Jane User Manual

Richard Fujimoto
Rob Morgan
Kalyan Perumalla

College of Computing
Georgia Institute of Technology
February 22, 1999

Table of Contents

1. CHAPTER 1 - INTRODUCTION

Jane is primarily a graphical user interface system to TeD simulations[1]. In addition to providing default graphical controls and displays for TeD models, Jane provides robust and extensible mechanisms for incorporating user-defined model-specific views. Jane is simulator-neutral, making it reusable across different simulators and models. The current Jane software consists of the following:

- A server written in C
- A client written in Java
- A simulation instrumentation API in C/C++.

1.1 Goals and Motivation

The purpose behind the Jane software system is to provide for a number of capabilities, including:

- Remote control of simulations over a network (from a laptop, for example)
- Default graphical views and control for simulations
- Default graphical views for runtime visualization of TeD models
- Easy development of application-specific animations of TeD models
- Scripting for programmatic control of simulations to aid in pre/post simulation analysis.

Jane is written with the intent to be language neutral, and to be easily extendable for model specific animations. Jane supports the use of the simulation as a sub-component for a larger design problem (for example, Network Design & Optimization). It is useful for programmatic "design of experiments" – running multiple sequential or concurrent simulations (for good confidence intervals), and for the analysis of simulation results.

1.2 Server and Client Architecture

The Jane architecture is based on the client-server model. The architecture of Jane has the following properties:

- Independent operation of client, server, and simulator processes
- Communication over any TCP/IP network
- Language-neutral communication protocol between clients and servers (permits the choice of any language — C, C++, Java, Tcl/Tk, etc.)
- A command protocol used for exchanging commands and data between client and server, and between server and simulator.

Figure 1 depicts a snapshot of a sample configuration of the clients, servers and simulations in operation.

[1] TeD is a language designed for modeling telecommunication networks (see http://www.cc.gatech.edu/computing/pads/ted.html).

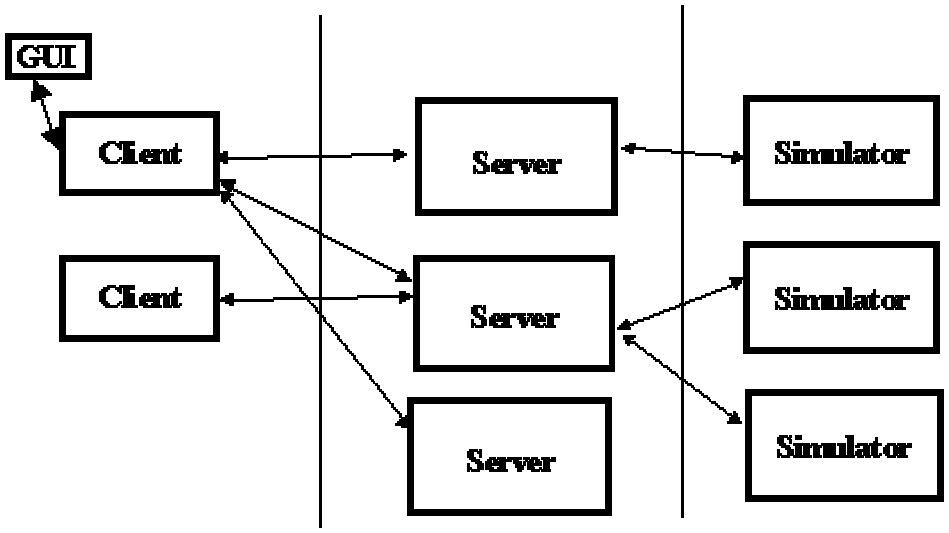


Figure 1: Client / Server Architecture for a simulation session in Jane.

Server-client and server-simulation communication is handled with a network protocol that ensures safety across simulation and model layers sharing a single communication channel. The communication architecture logically enables a reliable sequenced datagram with unlimited packet length.

1.2.1 Client

The Jane client is an extensible, lightweight client written in Java. The client provides a standard set of features to allow for simulation control and monitoring. In addition, the client has an API that allows customization through user objects.

The client performs three types of functions:

- Simulation display and control
- TeD display and control
- Model display and control.

1.2.2 Server

A simulation server written in C is included with the TeD software. The server acts as a broker of messages between the client and the simulation. It also acts as an underlying controller of the simulation, as directed by the client. For example, when the client asks to start a simulation, the server is in fact the module that actually performs the start. Simulation progress and control is reported directly to the server, which then relays this information to the client.

1.2.3 Simulation

The simulator communicates directly with the server. This is done automatically and transparently – the TeD models are unaware of this connection.

2. CHAPTER 2 – USER MANUAL

This chapter serves as a user manual for the client, server, and simulation modules included with TeD/Jane. Included in this section is a list of the API procedures that allow extension of the client module, including detailed instructions for customization.

2.1 Simulation

For information on how to develop and compile TeD simulations, refer to the user manual included with the TeD software release.

