Agenda

- Purpose: Present and discuss features that will make you more effective in your modeling and simulation work
- “Advanced” features covered in this session
  - Preference dialog box
  - “model” attribute
  - Grouping of global attributes
  - Parametric studies
  - Attribute templates
  - Application delay tracking
  - Checkpoint/restart on Linux
  - Using IBM Purify or Quantify with simulations
  - XML representation of process and node models for versioning
“Advanced” Features Covered Elsewhere

- Graphical debugger: Sessions 1502 & 1503
- Memory tracking/profiling: Sessions 1502 & 1550
- gdb integration on Linux: Session 1503
- Parallel simulations: Session 1551
- 3DNV: Sessions 1942 & 1943
- Wireless and TMM: Sessions 1527 & 1530
- Cosimulation: Session 1532
- System-in-the-Loop (SITL): Session 1933
- Model Support Library: Session 1928

OPNET Preferences

- Used to control the behavior or aspect of many OPNET features
- Frequently used for advanced features to tweak things
- 11.5 Preferences dialog box
“New” 12.0 Preferences

- Revised Preferences dialog box

- Search in names or values
- Used to show table with old and new names
- Hierarchical display by groups or origin
- “New” name
- “Old” name still available. Use this in .ef and on command line

“model” Attribute Now Advanced

- Attribute rarely changed once node created
  - Changing model for a Standard Model node after other attributes have been set can introduce inconsistency issues
  - Only shown in “Advanced” attribute view

- Warning when trying to change its value
- Use node_model_change_warning preference to ignore warning
1574 Modeling and Simulation Using Advanced Modeler Features

Grouping of Global Attributes

- Define group in process model using Interfaces > Global Attributes

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Value</th>
<th>Type</th>
<th>Unit</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic Scaling Mode</td>
<td>Traffic</td>
<td>integer</td>
<td></td>
<td>Background Traffic</td>
</tr>
<tr>
<td>Traffic Scaling Factor</td>
<td>Traffic</td>
<td>double</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>Custom Application Tracing</td>
<td>Applications</td>
<td>integer</td>
<td></td>
<td>Do Not Export</td>
</tr>
<tr>
<td>Application Tracking</td>
<td></td>
<td>string</td>
<td></td>
<td>(Primary/Model Directory)</td>
</tr>
</tbody>
</table>

- Attributes are grouped in the Configure/Run DES dialog box

Statistics and Parametric Studies

- The main purpose of Modeling and Simulation
  - Modeler 12.0 significantly enhances statistic collection and display
    - Enhanced “Choose Individual DES Statistics” dialog box
    - Revised Simulation Sequence Editor
    - Distribute simulations
      - Execute multiple concurrent simulations on a grid of workstations
    - Create parametric graphs easily in revised Results Browser
Selecting Statistics in Project Editor

- **Choose Individual DES Statistics** in right-click in Scenario workspace

- **Summary info pops up without selecting**
- **Data from probe file**
- **Detailed info for selected stat**
- **View/edit vector probe information or right-click on statistic in treeview**
- **Specify scalar probe directly**

Selecting Statistics in Probe Editor

- **Accessed using one of**
  - Menu **DES > Select Statistics (Advanced)** in Project Editor
  - Open ‘**Probe Model**’ file instance
- **Some probes only available in this editor**
  - **Coupled Node Statistics (wireless)**
  - **Custom Animation**
Configure Simulations

- Simulation information stored in ‘Simulation Sequence’ (.seq) file
- Edited using
  - The “running man” toolbar button, or
  - Menu DES > Configure/Run Discrete Event Simulation…
    - Opens “Simple”, “Detailed”, or “Advanced” version of the configure editor
    - Based on Default Configure/Run Dialog Box Mode preference

- Menu DES > Configure/Run Discrete Event Simulation (Advanced), or
- Open ‘Simulation Sequence’ file instance
  - Opens “Advanced” version of the configure editor

“Simple” Configure/Run DES

- Very small set of configurable options
- No parametric support
- Default mode for IT Guru and SP Guru
“Detailed” Configure/Run DES

