AFRL-IF-RS-TR-2005-352
Final Technical Report
October 2005

ONLINE SIMULATION AND CONTROL

Science Applications International Corporation (SAIC)

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. K147

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2005-352 has been reviewed and is approved for publication

APPROVED: /s/

WAYNE A. BOSCO
Project Engineer

FOR THE DIRECTOR: /s/

WARREN H. DEBANY, JR., Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE
OCTOBER 2005

3. REPORT TYPE AND DATES COVERED
Final Jul 00 –Sep 05

4. TITLE AND SUBTITLE
ONLINE SIMULATION AND CONTROL

5. FUNDING NUMBERS
C - F30602-00-C-0189
PE - 62301E
PR - K147
TA - 18
WU - A1

6. AUTHOR(S)
Jerilyn J. McElwee,
Bradley S. Gaspard,
Elaine Keith

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Science Applications International Corporation (SAIC)
8301 Greensboro Drive
McLean Virginia 22102-3600

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency	AFRL/IFGA
3701 North Fairfax Drive	525 Brooks Road
Arlington Virginia 22203-1714	Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2005-352

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Wayne A. Bosco/IFGA/(315) 330-3578/ Wayne.Bosco@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 Words)*
Defense Advanced Research Projects Agency (DARPA) invested five years in research to improve the fidelity and performance of network simulations. Universities, industry and government have worked together to demonstrate new state of the art research in the areas of network modeling and simulation. Within the DARPA Network Modeling Simulation (NMS) program, SAIC performed research in the network area of Online Simulation and Control (OSC). The goal of the OSC project was to determine value added military uses of high performance network simulation and to research the infrastructure that is required to enable eventual deployment of network simulation to those locations. DARPA’s investment has improved the state of the art and demonstrated to DoD observers the potential of network modeling and simulation. The OSC framework provides a capability to apply network simulation to real-time network management. OSC allows simulations to run in conjunction with a real network as a decision aid. The simulations can be used to explore alternative network tunings and to compare them with the performance of the actual network as they both execute. The OSC framework is currently delivered as a virtual test bed for testing the ability of a simulation to provide accurate results in near real-time from the limited information available from the real-network. The framework includes standard scenarios and network traffic for study. It is modularly adaptable to real test beds as well

14. SUBJECT TERMS
Network Simulation, High Performance Networking, Network Modeling Simulation, NMS, Global Information Grid, GIG

15. NUMBER OF PAGES
31

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

LIST OF FIGURES

1. Overview

1.1. Background

Defense Advanced Research Projects Agency (DARPA) has invested five years in research to improve the fidelity and performance of network simulations. Universities, industry and government have worked together to demonstrate new state of the art research to the Department of Defense (DoD) and others at principal investigator meetings. Within the DARPA Network Modeling Simulation (NMS) program the basic goal of the Online Simulation and Control (OSC) project has been to determine value added military uses of high performance network simulation and to research the infrastructure that is required to enable eventual deployment of network simulation to those locations. DARPA's investment has improved the state of the art and demonstrated to DoD observers the potential of network modeling and simulation.

The OSC framework provides a capability to apply network simulation to real-time network management. OSC allows simulations to run in conjunction with a real network as a decision aid. The simulations can be used to explore alternative network tunings and to compare them with the performance of the actual network as they both execute. The OSC framework is currently delivered as a virtual test bed for testing the ability of a simulation to provide accurate results in near real-time from the limited information available from the real-network. The framework includes standard scenarios and network traffic for study. It is modularly adaptable to real test beds as well.

The primary focus of OSC has been to improve network management by the use of near-real-time simulation. For network management the simulation is used to estimate the Quality-of Service (QOS) as perceived by the network users. The simulation is also used to allow near-real-time "what-if" studies supporting network tuning in the presence of time varying traffic conditions.

The use of simulations to support design decision on complex systems or to support the warfighter can take months. Creating the scenarios and importing them into the simulation tool, to run the simulation, and to analyze the results can be a time-consuming process. OSC can shorten this time cycle.

1.2. Document Overview

This document consists of the following sections. Section 1 provides an OSC background description. Section 2 is a more detailed description of the OSC design and architecture. Section 3 provides a summary of the major accomplishments completed under this task. Section 4 describes some latter work which looked at shifting from a basic client/server type of architecture to one based on Web Services.

2. OSC Design

The OSC framework incorporates network simulation into network management as shown below.

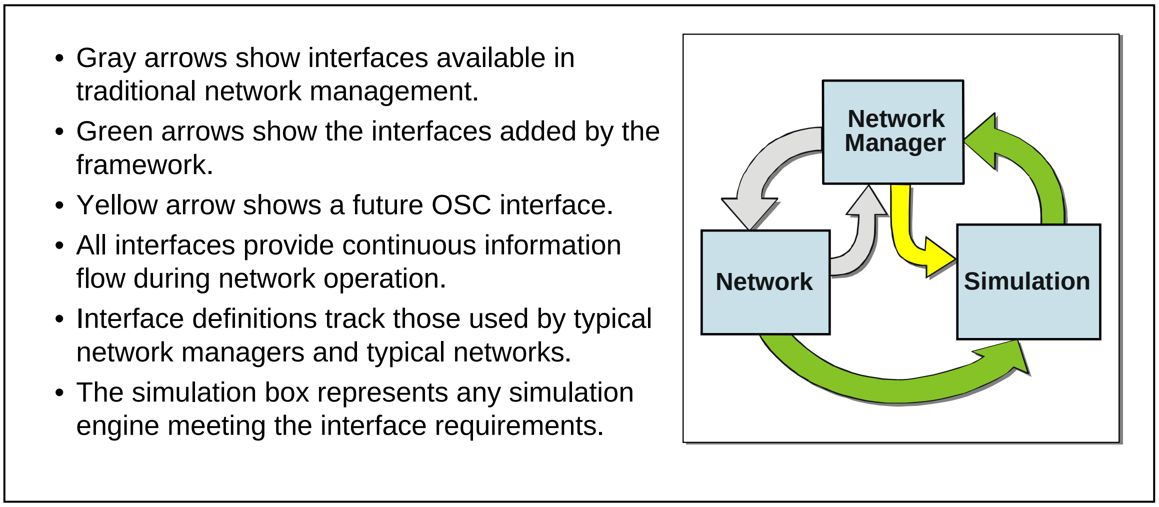


Figure 2-1. OSC Framework

The integration paradigm provides execution of one or more simulations simultaneously with the network. Traffic "sniffers" extracting actual or simulated network traffic continuously feed network traffic information to the simulation. In addition, the system continuously processes the simulation outputs to provide summaries relevant to the network manager. In the current implementation, the simulation can be stopped, rebuilt and restarted when the network topology changes. This is appropriate at the current time, since none of the simulation engines that we have explored can change the network topology in a natural way while the simulation is executing.

The paradigm used by the framework was chosen for its ability to provide a number of key features:

- Reduced demands on simulation speed to provide timely results.
- Continuous simulation validation.
- Simplification of presentation of simulation results to the network manager.
- Simplified management of simulation execution allowing the network manager to exploit the simulation with little effort.

The system architecture is shown in Figure 2.2.