The TeD simulation executable can be used, as usual, as a standalone text-console application. The same executable can be used, *without modification or recompilation*, under the Jane client's control. The TeD executable automatically realizes whether it is run as a stand-alone simulation that is run as usual, or spawned under the control of the Jane client/server. The TeD simulations can be controlled using the Jane client, and the graphical feedback of simulation performance can be monitored with Jane.

If model-specific animations are desired, the models have to be instrumented using the instrumentation API of TeD. The relevant instrumentation macros are (please refer to the TeD documentation for precise details of syntax and semantics):

- `EXPOSE()` and `EXPOSE_ARRAY()` – to specify those entity variables which can be accessed by the TeD runtime system for the purposes of display and control.
- `SCHANNEL()` and `SEVENT()` – to send instrumentation events.
- `OUTPUT()` and `OUT()` – to specify event-specific data for an instrumentation event.

2.2 Server

The server is a distinct UNIX process, which runs as a daemon, accepting connections from clients. It caters to requests from the clients, performing operations such as spawning the simulation process, conveying information between the client and simulation, and taking care of the termination of the simulations. Since the server is a process separate from the client or simulation, it can survive both simulation crashes as well as client crashes, and also help the client in terminating a runaway simulation if necessary. The server is sufficiently robust, and performs all the necessary clean up operations upon normal or abnormal termination of clients or simulations.

As a simple security measure, the server enforces a limit of at most 5 simultaneous client connections. If a client tries to connect to the server while the server is servicing 5 clients, the new client connection is refused. The server generates a log of its operations on its standard output/error streams.

2.2.1 Installation

The server is included in the TeD software release under `GTW/SRC/SERVER`. The server executable is automatically generated as part of the TeD installation. The server must be executed on the same machine on which the TeD simulation executes.

2.2.2 How to Run

The suggested way to run the server is as follows:

```
cd GTW/SRC/SERVER
OBJ/ARCH/tedserver |& tee tedserver.log &
```

Use `SGIMP` or `SPARC` for *ARCH* if you are running the server on an SGI or Sparc machine, respectively. The output of the server will be captured in the file **`tedserver.log`**. Initially, the server tries to use the port 12345 for providing its service. In case that port is already in use (which is possible when other users are also running their own servers on the same machine), the server tries to find the next available port. Finally, when the server finds and binds to a vacant port, it displays the chosen port number. This is the port number that must be specified in the client's "Server/Connect" dialog to connect to this server[2]. Once started, the server can be left running (as a daemon) all the time – the server consumes negligible resources when it is idle.

2.2.3 How to Terminate

The normal way to terminate the server is via the **`kill`** command of UNIX shells (using any trappable termination signal, such as **`SIGTERM`**). You can find the process ID (pid) of the server by looking at first output statement in the server log. The server can then be terminated using the command:

`Kill` <*server-pid*>

The server performs any and all cleanup necessary (closing active connections, and terminating active simulations) before exiting.

2.3 Client

The Jane Client is a stand-alone Java program. Portions of the client use the new JFC package produced by Sun Microsystems, informally known as "Swing". The client can be run on any operating system that has installed the most recent Java runtime engine as well as Sun's "Swing" foundation classes.

The client talks to the server via a TCP/IP socket, using a specific communication and command protocol.

2.3.1 Installation

There are 2 parts to the Jane client installation:

[2] To help users control or restrict the clients that can connect their servers, in future we intend to support password-based authentication or use secure socket interface.

- Install the Java runtime environment
- Install the Java Foundation Classes (JFC/Swing)
- Install the Jane classes.

Note: future versions of Sun's Java Runtime Environment will include the Swing foundation classes, thereby eliminating the need for the extra Swing installation.

Since the Jane Client is written in Java, it is necessary to have Java installed on the machine on which the client will be run. A Java installation contains the libraries and the run-time engine needed to run Java standalone applications. The Java support files come in two forms: the Java Runtime Environment (JRE) and the Java Developer's Kit (JDK). The JRE is a runtime-only subset of the Java Developer's Kit, while the JDK allows full development and running of Java applications. Both the JRE and the JDK are available from Sun Microsystems' web site, http://java.sun.com. The JDK is intended primarily for development environments. The JRE is sufficient for running the Jane client, and Sun recommends only the installation of the JRE instead of the JDK for released applications (such as Jane).

Installation of the Jane client software itself consists of copying the class files to the Jane directory created on your system.

Before running the Jane client software in your environment, you should have the ability to run stand-alone Java and JFC/Swing 1.0.2 applications under JDK (or JRE) 1.1.6 or later. If you are already able to run stand-alone Java applications with the Swing Foundation Classes, you can simply install and run the Jane Client software. Otherwise, the JDK (or JRE) 1.1.6 and Swing environment installation is described below.