- Similar to 11.5 dialog box
- Default for Modeler

“Advanced” Configure/Run DES

- Simulation Sequence Editor
  - Edit Attributes
  - Execute the sequence
  - Accepts changes, does NOT execute
Parametric Studies Workflow

- Select a model to study
- Choose the parameter(s): any attribute, preference, or property of model
  - Parameter(s) will vary over a range of values
- Select statistic(s) of interest
- Execute group of related simulations by varying the parameter(s) over their respective value ranges
- Plot parametric graphs showing data evolution as a function of parameter values

Parameters in Configure/Run DES

- Using Detailed or Advanced
  - Specify multiple values for
    - Global Attributes
    - Object Attributes
    - Seed
  - Defines the number of Runs in a simulation Set
- One or more Sets make up a simulation Sequence
"Traffic Growth" Parameter

- Specifies values for ‘Traffic Scaling Factor’ global attribute
- Attribute does not appear in the ‘Global Attributes’ panel

Number of Runs is product of each parameter’s value count
In this case (Traffic Scaling) x (Packet Interarrival Time) = 3 x 12 = 36

Traffic Scaling Factor set to 1, 1.1, and 1.21

Viewing Parameters in Each Run

Same values for all runs
Output files with per-run suffix
Parametric values for each run
Per-Run Suffix for Simulation Output Files

- All standard outputs are suffixed with -DES-<run#>
  - .ov, .ef, .ah, .ot, .3dnvhist, .desinfo
  - Single run outputs are suffixed with -DES-1 systematically
    - Includes .ef, i.e., command-line sim likely to look like
      * op_runsim -net_name project-scenario1 -ef project1-scenario1-DES-1

- Custom output files should be as well to avoid conflicts between runs
  - Base name (“<network>-DES-<run#>”) obtained using
    ```
    char* base_output [BIG_ENOUGH_SIZE];
    op_sim_info_get (OPC_STRING,
        OPC_SIM_INFO_OUTPUT_FILE_NAME, base_output);
    ```
  - Or
    ```
    int len;
    char* base_output
    op_sim_info_get (OPC_INTEGER,
        OPC_SIM_INFO_OUTPUT_FILE_NAME, &len);
    base_output = op_prg_mem_alloc (len+1);
    op_sim_info_get (OPC_STRING,
        OPC_SIM_INFO_OUTPUT_FILE_NAME, base_output);
    ```
  - Then use that for your own file names

Parameters Across Multiple Sets

- Only available in the Simulation Sequence Editor
  - Display the variations between runs
  - Allow selection of which set(s) or run(s) to execute
  - Editing simulation set displays detailed Configure/Run DES dialog box

![Simulation Sequence Editor](image)
Parameters Across Multiple Scenarios

- Execute simulation(s) for several scenarios in one or more projects
  - Vary anything you want in these scenarios
  - Require some scheme to produce appropriate scalar value for each scenario
    - Number of nodes, QoS arbitrary “index” value, etc.

DES Execution Manager

- Displayed whenever there is more than one run to execute
  - Summary info for each run as it is executed
- Simulation progress
  - Only dialog box shown if there is only one run
  - Can be shown for any executing or executed run in the sequence
Distributed Execution of Simulation Runs

- New in 12.0: Ability to launch runs on remote machines

Distributed Execution Workflow

- **op_des_server** process running on remote host(s)
  - All hosts access common set of model directories
  - Set shared file access via network file systems like NFS or SMB
  - Use local or shared installation of OPNET core libraries and programs

Remote execution workflow

1: Launch request
2a: Failure
   - Too many sims
   - Unable to launch
2b: Launch DES
3: DES/GUI Comms
4: DES completed/failed
5: DES completed/died

![Diagram of distributed execution workflow]
Setting Up Distributed Simulations

- Set of preferences to specify
  - Whether the feature is used
  - Which machines to try to contact (can include ‘localhost’)
  - Default port number to contact `op_des_server` on remote machine(s)
  - Number of potential concurrent simulations allowed on a given machine