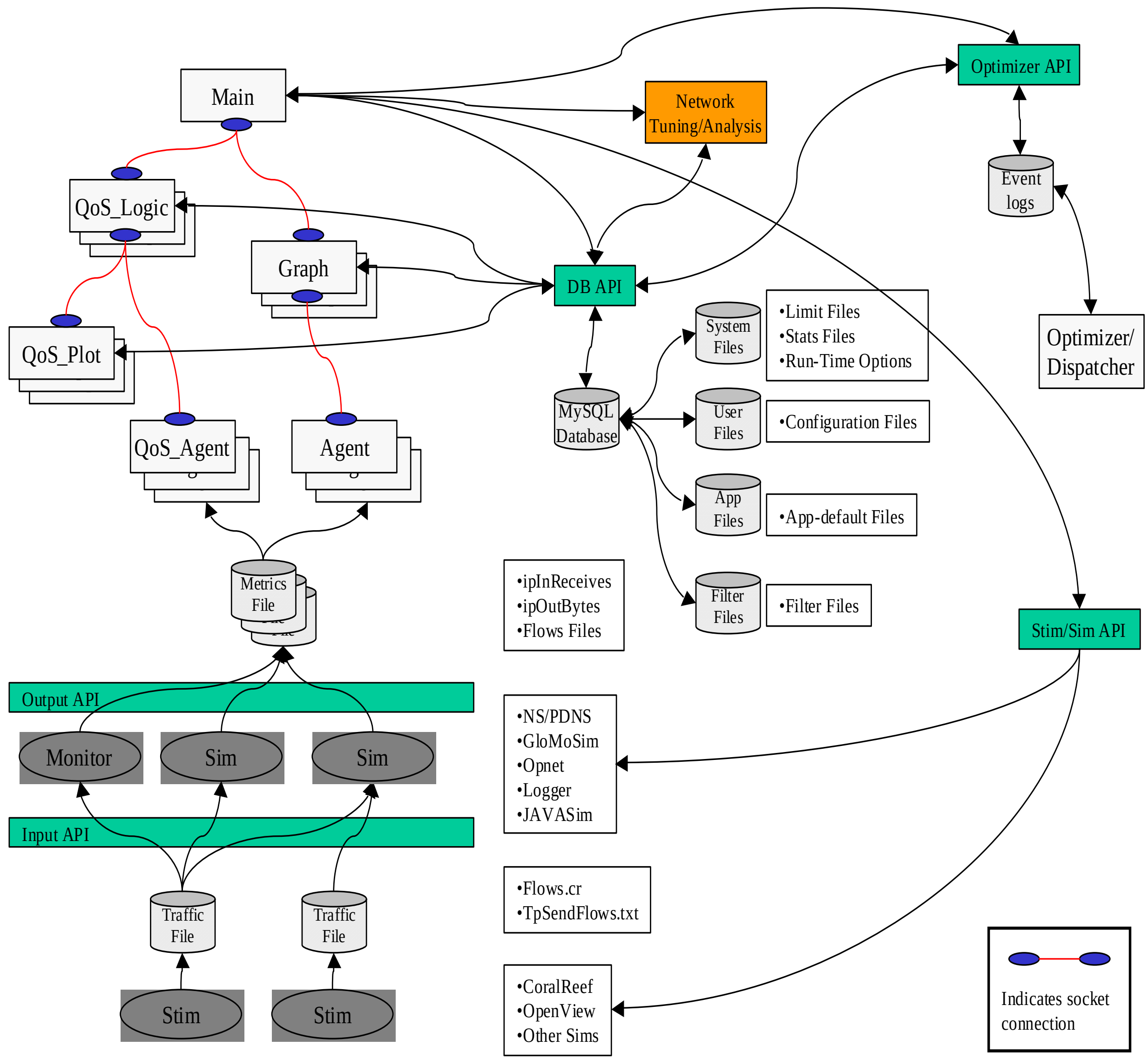


Figure 2-2. OSC Architecture

2.1. System Requirements

The OSC application is written in Tcl/Tk/BLT and uses either MySQL or Postgress as the database server. OSC has been tested using Solaris 6 and 8 and Linux Red Hat 7.1 and 8.0 and Windows 2000. OSC requires:

- Tcl version 8.3.2, and
- Tk version 8.3.2, and
- BLT version 2.4u, and
- MySQL Version 3.23.52, or
- Postgress Version 7.3.2

An Application Programming Interface (API), developed in "C", allows for the integration of simulator engines.

2.2. Displays

The OSC Framework is a graphical user interface comprised of a set of displays that are launched from one primary display (Main). The Main display also controls the operation of the OSC Framework by defining the network interface(s), Simulation interface(s), the Graphical statistical displays and the launching and/or termination of the configuration (Figure 2-3).

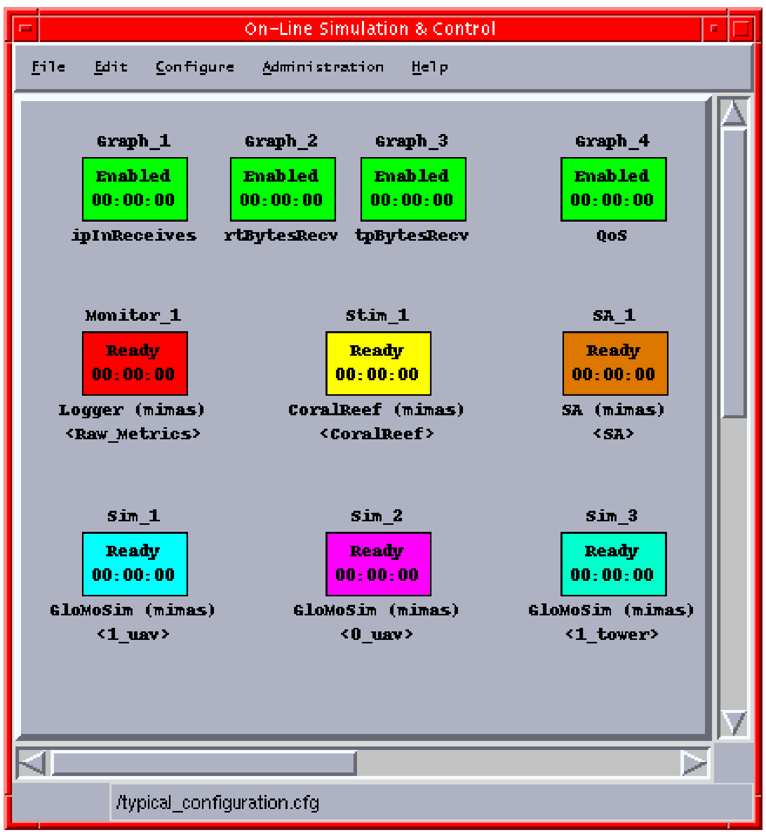


Figure 2-3. Sample Main Configuration window

2.3. Configuration

The Main display provides the ability to configure the desired operational approach, store it and then load and execute a configuration quickly. Each configuration item is identified with an icon on the Main display.

A configuration is saved to a file that can be recalled and edited if desired. Once a configuration is created, the user has the option to launch an individual item, a subset of the configuration, or the entire configuration. The OSC Main process controls and synchronizes all processes.

A configuration in OSC may include Stimulators, Simulators, Clusters, Replays, Monitors, Optimizer, Situational Awareness (SA) and Graphs. To add a process to the configuration, the user selects the item desired from a drop down menu. Once selected, the system generates an icon to represent that entity. After creating the icon, the user can configure the item by "right clicking" within the icon space. The data flows are defined by the system based on the user-developed configuration. For instance, Stimulators feed Simulators, while Simulators feed Graphs.

2.3.1. Stimulator

A Stimulator is the source of network traffic data that will feed into the Simulations in real-time (Figure 2.4). The Stimulator can be a simulation of an instrumented network, a real network or captured and stored traffic information from a real network feed into the OSC framework. When the Stimulator is also functioning as a Simulation running in real-time, its use is special from the other simulators in the configuration, which makes it a Stimulator. The special feature is that selected components of its outputs become inputs to each of the Simulators.