2.3.1.1 JDK (or JRE) 1.1.6 and JFC 1.1 (Swing) installation

If you do not already have Java (version 1.1.6 or higher) installed, you need to obtain a copy of the Java software and install it. The JRE installation and documentation is available at the following address: http://java.sun.com/products/jdk/1.1/jre/index.html. If the JRE is installed, the JDK does not need to be installed. If you would rather install the full JDK package, it is available at http://java.sun.com/products/jdk/1.1. Follow the instructions given in the installation procedures. In particular, make sure you have the CLASSPATH environment variable set up correctly. After the installation, make sure that you are able to execute the example Java programs included with the JDK (or JRE) package.

Once The JRE (or JDK) environment is installed and operational, the JFC 1.1 must be installed (JFC was previously called Swing 1.0.2). The JFC 1.1 can be found at http://web3.javasoft.com/products/jfc/ which also includes installation instructions.

Once both the JRE (or JDK) and JFC (Swing) environments are installed and operational, the Jane Client software can be installed and run.

JAVA FAQ**:** Sun maintains its own Java FAQ, which answers many general questions about Java in general, including downloading and installing. It is available at: http://java.sun.com/products/jdk/faq.html

JDK 1.1.6: For general documentation concerning the operation and installation of the JDK 1.1.6, refer to: http://java.sun.com/products/jdk/1.1/docs/index.html

2.3.1.2 Client Installation

The Jane client software is available online via http://www.cc.gatech.edu/computing/pads/jane.html. It consists of a set of Java classes and a batch file used for executing the Jane client program. Upon installation, the class files are in the **JANE/CLASSES** directory, and the batch file is `JANE/janeclient.bat` or `JANE/janeclient.csh`.

2.3.2 How to Run

The Jane client is run using the batch file depending on the platform upon which it was installed. Change directory to the `JANE` directory where Jane is installed. Then, on the Windows platform, execute the batch file `janeclient.bat`. On Solaris, execute the C shell script `janeclient.csh`. These batch files invoke the client with default views of the model animation, and hence these can be used to execute *any* TeD model. If you intend to develop your own views/animation of your particular model, and intend to incorporate the views into the client display, see section 2.3.5 on how to incorporate model-specific views.

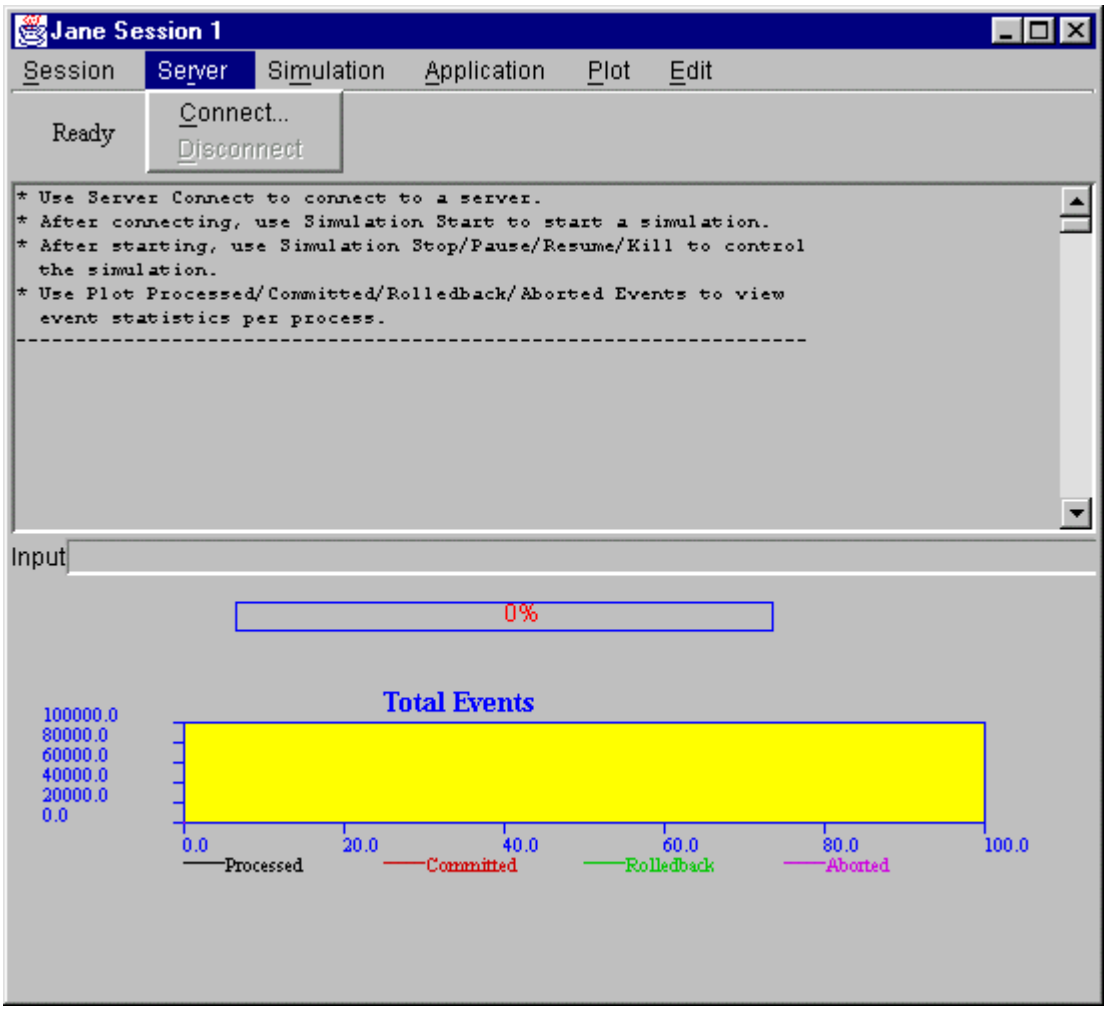
Note: the Jane Client and Jane Server can be run independently. However, the Jane Server must be up and running before the Jane Client initiates attempts to connect to the server.