<table>
<thead>
<tr>
<th>Host</th>
<th>Port</th>
<th>Concurrent Simulations</th>
</tr>
</thead>
<tbody>
<tr>
<td>zaphod</td>
<td>9999</td>
<td>Up to 2 concurrent executions</td>
</tr>
<tr>
<td>number2</td>
<td>7007 (pref)</td>
<td>Up to 1 concurrent execution (pref)</td>
</tr>
</tbody>
</table>

Distributed Simulation Constraints

- Launcher program is NOT a daemon (Unix) or service (Windows)
  - Must be manually started on the remote machines
  - Requires to be logged in as an appropriate user
  - Simulation will run as launched by that user

- Currently limited to same operating system for GUI and launchers

- Require that all launchers share access to at least the directory where run results are generated
  - Best to share a common set of mod_dirs directories
A Minute of Silence for Departed Friends

- **Analysis Panel Editor**
  - No longer a separate editor for “advanced” result viewing
  - Replaced by enhanced Results Browser in Project Editor
  - Analysis configuration (.ac) files still used to store Result panel settings

- **output scalar files (.os)**
  - Scalar data now written in .ov (whether or not vector data also present)
    - Per-run data written in corresponding .ov file
  - No longer accumulating scalar data into a single .os file shared by all runs
  - Pre-12.0 scalar files still accessible for analysis

Results Browser

- **Enhanced version of previous vector data viewer**
  - Right-click in Project Editor and select ‘View Results’
  - Choose DES > Results > View Results
  - Click on View Results toolbar button

![Results Browser diagram](image-url)
Selecting OV files in Results Browser

- By default, focus only on current scenario
  - Allow selection of individual runs

- Can display all OV files for scenarios in the current project

- Can display all OV files in the model directories

Selecting Vector Statistics to Plot

- Statistics found in selected OV files are shown in hierarchical tree
  - Choose to view everything or only statistics present in all selected files
  - Each statistic shows which files provide data
    - Allow you to further refine which to use for a given statistic

Select files of interest

Specify result focus

Display data from all available sources

Display data only from selected source
### Selecting Recorded Scalar Data

- Open Results Browser and select ‘DES Parametric Studies’ tab

- Recorded scalar data
  - 12.0 .ov
  - Pre-12.0 .os

- Presented in more details in the lab

### Viewing Vector’s Scalar Data

- Scalar probe generates scalar value during simulation from raw stat
- GUI can generate scalar values from collected vector data
  - Data based on any vector processing (buckets, glitch removal)

- Data from scalar probes
  - Generated during simulation

- Scalar from OV data
  - Computed on-the-fly
Enhanced Graphic Options

- Exert more control over your outputs with new charting features in release 12.0
  - New line styles like linear symbol, area, multi-color bar
  - Log scale for X or Y axes
  - Optional 3D look
  - Background schemes

Enhanced Graphing Options

- Of course, it depends on your idea of the “perfect” look
Lab 1: Let’s Find the Matisse in You…

- Based on CSMA/CD tutorial
  - Select bus statistics
  - Execute 12 simulation runs for each scenario
  - Generate parametric graphs based on scalar and vector data

Break
Modifying Many Objects at Once: Take 1

- Old approach (still valid)
  - Select a set of objects
  - Shift-click on objects
  - Drag-click around objects
  - Use right-click **Select Similar Nodes**
  - Right-click on one
  - Edit the attributes
  - Check the ‘Apply changes to selected objects’
  - Click OK
  - Voila!

- Great for homogeneous changes

- But what if you want to look at the attributes of many objects at once or make heterogeneous changes…

---

Modifying Many Objects at Once: Take 2

- Right-click on an object and select **Edit Similar <type>**

  - Allows you to view or edit all of the top-level attributes of these nodes
  - Compound attributes open sub-editors

- But what if you want to focus on a subset of attributes…
1574 Modeling and Simulation Using Advanced Modeler Features

Modifying Many Objects at Once: Take 3

- Attribute Template
  - Defines which attributes to show in a table
  - Allows you to flatten compound hierarchy to edit deep-nested attributes more easily
- Select set of nodes
- Menu Edit > Edit Objects Using Template...
  - Pick an appropriate template
  - Click Generate
- Review/modify at your heart’s content
  - Can copy a cell and paste its value to other cells