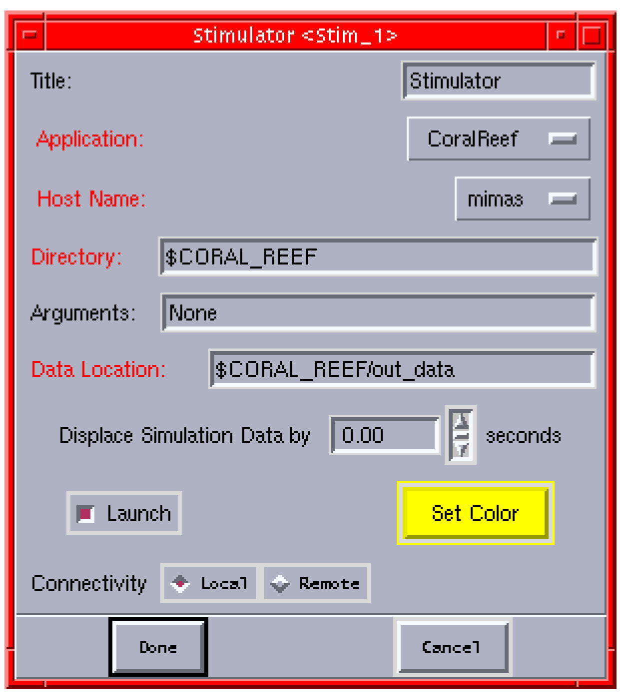


Figure 2-4. Sample Stimulator Configuration window

The OSC framework allows plug-and-play of a choice of simulation engines as a Stimulator. To make a good test bed, we have used simulation engines integrated with the SEAMLSS environment to model network users using the network in a doctrinally defined manner. This is not required in general. This external traffic injection mimics real users on a simulated network, thus increasing the credibility and relevance of results obtained with the test bed for use on real networks.

A Stimulator is instrumented to output flow information generated by one node, and sent to another. The flow information contains information such as which node sent the packets, which node(s) are to receive the packets, and how many bytes and packets are being sent. The reporting interval is controllable from the Stimulator itself and is not part of the configuration window. The individual nodes in the simulations are identified by IP address, as in a real network. This is currently the "tpSendFlow" file, which stands for "transport layer flows". If there is no Stimulator in the configuration, then the Simulators use only traffic set in the Simulator. The format is the same independent of which layer outputs the flows.

2.3.2. Simulator

A Simulator in the main OSC interface represents a simulation modeling the actual network, or some derivation (Figure 2-5). The Simulator may obtain external message traffic from a Stimulator. If so, the simulator may model some or all of the individual IP addresses given by the Simulator.

There may be many simulations simultaneously configured and launched in the OSC interface. Each reads the "tpflows" from the stimulator, if so configured. Each Simulation models whichever fraction of the traffic is appropriate for it to model.

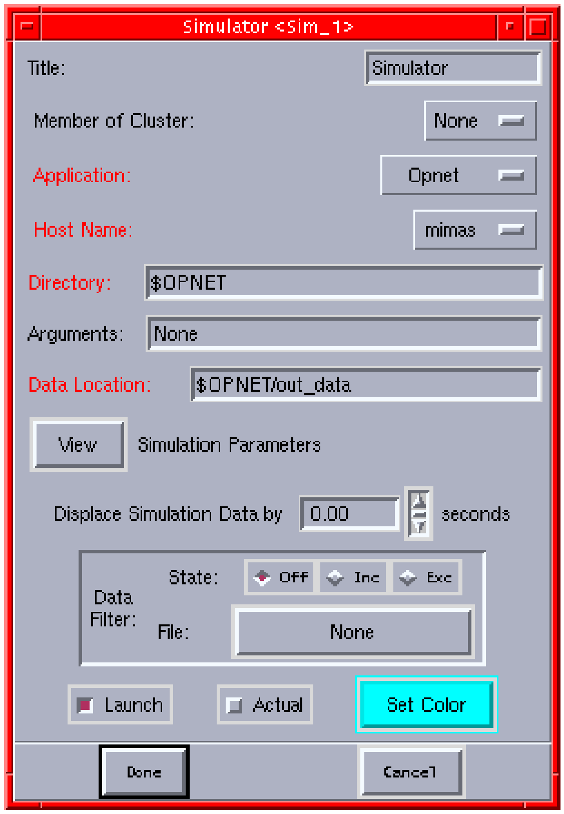


Figure 2-5. Sample Simulator Configuration window

2.3.3. Replay

A Replay provides data for graphing in the same way that a Simulator does, except the data was previously generated and recorded. A Replay is a Simulator where the "launch" button is set to "off" in the configuration window. Refer to the Simulator section for more detail. To add a Replay to a configuration, select Edit, create, "replay" from the main menu. A Replay box icon will appear on the screen and can be configured by "right clicking" on the icon.

2.3.4. Cluster

A group of Simulators or Replays can be grouped into a Cluster (Figure 2-6). A Cluster is used when each Simulator or Replay is a piece of a total network, perhaps distributed among multiple machines. The Cluster defines the cumulative set of Simulations or Replays that describe the network.

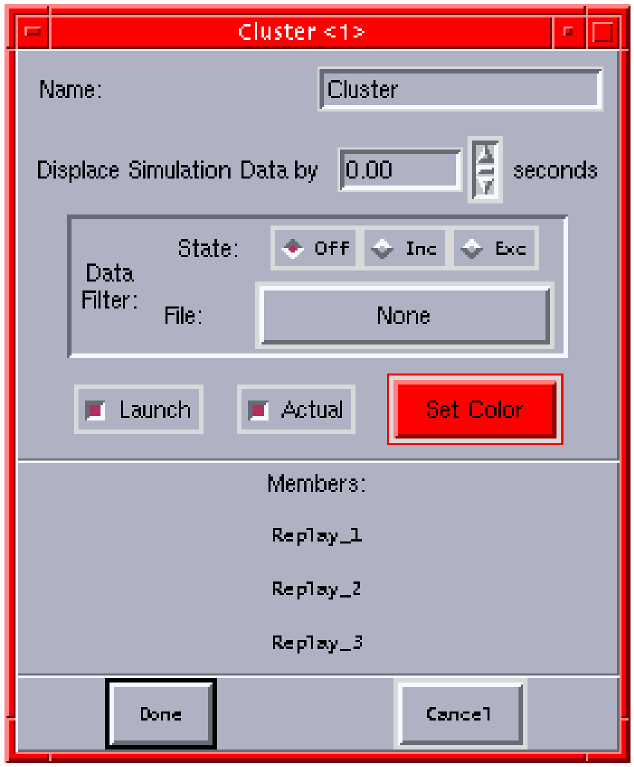


Figure 2-6. Sample Cluster Configuration window

2.3.5. Monitor

The Monitor process ingests network traffic collected, and reports the statistics as they are input (Figure 2-7). A Monitor is especially useful for comparing the results of a simulator, or comparing the upstream and downstream feeds across another gateway or hardware device. Only primitive metrics are collected by the Monitor since the Monitor process cannot determine delays in messages.

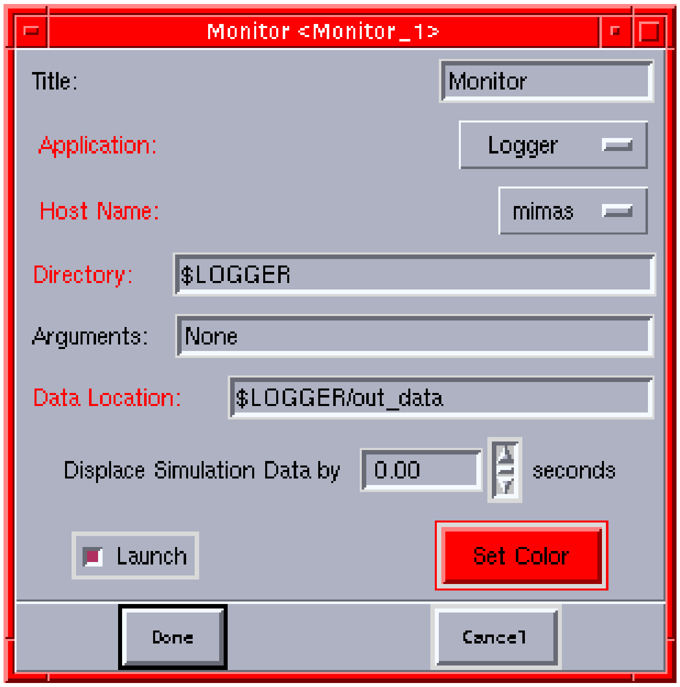


Figure 2-7. Sample Monitor Configuration window

To create a Monitor, select Edit, create, "monitor" from the main menu. A Monitor icon will appear on the screen and can be configured by "right clicking" on the icon.

