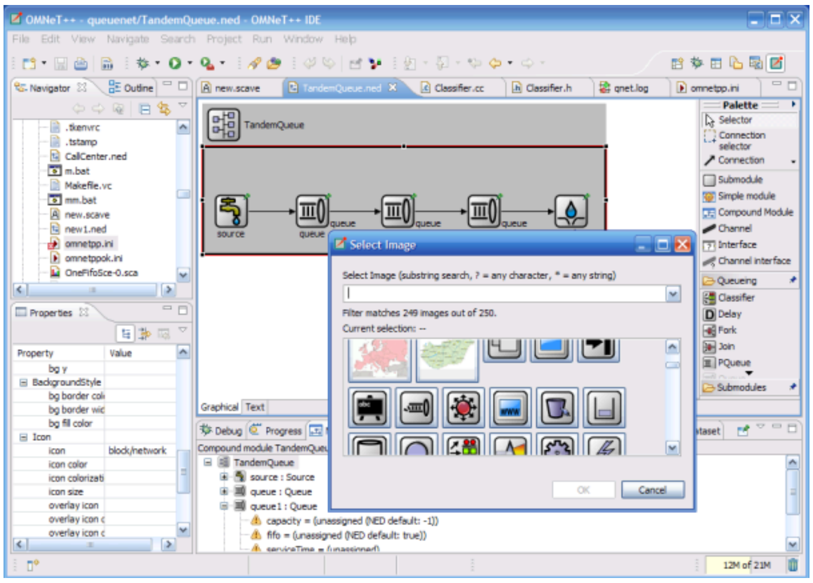
A QUICK OVERVIEW OF THE OMNeT++ IDE

Introduction

The OMNeT++ Integrated Development Environment is based on the Eclipse platform, and extends it with new editors, views, wizards, and additional functionality. OMNeT++ adds functionality for creating and configuring models (NED and ini files), performing batch executions, and analyzing simulation results, while Eclipse provides C++ editing, SVN/GIT integration, and other optional features (UML modeling, bug-tracker integration, database access, etc.) via various open-source and commercial plug-ins.

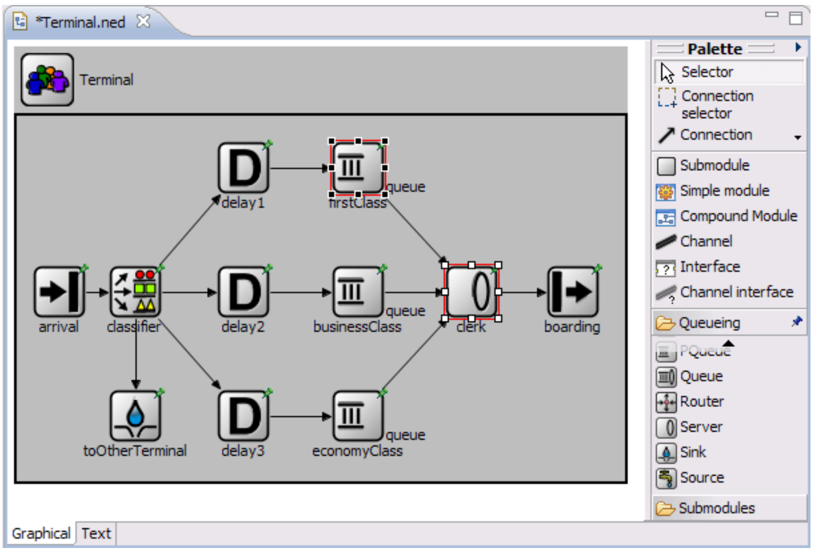
The following screenshots introduce the main components of the OMNeT++ IDE.



The OMNeT++ IDE, editing a NED file

The NED Editor

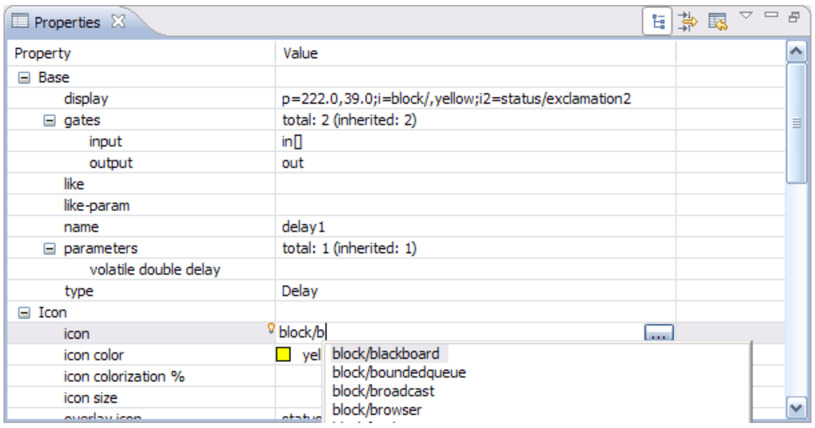
The NED Editor can edit NED files both graphically or in text mode, and the user can switch between the two modes at any time, using the tabs at the bottom of the editor window.