Creating Templates

- Menu Edit > Edit Attribute Template...
  - Optionally use an existing template as a base
  - See documentation for details on the many options available
### Application Delay Tracking

- **New in 12.0**: Provide the ability to record and review packet story
  - Path(s) taken by each bit, including any retransmissions
  - Time spent in “Network” (transmission) or “Application” (rest) along the way

### ADT in Standard Models

- **Included in Application standard models**
- **Enabled via** `Application:Supported Profiles.Application Delay Tracking` compound attribute

- **Simpler to use** `Edit > Edit Objects Using Template…`
  - Choose `Application:Application Delay Tracking` report
ADT Data Viewing

- Data location specified using global attribute
  - Location can be any directory, even one not listed in ‘modDirs’ preference

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Tracking Dir</td>
<td>-</td>
</tr>
<tr>
<td>Custom Application Tracking</td>
<td>On Hot Spot</td>
</tr>
<tr>
<td>Global Erase Mode</td>
<td>Enabled</td>
</tr>
<tr>
<td>FIT/UDP Suite Mode</td>
<td>Disabled</td>
</tr>
</tbody>
</table>

- Open generated data using
  - DES > Results > View Application Delay Tracking

- Session 1432: Modeling Applications with the ACE Whiteboard
  - Includes lab making use of ADT

Application Delay Tracking Items

- “Application”
  - Package intended primarily for application layer
  - Can be used for any communication layer

- “Application” type
  - Arbitrary string

- “Application” instance
  - For a given type and a given “source” node

- Message
  - “Application” payload associated with a given “Application” instance
  - Correspond to the OPNET Packet at the “Application” layer

- Packet
  - Either the original payload packet (whole message)
  - Anything containing parts (SAR) or whole of the original payload
ADT Data Generation

- New **op_apptrack** Kernel Procedures
  - `create` Initialize package, return handle used in other KPs
  - `type_register` Register an application type, return type ID
  - `instance_create` Create instance of application type for a given node
  - `pk_tag` Flag a payload packet as a message for a given application
  - `pk_reception_record` Indicate that a message has been fully received
  - `instance_complete` Destroy application instance
  - `destroy` Wrap things up

Handful of more esoteric ones

- Simulation Kernel
  - Maintains message association of packet going through SAR
  - Records transmission and reception time for any packet containing bits of a message
  - Writes all ADT information into specified output file

ADT Limitations

- Standard models
  - Only applications and Wireless MACs instrumented at this time
    - Delay in other layers (TCP/IP) accounted for in “Application” time, but no detailed breakdown available

- API
  - Not supported in the parallel kernels at this point
    - KPs just act as no-op
Lab 2: Using ADT APIs

- Enhanced Packet Switching II Tutorial
  - Use `op_apptrack` Kernel Procedures to record packet delays and routes.
  - Use the Application Segment Tracking View to review these delays and routes
Checkpoint/Restart

- Only available for 32-bit sequential simulations on Linux
  - Library and Linux kernel module from Berkeley Labs Checkpoint/restart
- Checkpoint
  - Take snapshot of simulation process image and open output files
- Restart
  - Resume execution at place of checkpoint
- Constraints
  - Better be on same machine (at least same hardware and Linux kernel)
  - Can’t attach to restarted process with gdb
  - No graphical ODB support
    - If simulation originally run with ODB enabled, can use console-based ODB after restart

- Configuration and execution details in appendix of this presentation

Use Cases

- Debugging
  - Run simulation with ODB
  - Checkpoint at specific place or at regular intervals
  - Restart a checkpoint
    - Eliminate loading/execution phase up to the checkpoint
    - Continue debugging at point in simulation during checkpoint
- Parametric study
  - Restart checkpoint with varying random seed
  - Future: expecting to be able to modify attributes
Investigating Memory Issues

- Built-in OPNET features to help resolve...
  - Lack of memory: Memory statistics in Simulation Progress dialog box
    - Check for increasing categories to identify likely leaks
  - Slow execution: OPNET profiling
    - Review time spent in model or kernel space
  - Crashes or strange behaviors: ODB / gdb / Visual Studio / dbx
    - Step through OPNET events or C/C++ statements to investigate
  - All this covered in Sessions 1502 and 1503

- Sometimes not enough ➔ bring in bigger guns
  - Commercial profilers or memory trackers
    - Example: IBM Quantify and Purify

IBM Purify and Quantify

- Purify
  - Track memory access and generate report of suspicious activities
    - Free Memory Read/Write
    - Array Bounds Read/Write
    - Memory leaks
    - Etc.