2.3.6. Graph

Graphs provide the opportunity to create displays of data that is the output from Stimulators, Simulators, Replays and/or Monitors (Figure 2-8). Each Graph icon in a configuration creates at least one displayed Plot. Each Graph displays data for one statistic collected from one or more data sources. This allows for the simultaneous comparison of data from different Stimulators, Simulators, Replays and/or Monitors.

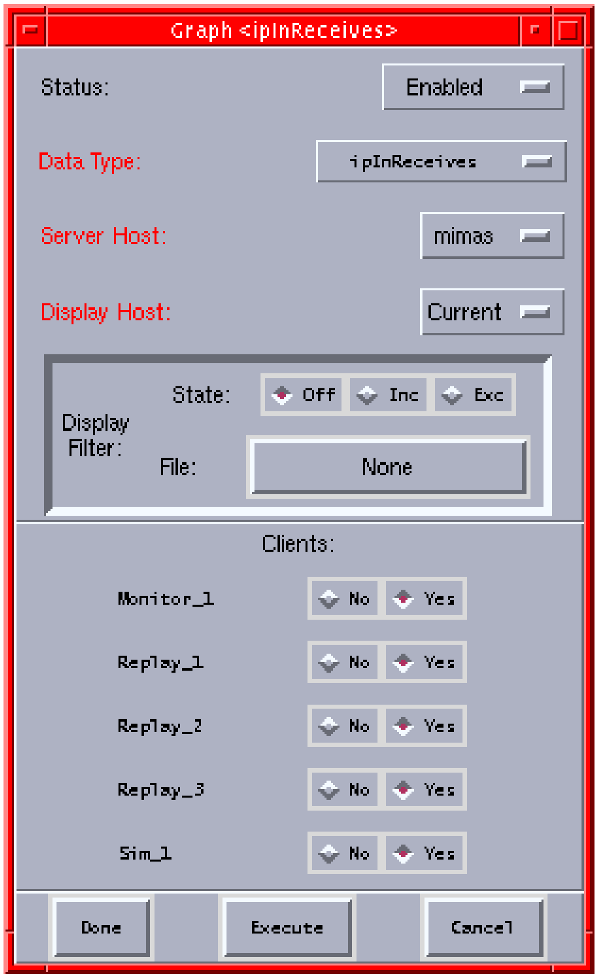


Figure 2-8. Sample Graph Configuration window

To create a Graph, select Edit, create, "graph" from the main menu. A Graph icon will appear on the screen and can be configured by "right clicking" on the icon.

Each Graph icon created results in one Graph being displayed. That display is created when the configuration is launched, or the individual Graph is executed. The Graph display can be viewed on the current platform, or any other platform configured in the Display Host icon. An example graph is shown in Figure 2-9.

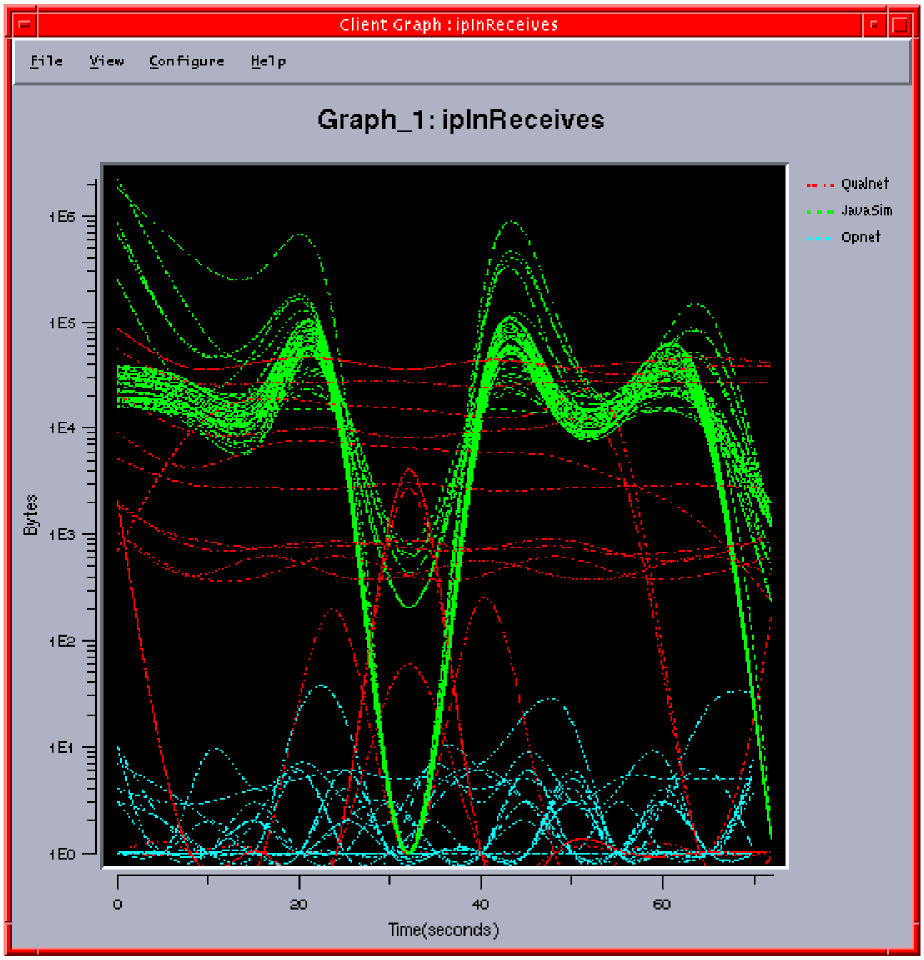


Figure 2-9. Sample ipInReceives Graph window

2.3.7. Optimizer

The Optimizer process allows several side-by-side simulations to be compared dynamically (Figure 2-10). In general, the Optimizer accepts a list of variables, each with a series of values, and generates a single simulation run for a subset of these variable/value combinations. Each of these simulations generates a QoS metric that the Optimizer uses as a condition for determining whether this run is better or worse than other runs. The Optimizer, in effect, tries to maximize the QoS value. The process of generating new simulation parameters, running the simulation, and obtain the QoS metric continues until either the list is exhausted or the Optimizer cannot maximize the QoS value any more. Once complete, the Optimizer informs the Network Manager which run produced the best overall QoS and the simulation parameters comprising that run. Hence, the Optimizer finds the optimal network configuration based on the list of parameters supplied. Optimizers have their own plots. The first is the QoS value per Simulation. The second is the QoS averaged over several simulations (referred to as a generation). The generation plot also displays the local minimum and maximum QoS values of the members.

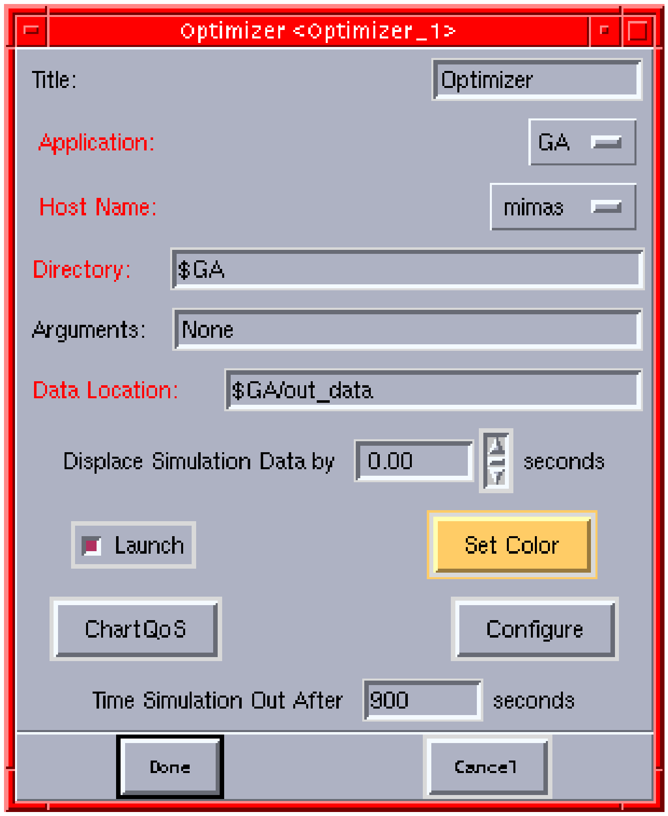


Figure 2-10. Sample Optimizer Configuration window

2.3.8. SA (Situational Awareness)

A SA process allows the user to visualize the data on a background map (Figure 2-11). The SA process reads the Scenario Description File (SDF) file and generates positional information about each item. This information is injected into the database, with updates performed at the relevant time as specified in the SDF file. Thus, an external process, such as any Joint Mapping Tool Kit (JMTK)-like Graphical Information System tool can display the current positions of the nodes processed in the simulation. Furthermore, the OSC can accept node positions from an external process provided the external process injects the data into the database.