2.3.3 Example Scenario

The following is a sample scenario of using the Jane client.

- **Start the Jane Client session**

  When the Jane client is started, the initial session screen is displayed as below:

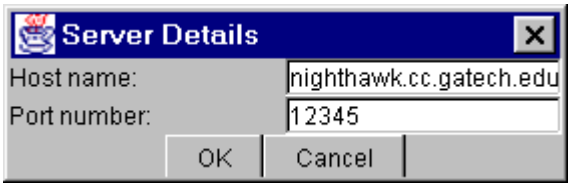


In addition to the menu, the session window contains

- A status line at the top for displaying the status of the client's various operations
- A large output display which will contain the text output of the simulation
- An input field which is used to send standard input to the simulation
- A progress bar which shows the progress of simulation time when the simulation is running
- A plot depicting the total event metrics of the simulation.

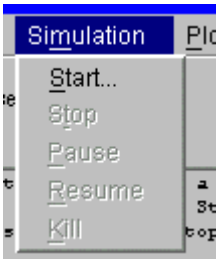
- **Connect to the Server**

The connection to the server must be established in order to perform all subsequent simulation controls. To connect to the server, choose "Server/Connect" from the menu, to bring up the server connection properties dialog:



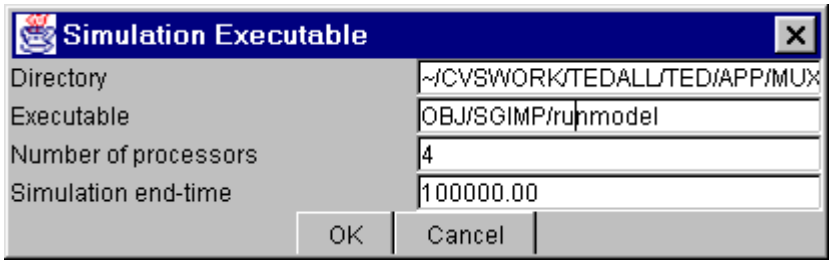
Specify the host name and the port number for the server, then press OK. The Jane client connects to the specified server, and then notifies the user in the status line that the connection was successful and waits for the next command.

- **Start the Simulation**



The user then starts the simulation by choosing from the menu "Simulation/Start": The Jane client prompts the user for the simulation parameters needed in order to begin the simulation. The parameters are

- the *directory* within which the simulation must be executed,
- the *executable name* (path relative to the *directory*),
- *the number of processors* to use for the parallel simulation,
- and the *simulation end time*.