- Quantify
  - Profile execution and generate tabular and graphical reports

- Both work by instrumenting an application and its libraries
  - Slower execution but potentially a wealth of information
Purify & Quantify on Unix Systems

- Invoked during **linking** phase for an executable
  - `purify g++ -m32 -o myapp myobj.o -lmt`
  - First instrumentation by Purify of the listed object files and libraries
  - Instrumented copies placed in a separate cache directory
  - Then linking of the executable with the instrumented copies
  - Typical output:
    ```
gcc version 3.4.4 20050721 (Red Hat 3.4.4-2)
Instrumenting: crt1.o crti.o crtbegin.o myobj.o
libgcc.a................................. libgcc_s.so.1
libc.so.6 libc_nonshared.a........... cthewd.o cthew.o libc.so.6 libc.so.6.1 Linking
    ```
  - Dynamically loaded libraries instrumented on the fly
  - Same for Quantify

- Run instrumented executable as usual
  - Automatically opens a child graphical Purify/Quantify console with report(s)
Quantify on Unix Systems

Function time list

Detailed function information

Source code information

Purify & Quantify on Windows Systems

- No special linking of executable
- Run executable inside Purify or Quantify tool
  - Instruments executable and associated libraries on the fly
  - Display reports in the tool
  - More details in upcoming lab
Purify Reports on Windows

Quantify Profiling on Windows Systems

Function timing info
Function details
Calling graph
Purify & Quantify of OPNET DES

- Execute simulation independently of OPNET GUI
  - Best to generate a static simulation with `op_mksim`
  - Link executable with debugging information for Purify
  - Allow use of Purify or Quantify APIs in model code
  - See appendix for refresher on `op_mksim` vs. `op_runsim`, and on how to link external libraries with OPNET simulations

- Run the simulation normally first
  - Generate appropriate .ef file(s)
  - Get simulation arguments from `<user_dir>\op_admin\session_log`

Example execution command:
```
C:\PROGRA~1\OPNET\12.0.A\sys\pc_intel_win32\bin\op_runsim_dev
-opnet_user_home C:\ -net_name ACE-local_net
-noprompt -ef ACE-local_net-DES-1 -DESinfo ACE-local_net-DES-1
-exec_id 1 -opnet_port 3053 -parent_pid 848
```

Purify of an OPNET DES Execution

- Purify not aware of OPNET memory optimizations
  - Need to disable use of these optimizations for best results
    - `mem_optimize FALSE`

- Some kernel operations could generate Purify errors or warnings
  - Event handle `{ pointer; id }` in `op_ev_valid()`
    - Check pointer for NULL-ness
  - Compare event id in handle with pointer->id
    - Handle to executed or cancelled event could generate Free Memory Read
Purify & Quantify with OPNET DES on Unix Systems

Need to include purify or quantify in linking phase of executable
- Easy with `op_mksim`
  - Generating .sim executable
  - Set `bind_static_prog` to `bind_gcc_purify` or `bind_gcc_quantify`
- Trickier with `op_runsim`
  - Wrapper invoking `op_runsim_dev`, `op_runsim_opt`, etc. based on `kernel_type` preference
  - Default `op_runsim_dev` and other kernel variants NOT linked with purify or quantify
  - Appropriate object files shipped with OPNET to re-link these apps
    - Instructions in `<opnet_dir>/12.0.A/sys/etc/README.C++`
    - Run simulation directly with the `op_runsim_xxx` executable

`fork()` during simulation loading
- Actual simulation execution proceeding as child process
- Use Purify/Quantify command-line option `-follow-child-processes`