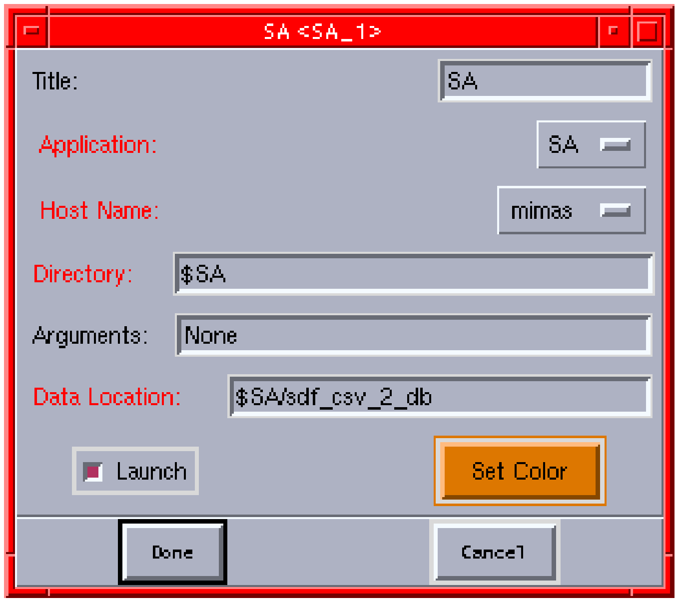


Figure 2-11. Sample SA Configuration window

2.4. *Real-time data flow*

A key aspect of integrating simulation engines for online simulation is input and output of data in real-time. In contrast typical network simulation uses a batch process, which OSC helps a simulation engine to replace with the real time requirements.

2.4.1. Traffic data input via files

Traffic data input from a stimulator (such as a real network or another simulation) into a simulation is accomplished through interface files (generated through the API). To inject external message traffic into the simulation, this file must be read. To properly synchronize file transfers, the file writer includes an "end interval" to indicate that it has provided all information for an interval. The file reader can then read each interval, and when the "end interval” marker is found, it can execute the simulation, up to the end of the interval knowing that no additional traffic information will be forthcoming for that interval. In some instances, there is also an “end run” indicating the end of the input file.

A simulation may also generate its own traffic. Thus, the simulator producing a “tpSendFlow.txt” file, and feeds this data to another simulation becomes a stimulator itself. A simulator may also read a tpSendFlow.txt file while producing its own.

A version in which the flow is via sockets rather than files is available. The key challenge is negotiation of the socket port numbers, as those may conflict if multiple users are on the system.

2.4.2. Data collection to graphs via sockets and agents

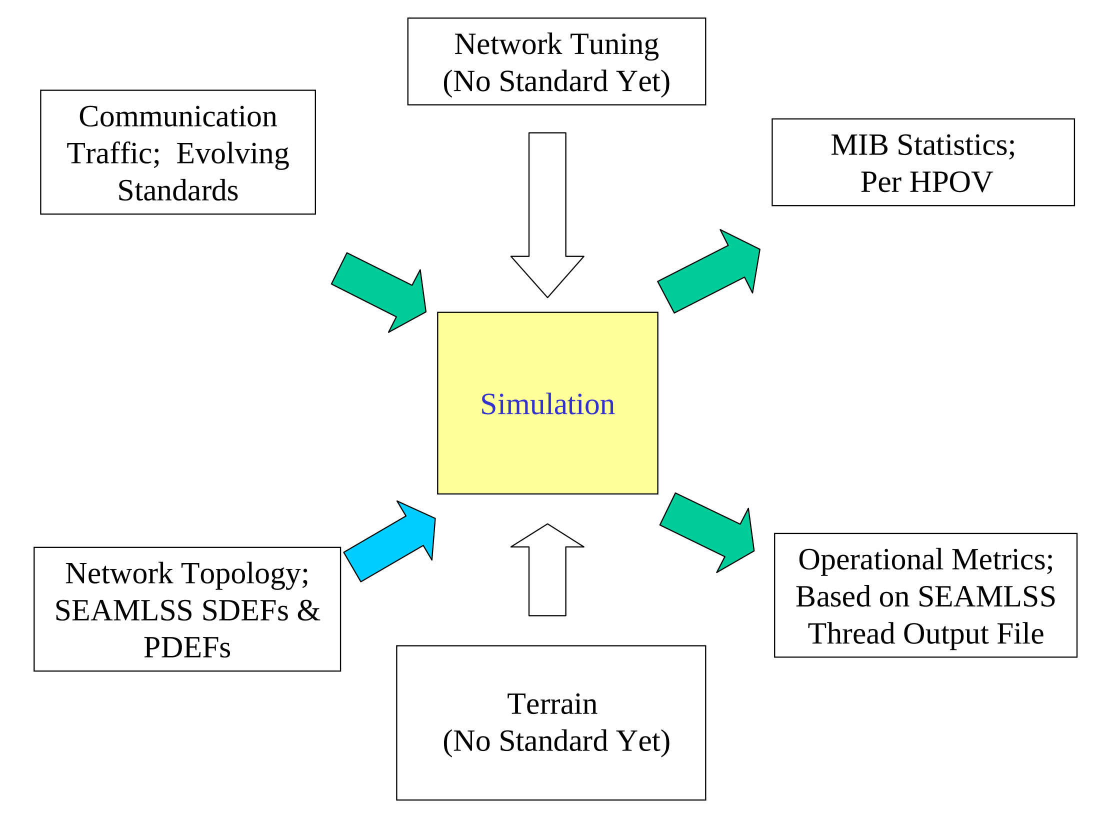
Data collection is output from each simulation as files to the sub directory "out_data" for that simulation. The data is in two formats, binary and American Standard Code for Information Interchange (ASCII) text files. A simulation can write either, both or neither of the formats.

When a graph is launched from the OSC main window, it, in turn, launches one or more agent processes. An agent is tied to a simulator for that particular data type. The agents look for the output files needed by the graph and transfer the data via sockets to the computer at which the graph process is running. The graph process reduces the data and creates the display. The simulation engine need only create the files that the agent reads.

2.5. Adding a Simulation Engine

2.5.1. Meeting the interface requirements

The interfaces required for inserting a simulation include both input and output specifications are shown in Figure 2-12.



Standard interfaces for plugging a simulation into the OSC framework. The standard interfaces in green are required now. They emphasize the real time data feeds. The simulation must read and write these flows to support real time. The standard flow in blue uses the SDF/PDEF files format from the NetWars/SEAMLSS efforts.

Figure 2-12. OSC Framework Interfaces

For each of the data flows there may be more than one format. For instance, the raw statistics are in both ASCII and binary data formats. All of the standards to date have at least an option to work via files. Using files, one process writes the file while one or more other processes read the file -- concurrently.

2.5.2. Adding a launch capability

To launch a new simulation engine from the OSC main interface, an application script named "app_start.csh" for executing that application must be generated and put into the "scripts" directory. Its location is a "script" sub-directory under the directory name specified in the administration menu for the application. The script will be called for any "stim" or "sim" using that application when a user specifies "execute configuration" from the main interface.