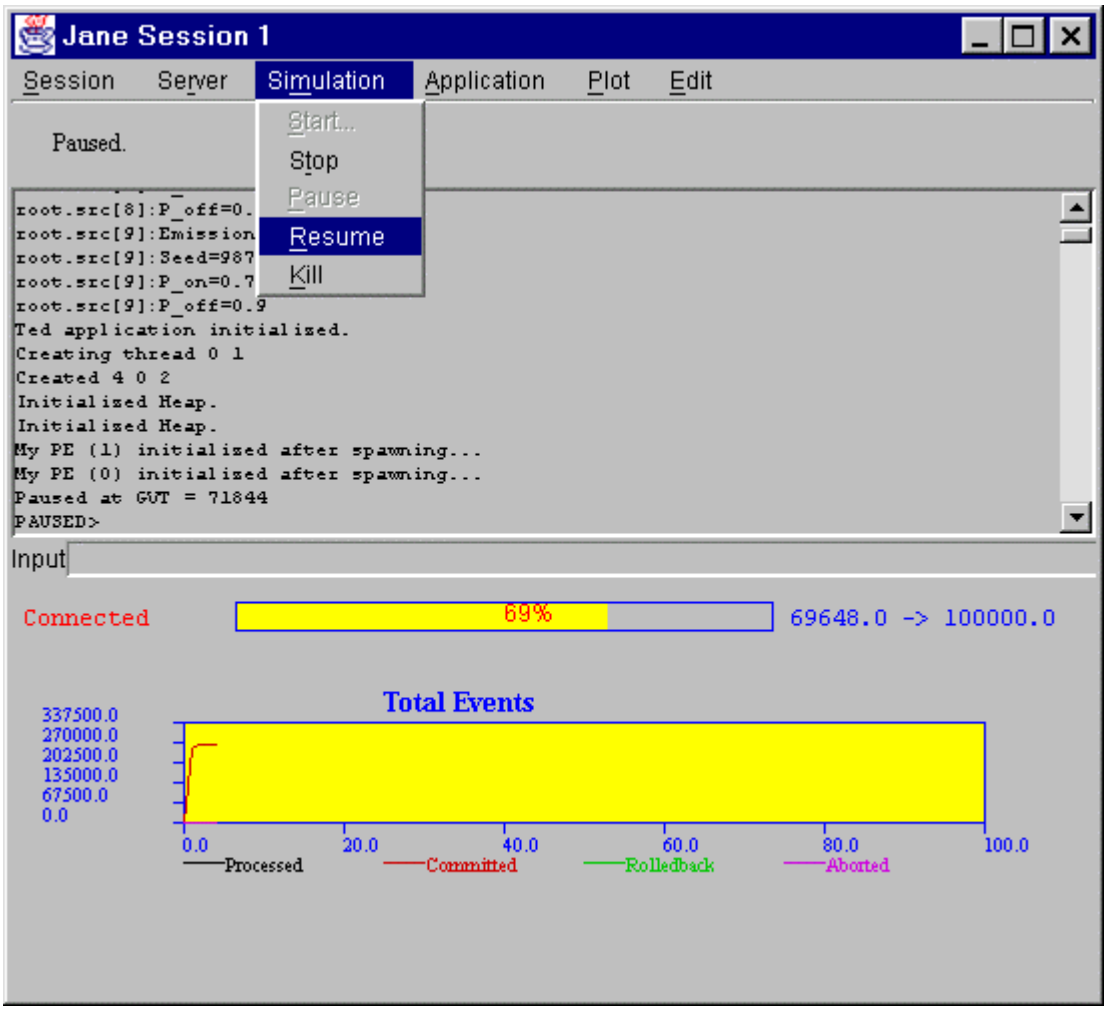
Note that the character ~ can be used to refer to the home directory of the owner of the server process. If the directory name does not start with that character, then the path to the directory will be relative to the directory in which the server was executed on the server/simulation machine.

Once these properties are specified, click OK to start the simulation as specified.

- **Simulation Progress**

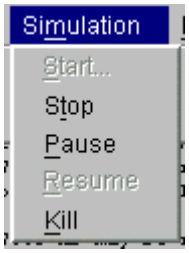
When the simulation starts successfully, the usual textual command-line interface is available to the TeD simulations. Command input for the same can be given using the input text field of the session window. (Try typing **`help`** and hitting ENTER in the Input text field).

The simulation progress can be viewed via the output window as well as the progress plots. The progress bar indicates the percent completion of the simulation relative to the simulation end time specified during the start of the simulation.



The processed, committed, rolled-back, and aborted events of the parallel simulation are monitored on the total event plot.

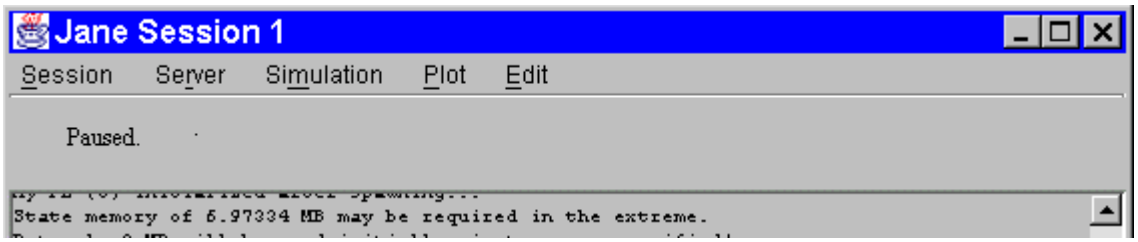
- **Pause, Resume, Stop, Kill Simulation**



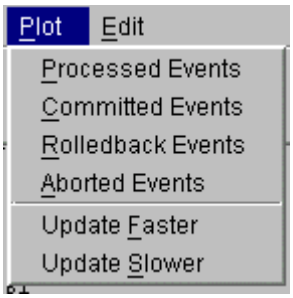
The simulation can be paused, stopped, or killed while it is running. In order to do this, select the respective option from the Simulation menu. The selected signal is sent to the simulation, and when the simulation receives and processes the signal, a confirmation is sent back to the Jane client. Consequently, the user encounters a

small delay in receiving the simulation response after selecting any simulation command.