Purify & Quantify with OPNET DES on Unix Systems: Tool Preferences

Settings used at OPNET
- See purify/quantify man pages for detailed information
- Additional options available
- Best specified using 3 shell variables
- Syntax below is for csh/tcsh. Update for your favorite shell.
- Common
  - `setenv PUREOPTIONS "-always-use-cache-dir -use-path-in-build=no -best-effort -lazy-load=no -cache-dir = <your_path> -follow-child-processes"`
- Purify
  - `setenv PURIFYOPTIONS "-pointer-offset=8 -chain-length=32"`
- Quantify
  - `setenv QUANTIFYOPTIONS ""`
Lab 3: Using Purify and Quantify

- Simple illustrations of using Purify and Quantify on Windows
  - Purify reports a few errors in a process models
  - Quantify generates a profiling of a tutorial
- Lab contains instructions for Windows and Linux

Versioning of Models Using XML

- XML representations available for Process and Node models
  - Import/Export available from Process and Node Editors
  - File > Generate XML Code
    - .pr.m ➔ .pr.xml
    - .nd.m ➔ .nd.xml
  - File > Import from XML File
Modeler Wireless Suite for Defense
Modeler Wireless Suite
OPNET Modeler

Documentation References

- Modeler Reference
  - Modeling Concepts
  - Data Analysis
    - Results Browser
  - OPNET Editors Reference
    - Project Editor
      - Discrete Event Simulation
        - Checkpoints
    - System Menus
      - Edit Menu Operations
        - Preferences
- Programmers Reference
  - Kernel Procedures
1574 Modeling and Simulation Using Advanced Modeler Features

Related OPNETWORK Sessions

- 1502 Debugging Simulation Models—Introduction
- 1503 Debugging Simulation Models—Advanced
- 1530 Modeling Custom Wireless Effects
- 1527 Accelerating Wireless Simulations Using Scalability Techniques
- 1551 Accelerating Simulations with the Parallel Kernel
- 1550 Accelerating Simulations Using Efficient Modeling Techniques
- 1528 Accelerating Model Development with OPNET Model Support Libraries
- 1532 Interfacing Multiple Simulators Using OPNET Co-Simulation Techniques
- 1933 Developmental and Interoperability Testing with OPNET System-in-the-Loop
- 1943 Creating Custom 3D Network Visualizations Using Modeler
- 1432 Modeling Applications with the ACE Whiteboard

Take-Away Points

- Revamped workflow for executing simulations and viewing results
- Mechanisms for viewing and changing attributes for a set of nodes
- New Application Delay Tracking API and viewer for additional analysis of packet routes and delays
- Checkpoint/restart functionality for DES on Linux
- Purify/Quantify tools available for serious memory investigations of DES
Appendix

- The following slides are provided for reference
  - Checkpoint/restart details
  - Comparison between dynamic and static simulations
  - Information about linking custom code or external libraries in simulations

Checkpoint Configuration

- Automatically during simulation
  - Set in the Configure/Run DES dialog box
  - Corresponding set of preferences for command-line execution
- Using the ODB command `checkpoint`
- Using `op_sim_checkpoint()` in model code
  - KP does nothing on Solaris, Windows, and 64-bit Linux
Checkpoint Operations

- Stop simulation execution
- Close every open output file known to the Simulation Kernel
- Create `<project>-<scenario>-c-<#>.i1.desckpt.dir` directory
  - Generate process checkpoint image
  - Copy all previously opened output files
  - `<#>` monotonically increasing during simulation if multiple checkpoints
- Reopen previously closed output files
- Resume simulation