The script is intended to be generic, however, at some point the script must contain the specific reference to the application to be launched. The checking of the input arguments is generic as is the removal of data from previous runs. The script simply facilitates remote management of directories on distributed disks. The application specific part is the launch of the pre-compiled simulation for the scenario, on the assumption that the simulation is pre-compiled. For a simulation engine that does compile the simulation engine, the script should include whatever is necessary to build the simulation.

When the script is called, one of the arguments handed to the script is the name of the working directory containing the information about what simulation is to be run and the root directory for all of the outputs from the simulation. When the script exits, OSC will check the exit condition.

To merge the build process for the simulations with the execution of the simulations, the scripts for all simulation engines can be modified to build the simulation before execution.

3. Overview of Specific Accomplishments

As a way of providing a summary of the major accomplishments achieved by SAIC during the NMS project in one location, these were taken from the quarterly summary reports.

- Participated in the DARPA PI meeting on November 9-10, 2004. Demonstrated web-based implementation of the NMS Remote Startup and Shutdown. Briefed on NMS web-based integration strategies.
- Completed requirements analysis, design and implementation of remote startup and shutdown of simulations for Global Information Enterprise Simulation (GIESim) using Remote Procedure Calls (RPC) style (query/response) SOAP based message exchange. A white paper was delivered to Dr. Kumar on September 28, 2004.
- Migrated all OSC APIs to be platform independent and delivered to Air Force Research Lab (AFRL) for integration into GIESim on July 27, 2004.
- Began working with Georgia Tech at Dr. Kumar's request to provide integration services for Georgia Tech, University of Maryland and the University of Southern California simulation projects. Leveraged upon work done for AFRL on the remote startup and shutdown of simulations.
- Conducted requirements analysis and defined the necessary modifications to the testbed in support of OSC integration with GIESim.
- Completed analysis of the applicability of existing fluid modeling to the wireless environment. The key gap was identified in the barrier to using fluid modeling to in the modeling of wireless networks.
- Participated in the January 2004 PI meeting. Supported the Joint Urban Operation (JUO) demo per Joint Forces Command (JFCOM's) interest. Provided NMS standardized scenario data for over 1000 nodes in an appropriate military command hierarchy for the JUO. Conducted OSC real-time analysis of the data.
- Completed OSC Framework Version 2.5.
- Presented OSC integrated with JAVASIM running "what-if" studies for 300+ nodes at the spring 2003 PI meeting.
- Integrated OSC into SAIC's Public Safety Integration Center (PSIC) testbed, where the OSC shows real-time display of QoS from simulation in the Defense Threat Reduction Agency (DTRA) Consequence Assessment Tools Set (CATS) to show value in homeland security scenarios.
- Performed software development of OSC Framework version 2.5.
- Collected quantitative metrics to apply to Modeling and Simulation program. Results are in the following figure:

Contributions to Key Metrics for applying M&S to DoD Needs.				
Metric	**Improvement**	**Initial Value**	**Current Value**	**Target Value**
Data Collection Bandwidth Requirement	$X10^2$ +	1Mbyte/min/node [3]	2Kbyte/min/node	600byte/min/node[4]
Ad hoc Wireless Simulation Speed 5	$X10^3$+	1500xslower than real time for 100 nodes	2x faster than real time for 100 nodes	2x faster than real time for 100 nodes
QoS and Mission Impact Analysis and Display latency	$X10^3$+	Next day if at all	<1 minute	Achieved.
"What if" studies speed	$X10^3$+	Days to Months	Real-Time	Achieved
Planning speed	X10	Weeds to Months	Days	Hours
Ease of Use of Simulation in Field	X10	3 months training	1 week training [6]	Achieved for now.
Plan Quality	X2	N/A—Value is by example test case		

Figure 3-1. OSC Contribution Metrics

- Developed requirements for OSC framework version 2.5 and for the spring 2003 integration experiments.
- Developed preliminary specification of the demonstration for the spring 2003 PI meeting.
- Presented and demonstrated at the fall 2002 PI meeting.
- Simulation results met/exceeded requirements for the period. Achieved simulation of 100 nodes in twice real-time speed with no loss of fidelity relative to state of the art.
- Finalized version 2 of OSC framework and began work on version 2.5.
- In the development phase of OSC version 2 software, and anticipate entering the testing phase by November 1st, in time to complete system testing in preparation for the November PI meeting.
- Completed scenarios to be used for the November PI meeting demonstration, and generated a larger scenario that will be useful through the remainder of the base program.
- Continued support to the Architecture Working Group.
- Generation and support of a Scenario Working Group for the Future Combat System (FCS) scenario production.
- Generated requirements for versions 2.1, 2.2, and 2.5 of OSC.
- Generated design documentation for version 2.
- Developed system testing procedures for version 1.2.
- Explored more enhanced features of the system for future versions including more QoS metrics, more enhanced displays, adopting display filtering and improved performance.
- In conjunction with DAIDA, generation of new tool to capture network information in real-time and record statistics directly from the new tool.
- Support for remotely executed simulations.
- Support for remotely located simulations.

- Integration of CORALREEF data as input source.
- Integration of Extended Littoral Battlespace (ELB) exercise data as input source.
- QoS calculation adaptations to support limitations of tools.
- Port of OSC to Linux.
- Maintenance of OSC, and support of Georgia Tech integrating Network Simulation (NS) using OSC APIs.
- Extension of the OSC framework to configure and manage distributed simulations.
- Development of a user's manual for the OSC framework software tools.
- Development of an API software package supporting integration of simulation engines.
- Packaging of the OSC framework for use by others on the NMS program.
- Built and studied interface for use of simulation by network managers in the field.
- Implemented a QoS summary calculation for network performance and integrated it with network management interface.
- Implemented a "parallel processing farm" approach to accelerate use of simulation on-line.
- Worked with SSS to integrate QualNet replacing PARSEC in the testbed.
- Integrated and used wireless 3G routing protocols including AODV and DAWN in addition to the previously integrated BellmanFord routing.
- Worked with Georgia Tech to partially integrate the NS efforts with our testbed.
- Worked with Rice University towards integration of their fractal efforts with our testbed.

4. Web Services Based Architecture for Simulation Management and Control – The Way Forward

Modeling and Simulation applications, as with other software systems, are typically developed to address the issues of a specific problem domain and satisfy the needs of a specific user community Modeling and Simulation (M&S) applications

- Are funded and developed by different organizations to schedules that are uncorrelated
- Target different hardware platforms and operating systems and are written in different programming languages
- Have different interface requirements, different input and output data format and content, and may have different constraints on throughput

Any strategy for interconnecting M&S applications must take these realities into account. One such strategy which does is the Extensible Modeling and Simulation Framework (XMSF). XMSF is defined as a set of standards, profiles, and recommended practices for web-based modeling and simulation. XMSF has outlined an extensible framework making use of Web-based services which would allow a new generation of M&S applications to interoperate.

A Web service is a software system identified by a Uniform Resource Identifier (URI) and whose public interfaces and bindings are described using Extensible Markup Language (XML). These definitions can be discovered by other software systems which can then interact with the Web Service in a manner described by its definition using XML messages conveyed by Internet protocols.

XML serves as the payload in Web Services messaging and as the descriptor of Web Services. XML forms a basis for organizations to agree upon and manage shared data within and across communities of interest.

SOAP is a protocol for accessing a Web Service. It is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses.

4.1. Adapting Existing Applications to Web Services

As a practical exercise in applying some of these principles we took an existing modeling scenario (Figure 4-1) consisting of an OPNET simulation along with some federates communicating with a Run-Time Infrastructure (RTI). Our objective was to expose these applications such that a remote user would be able to start them (and pass command line arguments), monitor the running application, and finally stop them - using SOAP messages (Figure 4-2).