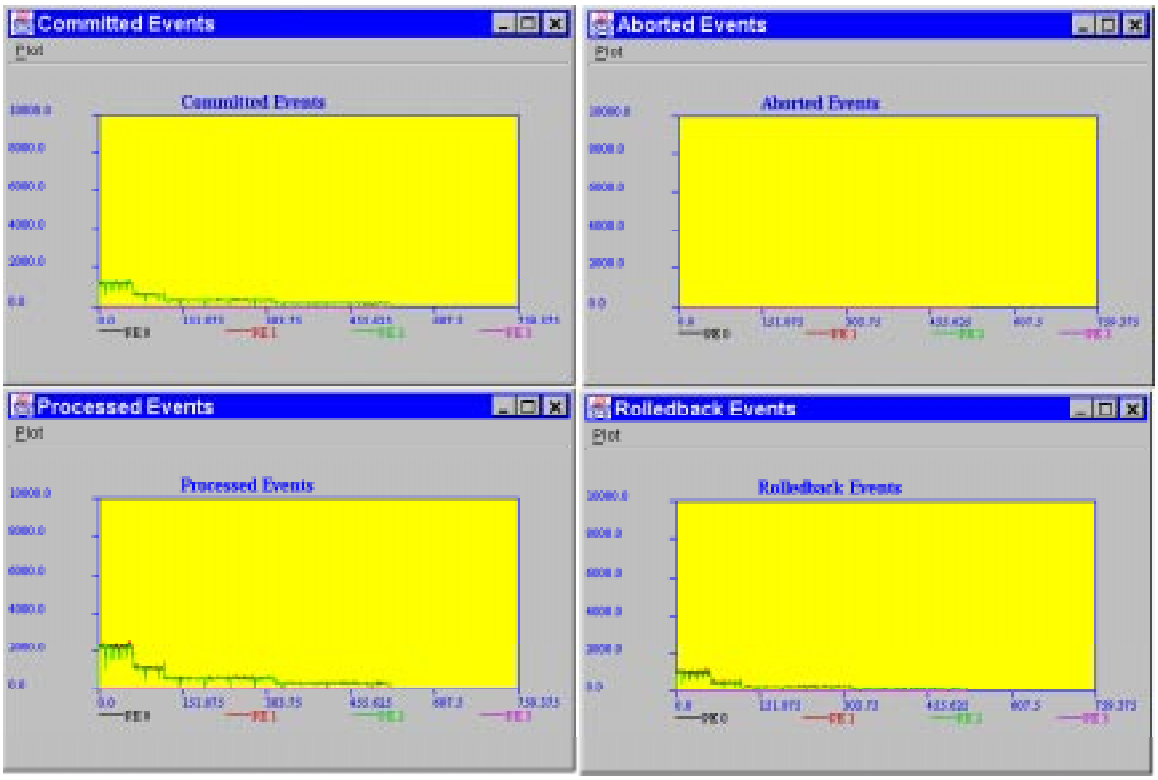
Once the simulation has processed the command, the status line reflects the current state of the simulation. For example, when the "Pause" command is chosen from the Simulation menu, after the simulation pauses, the status line displays "Paused."



- **Simulation Performance Plots**



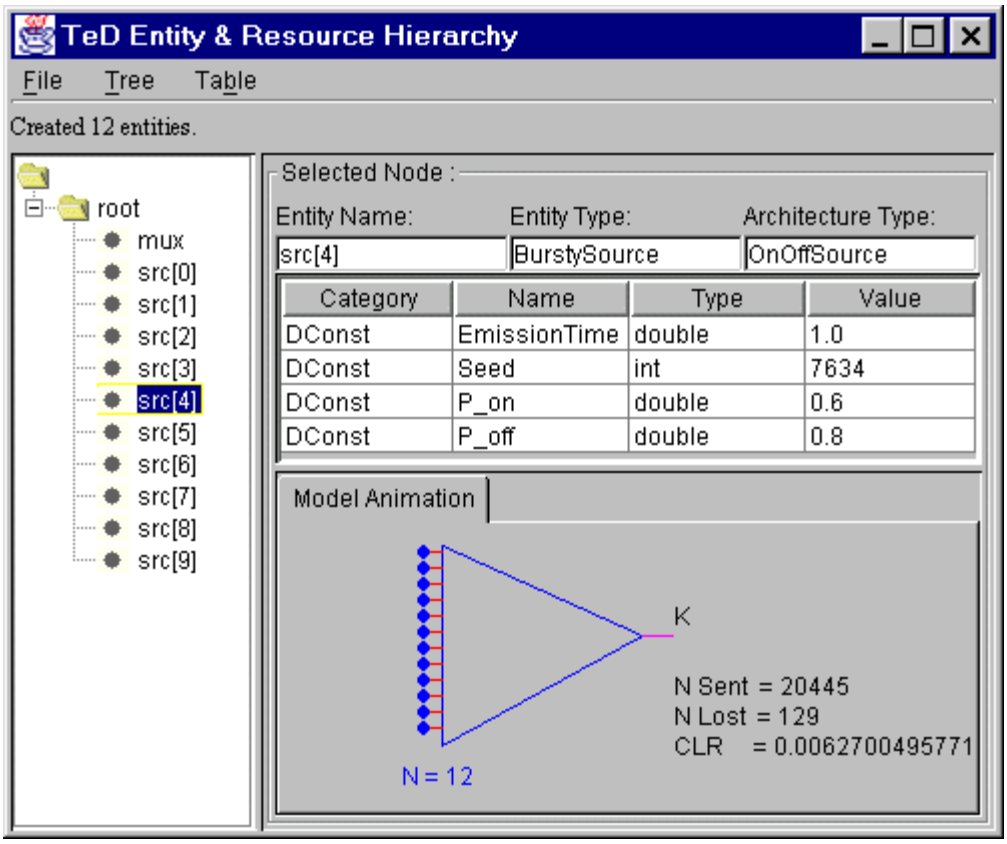
There are four types of plots that can be viewed to inform the user about the simulation. These plots show processed events, committed events, rolled-back events, and aborted events.