Example of Checkpoint in ODB

```
ODB 12.0.A: Event

* Time : 15.000375675226 sec, [15s .000ms 375us 675ns 226ps]
* Event : execution ID (744422), schedule ID (8790355), type (self intrpt)
* Source : execution ID (742962), top.ethcoax_net.stn_3.bursty_gen [Objid=171] (processor)
* Data : code (20)
  > Module : top.ethcoax_net.stn_3.bursty_gen [Objid=171] (processor) [process id: 27]
breakpoint trapped : "stop at time = (15) sec."
odb> checkpoint
Sim info saved at :/op_models/ethcoax_net_mine-scenario1-c2.i1.desckpt.dir/ckpt_sim_info
Context file saved at:/op_models/ethcoax_net_mine-scenario1-c2.i1.desckpt.dir/core_context
Path is:/op_models/ethcoax_net_mine-scenario1-c2.i1.desckpt.dir
About to copy /op_models/ethcoax_net_mine-scenario1-DES-1.desinfo
File /op_models/ethcoax_net_mine-scenario1-DES-1.desinfo has been copied
About to copy /op_models/ethcoax_net_mine-scenario1-DES-1_data.of
File /op_models/ethcoax_net_mine-scenario1-DES-1_data.of has been copied
About to copy /op_models/ethcoax_net_mine-scenario1-DES-1_time.of
File /op_models/ethcoax_net_mine-scenario1-DES-1_time.of has been copied
Wrote /op_models/ethcoax_net_mine-scenario1-DES-1.desinfo with file descriptor: 0
Wrote /op_models/ethcoax_net_mine-scenario1-DES-1_data.of with file descriptor: 1
Wrote /op_models/ethcoax_net_mine-scenario1-DES-1_time.of with file descriptor: 2
Dump generated. We are continuing
odb>
```
Restart Operations

- DES > Restart Discrete Event Simulation
  - Select an existing checkpoint
    - Currently only allows random seed to be modified
  - Launch `op_sim_restart` with the checkpoint information
    - Can be launched from command-line as well

- Example of output from restart

  ```
  op_sim_restart ethcoax_net_mine-scenario1-c2.i1.desckpt.dir/
  <opnet_dir>/12.0.A/sys/unix/bin/op_set_path_restart
  We have been restarted
  Restoring...
  NO PORT
  Path is: /op_models/ethcoax_net_mine-scenario1-c2.i1.desckpt.dir
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Progress: Time (21 sec.); Events (1,200,013)</td>
</tr>
<tr>
<td>Speed: Average (13,111 events/sec.); Current (5,119 events/sec.)</td>
</tr>
<tr>
<td>Time: Elapsed (1 min.)</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
  ```

Static vs. Dynamic Simulations

- Dynamic Simulation
  - Built by `op_runsim_X` (kernel-based)
  - Generate network repository (.nt.so/dll)
  - Executed by `op_runsim_X`
  - Specific to a given network model
    - Cause repository to be generated for each network model
  - External libraries must be dynamically linked (.so/dll)
  - Linking preferences
    - `bind_shobj_prog`
    - `bind_shobj_flags_devel_optim`
    - `bind_shobj_libs_devel_optim`

- Static Simulation
  - Built with `op_mksim`
  - Generate executable (.sim)
  - Executed on its own
  - Not as tied to a given network model
    - Can be used with any network model (IF common object files)
  - External libraries can be statically linked (.a/lib)
  - Linking preferences
    - `bind_static_prog`
    - `bind_static_flags_devel_optim`
    - `bind_static_libs_devel_optim`
  - Easier to use with Purify/Quantify

- Equivalent for everything else
Linking Custom Code/Libraries in DES

- List objects files and general link flags in \texttt{bind\_X\_flags}
- List libraries in \texttt{bind\_X\_flags} or \texttt{bind\_X\_libs}

**Windows**
- Object files and must be compiled with /MD flag
- OPNET built with Intel ICL 9 and Microsoft .NET 2003
- Some C++ mangling changes between Visual Studio 6 and .NET 2003
  - OPNET APIs are C-based, no name mangling
  - Best not to use two versions of a C++ compiler for custom code

**Solaris**
- OPNET built with SunPro 11.0
  - gcc/g++ code OK if dynamic library libstdc++ .so available
  - Use of static library libstdc++.a requires static sim or relinking op\_runsim\_X with gcc
  - cf \texttt{<opnet\_dir>/12.0.A/sys/etc/README.C++}

**Linux**
- OPNET built with gcc 3.4.x
- Model code should be compiled with gcc 3.4 or later
  - ABI change from 3.3.x to 3.4.x