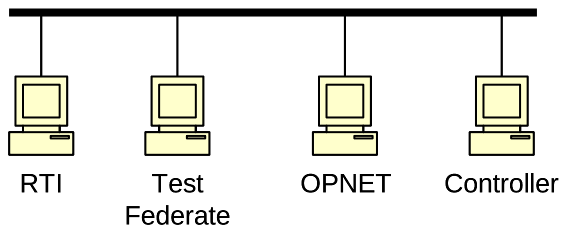


Figure 4-1. Original Configuration

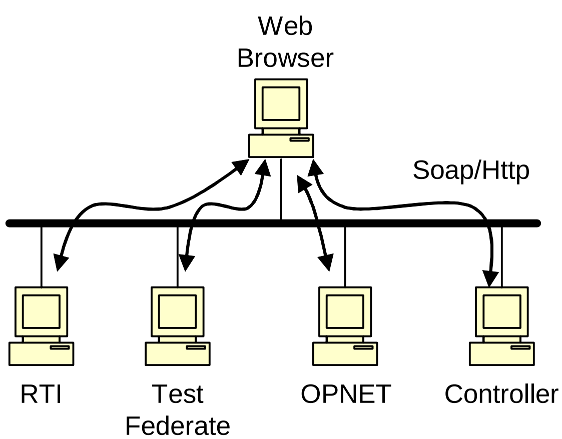


Figure 4-2. Test Configuration

4.1.1. SOAP Server

The integration approach made use of an open source SOAP framework. The Zolera SOAP Infrastructure (ZSI,), is a pure-Python module that provides an implementation of SOAP messaging as described in SOAP 1.1 specification. In particular, ZSI parses and generates SOAP messages, and converts between native Python data types and SOAP syntax. Python is an interpreted, interactive, object-oriented programming language particularly well suited for integration tasks and "gluing" together components written in C (C++), FORTRAN, Java, and others.

When the server is first launched an initialization file is read. The SOAP server is limited to handling requests for the applications that are specified in the initialization file. This has the advantage of only having to modify this file to add a new application for which start/monitor/stop services are to be provided (and then to restart the SOAP server). The windows implementation of the SOAP server uses the Windows Management Instrumentation (WMI) which provides management information and control API to the Windows operating system (Figure 4-3).

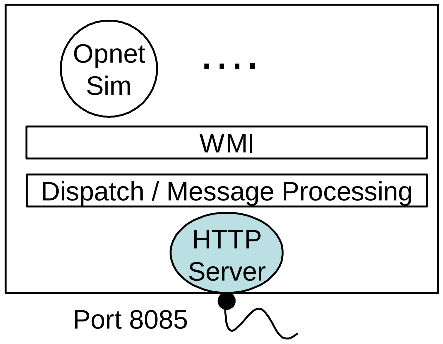


Figure 4-3. Windows Platform Architecture

4.1.2. Web Services Client

The Web Services Client uses the Liferay Portal to provide access to the modeling and simulation environment via the web using a web browser. Liferay is an open source portal and is designed to be application server independent[1]. Liferay will work on lightweight servlet containers like Jetty and Tomcat, or on Java 2 Platform, Enterprise Edition (J2EE) compliant servers like Borland ES, JBoss/Tomcat and WebSphere. Liferay will also work on many operating systems such as BSD, Linux, Solaris, Mac OS X, and Windows. Our portal uses the JBoss/Tomcat Application Server and the Windows Operating System.

Liferay provides many sample portlets as part of the environment. The web services client has been added to the environment as a portlet titled "SimConsole", and can be seen in Figure 4-4. This figure also shows a sampling of the many portlets provided by the Liferay Portal.

The SimConsole portlet interfaces with the Web Services SOAP Server for remote startup and shutdown of several modeling and simulation federates. Also developed for the client was a SimConsole JavaServer Pages (JSP). The JSP code provides the display that is shown in the figures below. In addition to the portlet and the JSP, several XML files were modified to provide the glue code for "SimConsole" action handling, path association, and portlet identification and categorization.

