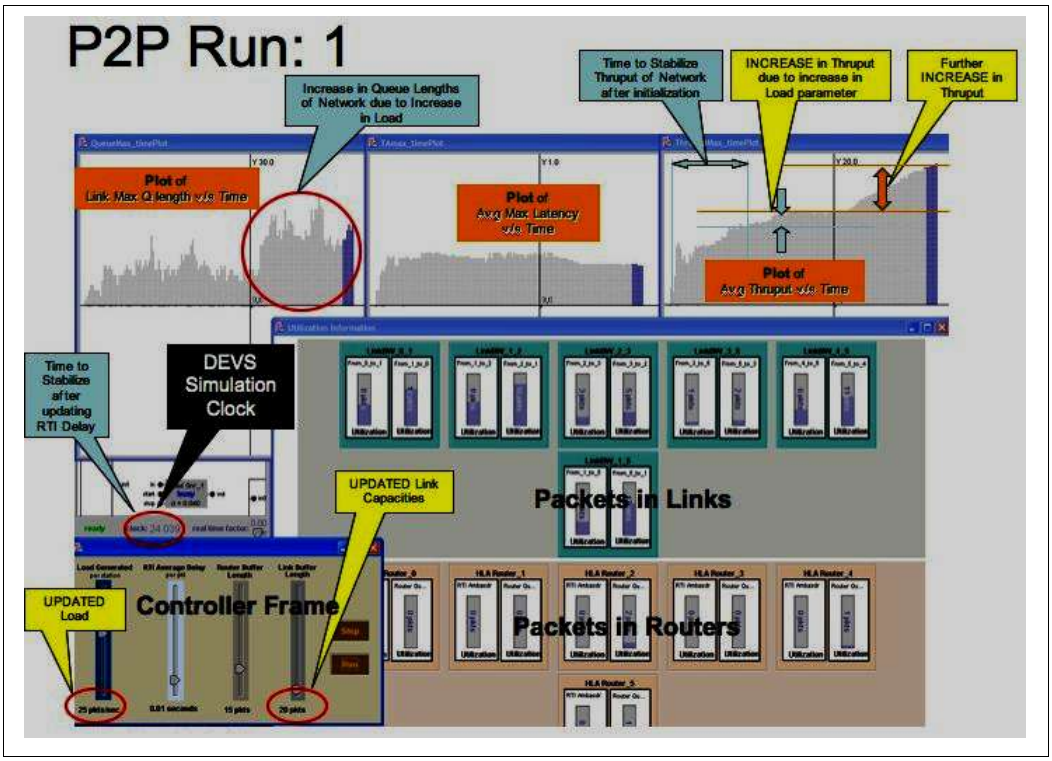

Figure 7. Snapshot of HLA P2P network simulation experiment

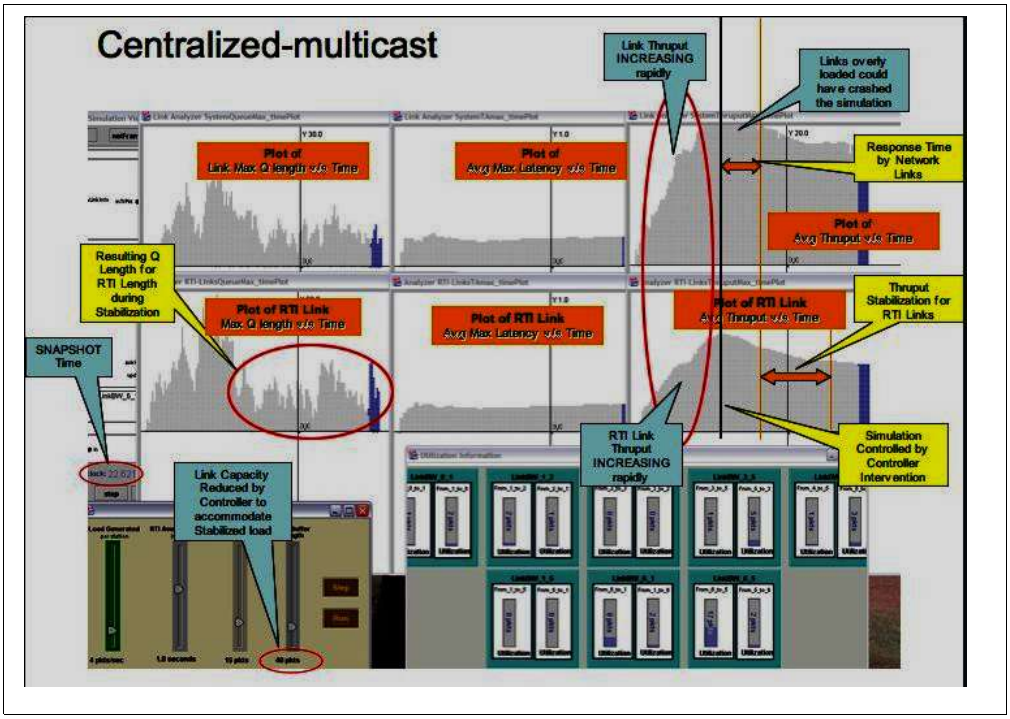

Figure 8. Snapshot of Centralized HLA network simulation

We continued our simulation run to see when the steady state arrived. It took an additional 10 seconds for the throughput to stabilize. Finally, we lowered the link capacities from 60 packets to 40 packets. The system achieves a steady state with load generated at 4 packets/sec, RTI Mean delay of 1.0 sec, and link capacity at 40 packets.

Experiment 2 successfully implemented the multicast capability of the RTI Executive. In addition to just tuning the steady state network configuration values, the network statistics gathered were *latency*, link *throughput* and maximum *queue length* displayed by links. The end-to-end *latency* values obtained during the simulation run were then compared with the available commercial RTI values. A full report [*Mittal et.al 2004*] is available from the authors.

Simulation Capabilities

The above visualization capability enabled us to see that throughput takes some time to stabilize and that the change in any network parameter has considerable transients along with long-term effects before a steady state is achieved. Getting this kind of qualitative data by mining the event-logs would require additional parsers and data-analyzers to be written and ultimately be displayed graphically by recreating the simulation from event-log. Using the new capability, this information is now readily available through the real-time visualization of system and the rapid feedback provided by the integrated visualization with the executing simulation.

6. DISCUSSION AND CONCLUSION

Consider the general case of simulations to plan the network capacity and configuration need for distributed simulation to support engineering studies. Such simulations need to describe network structure and behavior only at levels of abstraction that provide reasonable estimates of bandwidth requirements and take into account performance-level representations of the underlying middleware. Adopting appropriate levels of abstraction can be exploited to provide quicker turnaround of simulation results. But even more importantly, with appropriate visualization support, such abstraction can allow direct and "real-time" observation of dynamic network behavior that can deviate markedly from that characteristic of the steady state.