All of these plots are available from the "Plot" menu. In addition, the options on the plot menu can be used to increase or decrease the update rate of the plots.

- **TeD Resource Hierarchy Viewer / Editor**

  The Resource Hierarchy is a TeD specific visualization window that is distinct from the main Session window. This window can be displayed using the *Show Application* option of the Session window menu. When a simulation is started, the resource hierarchy displays the entities and their attributes in an easily navigable tree and table layout.



The details of TeD entities in the simulation, such as entity type, architecture type, and deferred constants, are displayed in the window. Clicking each entity in the entity tree shows the corresponding entity's details in the table pane next to the tree.

2.3.4 Features

When the program is first started, it is in an initial state that is disconnected from the server. The following features are the menu options available in the Jane client program.

2.3.4.1 Main Client Window

The initial Window that is displayed when Jane is first run is the main Jane client window. Most of the simulation operations are controlled from this window.

Session

The Jane Client has the ability to have multiple windows open simultaneously, each in independent communication with a server/simulation. This capability is very similar to the ability of a web browser to open multiple simultaneous new browsing windows, with each having the ability to view different web sites.

Session / New…

A new Jane client window is opened in the initial, disconnected state.

Session / Close

The current Jane client window is closed. The closure is immediate, meaning that all the current activity and any currently running simulation control will be lost.

Session / Exit

All open Jane client windows are closed and the entire Jane client program is terminated. Note that, as with the close option, all current operations are immediately ceased.

Server

The server connection is specified, established, and terminated with the server menu.

Server / Connect…

Choosing the Connect option prompts the user for the server name and port number. Once these values are specified, a connection to the server is established and control returns to the user.

Server / Disconnect

The current Jane session is immediately disconnected from the server. Any simulation control is lost.

Simulation

Simulation control can be performed through the Simulation menu options. The options Start, Stop, Pause, and Resume are very similar in operation to many common playback devices found today, such as tape players and VCRs. These controls are not available until a valid connection with a server has been established.

Simulation / Start

The simulation begins when the start option is selected. This option can only be chosen if the simulation is currently stopped.

Simulation / Stop

The simulation ceases when the stop option is selected. This option can only be chosen if the simulation is currently running or is paused.

Simulation / Pause

The simulation pauses in its current state. It is not stopped. This option can only be chosen if the simulation is currently running.

Simulation / Resume

The simulation resumes operation after having been paused, resuming operation from the exact state in which it was paused. This option can only be chosen if the simulation is currently paused.

Simulation / Kill

The simulation is killed and all simulation operations cease. Kill is useful if a simulation “runs away” or does not respond to the stop command.

Edit

The input text field below the output text area can be used to send input to the simulation.

Edit / Clear

All text in the simulation output window is erased, leaving a blank window. This operation does not affect simulation operation; it merely affects the text displayed on the simulation output window.

Edit / Cut

The text in the simulation output window that is currently selected is copied and placed in the clipboard.

Edit / Copy

The text in the simulation output window that is currently selected is copied and placed in the clipboard.

Edit / Paste

Any text that has been placed on the clipboard is pasted into the Input text box. This feature is only useable on the Input box.

Plot

Plot / Processed Events…

A graph is displayed which depicts the number of processed events per unit sample time. The sampling interval can be changed using the options under Plot menu to change the frequency of updates of the plots.

Plot / Committed Events…

A graph is displayed which depicts the number of committed events per unit time.

Plot / Rolled-back Events…

A graph is displayed which depicts the number of rolled back events per unit time.

Plot / Aborted Events…

A graph is displayed which depicts the number of aborted events per unit time.