[1] http://www.liferay.com

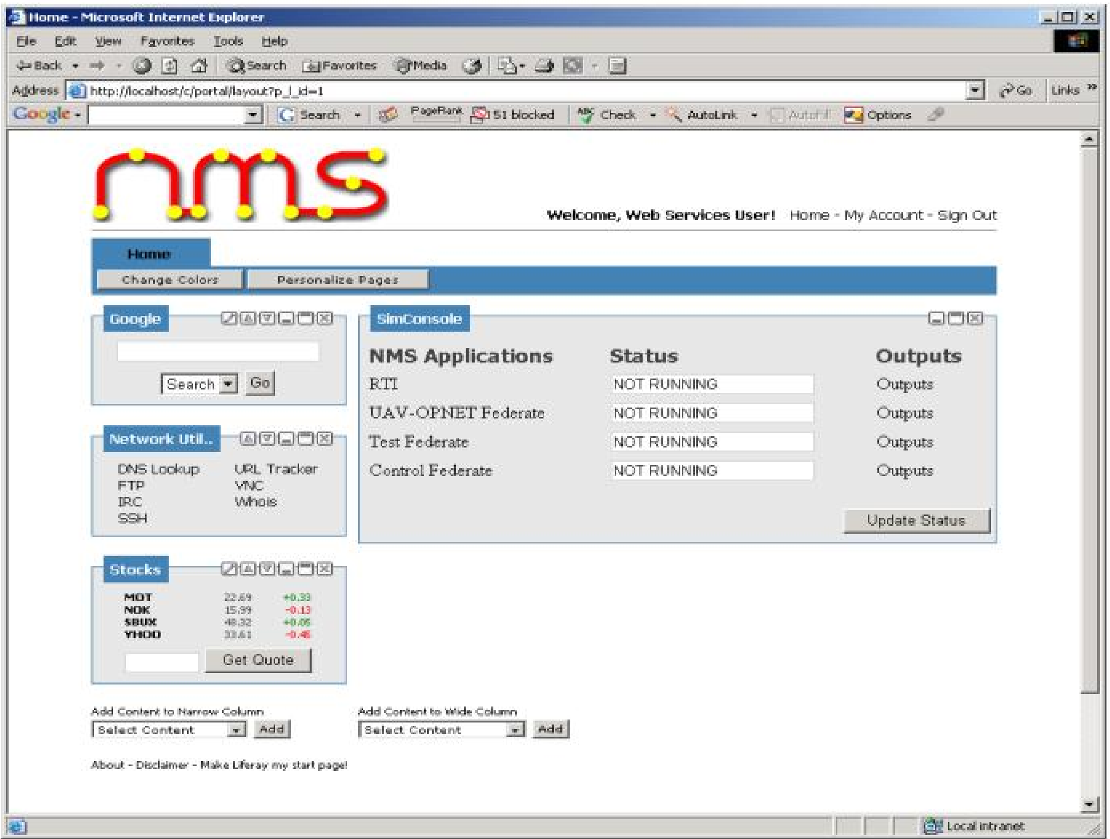


Figure 4-4. Web client - The "SimConsole" Portlet

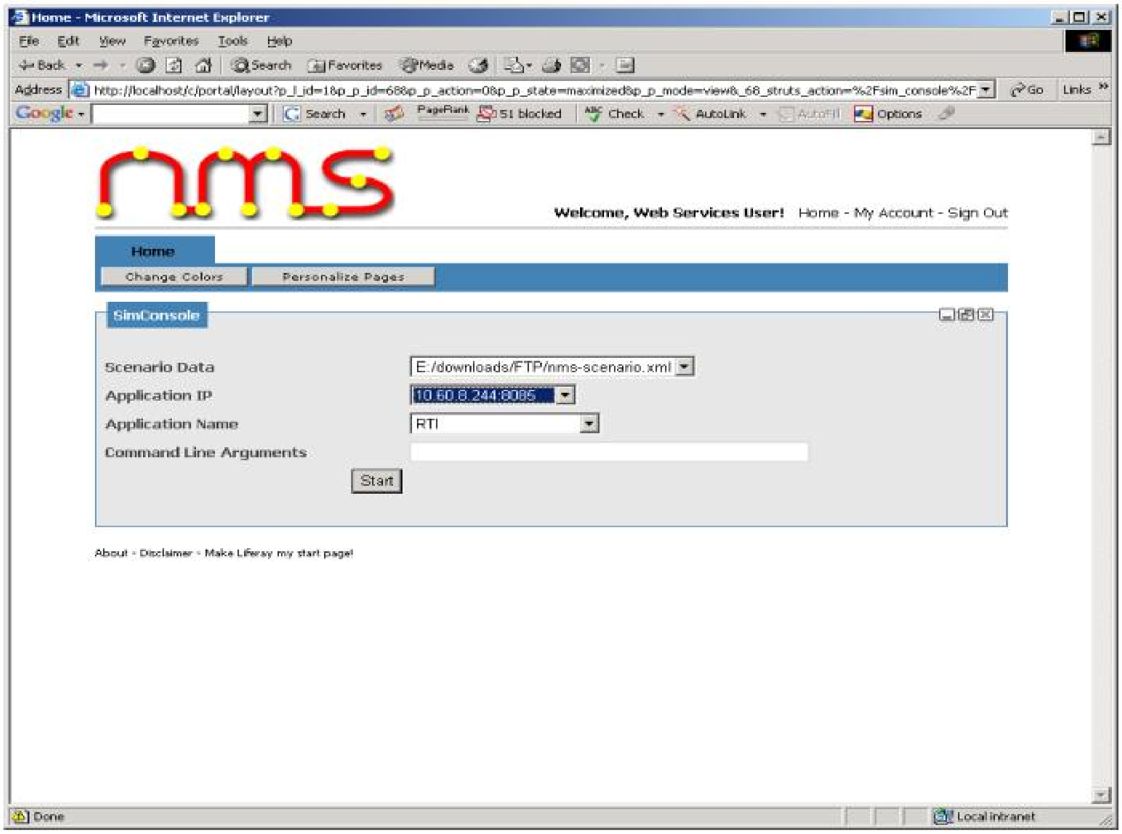


Figure 4.5. Web client – Launching a Federate

Upon selecting "RTI" in Figure 4-4, the display in Figure 4-5 will be shown in the web browser. Figure 4-5 shows the parameter specification for the RTI federate. It allows the scenario input data location and any command line arguments to be specified. It also allows the Internet Protocol (IP) address and communication port number of the Web Service "SOAP Server" application to be specified via the web services client. Selecting the "Start" button initiates the RTI "Start" command to be sent to the SOAP Server from the portlet interface, which in turn starts up the RTI federate where the SOAP Server is running.

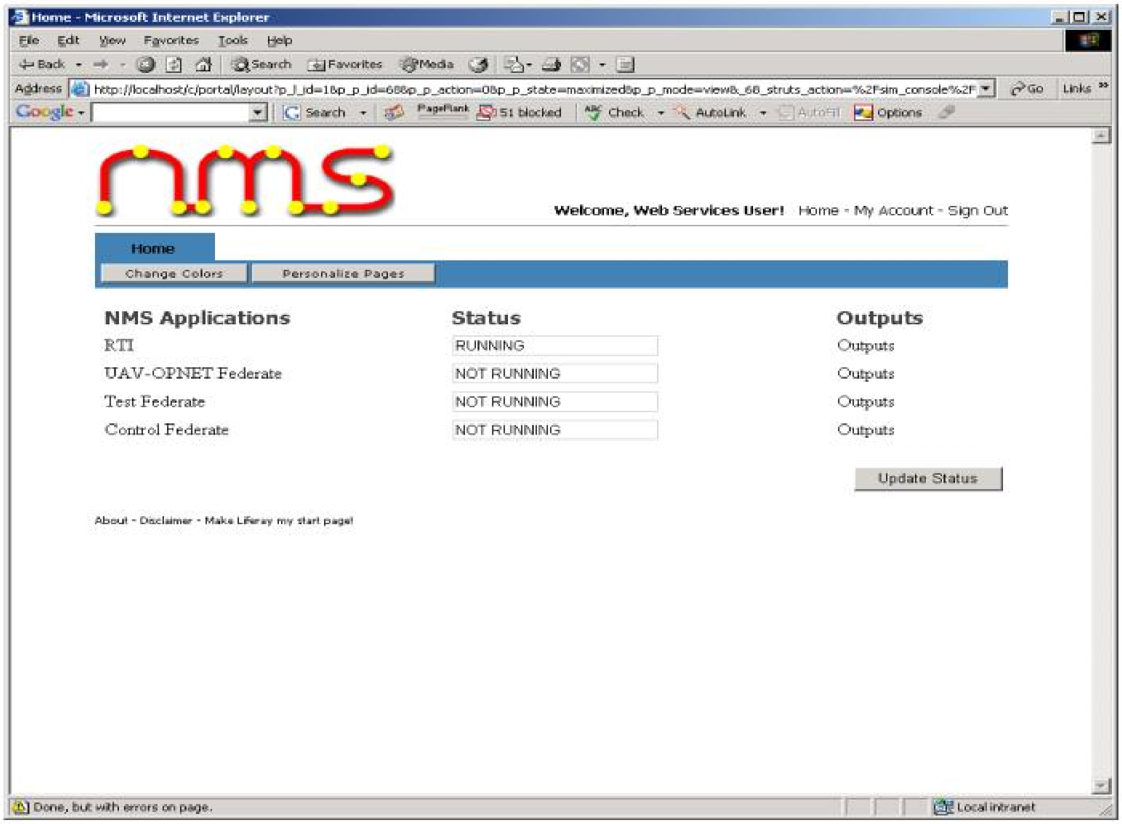


Figure 4-6. Web client – Summary Status

Once the RTI "Start" command was pressed (Figure 4-5) the SOAP Server attempts to start the RTI Federate on the specified IP. The web browser will then return to a status screen of the SimConsole portal (Figure 4-6). This screen indicates the status of each federate included in the modeling and simulation web services implementation. The "Update Status" button will allow for polling to the SOAP Server to find out the latest status of the running processes. This screen will be updated accordingly.

4.2. Web Services Summary

A simple Remote Procedure Calls (RPC) style SOAP message service capable of launching, monitoring, and stopping remote M&S applications from a standard browser was successfully demonstrated at the DARP NMS PI Conference in Monterey California in November 2004. This is really only a first step. To be a useful service a Web Service Definition Language (WSDL) would need to be published and the XML structure of the simulation inputs (like traffic, topology and mobility) as well as the outputs have yet to be defined and standardized.

5. Conclusion

As can be seen from our accomplishments summary in Section 3.0, SAIC has made contributions in many areas of the NMS project over the course of this contract. Our primary focus throughout was on developing a framework which could be used to quickly and easily integrate modeling tools providing a common user interface for configuration and results analysis. OSC has been successfully demonstrated on numerous occasions and integration with modeling tools such as Opnet and QualNet and network sniffers like CoralReef have been accomplished. An integration approach using Web Services has also been explored and shown to be viable strategy for NMS integration.

6. ACRONYMS

AFRL	Air Force Research Lab
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
DARPA	Defense Advanced Research Projects Agency
DoD	Department of Defense
DTRA	Defense Threat Reduction Agency
ELB	Extended Littoral Battlespace
FCS	Future Combat System
GIESim	Global Information Enterprise Simulation
HPOV	Hewlett Packard Open View
IP	Internet Protocol
J2EE	Java 2 Platform, Enterprise Edition
JFCOM	Joint Forces Command
JMTK	Joint Mapping Tool Kit
JSP	JavaServer Pages
JUO	Joint Urban Operation
M&S	Modeling and Simulation
NETWARS	Network Warfare Simulation
NMS	Network Modeling and Simulation
NS	Network Simulation
OSC	Online Simulation and Control
PDEF	Platform Definition File
QoS	Quality of Service
RPC	Remote Procedure Calls
RTI	Run-Time Infrastructure
SA	Situational Awareness
SAIC	Science Application International Corporation
SDF	Scenario Description File
SEAMLSS	Simulation and Evaluation Adaptive Mobile Large Scale Network System
SOAP	Simple Object Access Protocol
URI	Uniform Resource Identifier
WMI	Windows Management Instrumentation
WSDL	Web Service Definition Language
XML	Extensible Markup Language
XMSF	Extensible Modeling and Simulation Framework
ZSI	Zolera SOAP Infrastructure