The current study has demonstrated the design and operation of a network simulation environment that provides a proof-of-concept of the approach just described. We note that dynamic surges due to changes in network loads or structures (such as breakdowns) are well perceived with the use of the visualizer. In contrast, these surges might be easily missed in scanning the detailed log of a traditional simulator with output at the end of the run only. The rapid feedback cycle supported by "real-time" intervention allows experimentation with parameters and structures and results in effective network configuration that is difficult to achieve when turnaround requires hours or days and where important transient situations might never be considered.

References

[Kim et.al 1990] T.G. Kim, C. Lee, E.R. Christensen, B.P. Zeigler, "System entity structuring and model base management", IEEE Transactions on Systems, Man and Cybernetics, Volume 20, Issue 5, Sept-Oct., 1990

[Dahman et.al 1997] Judith S. Dahmann, Richard M. Fujimoto and Richard M. Weatherly, "The Department of Defense High Level Architecture," Proceedings of the 1997 Winter Simulation Conference

[Ospf] www.ospf.org

[Ns2] http://www.isi.edu/nsnam/ns/

[Opnet] www.opnet.com

[Kolek et.al 2000] Stephen Kolek, Steve Boswell, Harry Wolfson, "Modeling and Predicting HLA Federation Performance," 00F-SIW-085, Distributed Systems Performance Modeling Office (DSPMO), MIT Lincoln Laboratory, September 2000

[Zeigler et.al 00]Bernard P. Zeigler, Herbert Praehofer, Tao G. Kim, "Theory of Modeling and Simulation," Academic Press, 2 Edition, January 2000

[Akerkar 2004] Salil R. Akerkar, "Analysis and Visualization of Time-varying data using the concept of 'Acitiviy Modeling'", MS Thesis, University of Arizona, 2004

[WongBerg 1995] Pak Chung Wong and R. Daneil Bergernon, "A too for hierarchical representation and visualization of Time-varying data", ICASE/LaRC Symposium, 1995

[MaCamp 2000] Kwa-Liu Ma and David Kamp, "High Performance Visualization of Time-varying volume data over Wide-Area nework", Proceedings of Super Computing 2000

[Mittal et.al 04] Saurabh Mittal, Bernard P. Zeigler, Phillip Hammonds, Mahesh Veena, "Network Simulation Environment for Evaluating and Benchmarking HLA/RTI Experiments", JITC Report, Fort Huachuca, December 2004.

[Yi 2003]M.R. Yi T.H. Cho "Hierarchical simulation model with animation", Engineering with Computers, Issue: Volume 19, Numbers 2-3August 2003, pps. 203 – 212