2.3.4.2 Resource Editor

The resource editor is split into 3 panes: The tree pane, the table pane, and the tab pane. The tree pane contains a tree-style hierarchical list of all the entities. The table pane is a table of entity-attribute pairs for the selected entity in the tree. The tabbed pane contains other key information for the selected entity, including any model-specific animation defined by the user. Clicking on an entity in the tree pane immediately causes the table pane and the tabbed pane to update with information relevant to the chosen entity.

Tree List

The file-like tree structure on the left side of the Resource Editor is the hierarchy of all the TeD entities. The entities can be expanded and collapsed by double-clicking to display or hide entity children.

Selecting an entity, by clicking on it with the mouse cursor, causes the chosen entity's name and attributes to be displayed on the right portion of the Resource Editor.

Table

There are three columns in the table: Name, Type, and Value. Each of the entries under these columns corresponds to the currently selected entity.

Tabbed Pane

There is currently a single tab that displays user-defined model-specific graphic or animation relevant to the currently selected entity. The information on this tab changes depending on entity selection.

2.3.5 How to Incorporate Model-specific Views

If you intend to develop your own views/animation specific to your model and incorporate them into the client display, you can do so by following the MUX view as an example, which is available under the `JANE/VIEWS/MUX` directory. The MUX view is designed for the TeD multiplexer example model (`TED/TED/APP/MUX`). It receives the serial instrumentation events `ATMCellLoss` that are periodically sent out by the multiplexer in the TeD model, and displays the metrics of multiplexer efficiency. Currently, it displays the same view, no matter which entity is selected by the user from the entity tree display. You can run the MUX view by changing directory to `JANE/VIEWS/MUX` and executing `muxview.bat` or `muxview.csh`.

Follow these steps in developing your own model-specific views:

1. Create a new directory under `JANE/VIEWS` called `MYVIEW`, and use that directory for all the next steps.
2. Create a file, called `MyView.java`, and use the following code, adding your own custom graphics code as necessary:

```
import java.awt.Graphics;
import java.awt.Color;
import jane.simulation.protocol.CSimProtocolInputRecord;
import jane.simulation.display.CSimulationSessionUserJane;
import jane.model.dispatch.CTedModelService;
import jane.model.display.CTedModelView;
import jane.model.display.CTedModelViewGenerator;
import jane.model.display.CDefaultJaneEnvironment;
// import any other necessary packages
```

```
class MyView extends CTedModelView
{
     public MyView( CTedModelService s )
     {
          super( s );
     }
     public boolean cmd( String inst_name, int ent_id, String
     evt_name, CSimProtocolInputRecord r )
     {
          // perform some processing… see CMuxView.java for details
          return true;
     }
     public void draw_cell_metrics( Graphics g )
     {
          // draw the updateable portions of the graphics canvas…
          g.drawString( "N Sent  = 30”, 10, 10 );  //example
     }
     public void paint( Graphics g, String inst_name )
     {
          // draw the static portion of the canvas…
          g.setColor(Color.blue);
          g.fillOval( 25 , 25, 15, 15);
          g.drawString( "Your Animation goes here...", 10, 10 );
     }
}

class CMuxViewGenerator implements CTedModelViewGenerator
{
     public CTedModelView create_view( CTedModelService s )
     {
          return new MyView( s );
     }
     public static void main( String args[] )
     {
          CMuxViewGenerator v = new CMuxViewGenerator();
          CDefaultJaneEnvironment.set_generator( v );
          // set other necessary variables…
          String host = "nighthawk.cc.gatech.edu";
          int port = 12345;
          String directory = "~/TED2.1/TED/APP/MUX";
          String executable = "OBJ/SGIMP/runmodel";
          int num_processors = 4;
          double sim_endtime = 10000.0;
          CSimulationSessionUserJane.set( host, port, directory,
          executable, num_processors, sim_endtime );
          new CDefaultJaneEnvironment();
     }
}
```

3. Follow the example code given in the sample file `CMuxView.java` to understand the operation of the classes. The `cmd()` method of the view class is invoked by Jane when an instrumentation event arrives from the TeD model. The `paint()` method is invoked by Jane when either a repaint of the canvas is necessary or if the user clicks on

a specific entity name in the entity tree. It is up to your view to decide what changes have to be effected in your display. The canvas on which your model view can be drawn is available in the view class using the `get_canvas()` method. You are free to create and manage your own windows/frames in addition to the default canvas supplied by Jane. You can use the constructor of the view class to create new window(s) if needed.

4. Copy the `compile.bat` and `muxview.bat` and/or `muxview.csh` from `JANE/VIEWS/MUX` to your view directory.
5. Create a directory called `classes`, and compile your classes using the `compile` batch file.
6. Compile the java file by running "compile.bat" or "compile.csh".
7. Run the batch file `muxview.bat` or `muxview.csh` on Windows or UNIX machine respectively. The Jane application will start as normal, with the difference being that your custom animation is used in the "Entity & Resource Hierarchy" form, available from the menu "Application/Show Application" on the main Jane window.