The NED Editor in graphical editing mode

In graphical mode, one can create compound modules, channels, and other component types. Submodules can be created using the palette of available module types. Visual and non-visual properties can be modified in the Properties View, or by dialogs invoked from the context menu. The editor offers many features such as unlimited undo/redo, object cloning, constrained move and resize, alignment of objects, and zooming.

Submodules can be pinned (having a fixed position), or unpinned (auto-layout). Graphical features that can be edited are background image, background grid, default icons (via display string inheritance), icon sizing and coloring, transmission range, and many others.

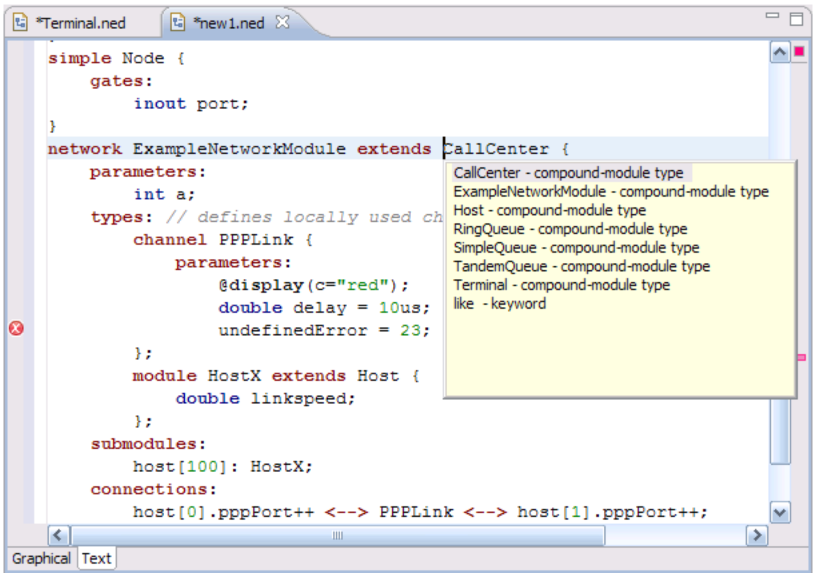

Properties

Property	Value
⊟ Base	
display	p=222.0,39.0;i=block/,yellow;i2=status/exclamation2
⊟ gates	total: 2 (inherited: 2)
input	in[]
output	out
like	
like-param	
name	delay1
⊟ parameters	total: 1 (inherited: 1)
volatile double delay	
type	Delay
⊟ Icon	
icon	block/b
icon color	🟨 yel
icon colorization %	
icon size	

block/blackboard
block/boundedqueue
block/broadcast
block/browser

Properties view

The Properties View lets the user edit graphical and non-graphical properties of objects. Special cell editors facilitate selecting colors, icons, etc. Undo and redo is supported for property changes too. The Properties View is also used with other editors like the Result Analysis editor, where it allows the user to customize charts and other objects.

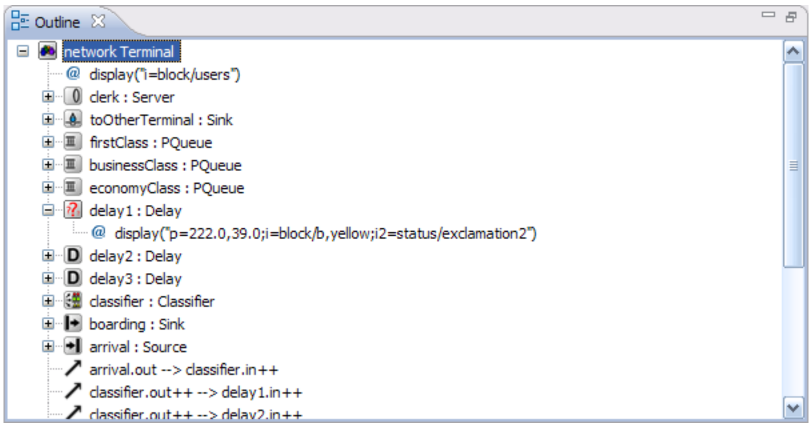


```
simple Node {
    gates:
        inout port;
}
network ExampleNetworkModule extends CallCenter {
    parameters:
        int a;
    types: // defines locally used ch
        channel PPPLink {
            parameters:
                @display(c="red");
                double delay = 10us;
                undefinedError = 23;
        };
        module HostX extends Host {
            double linkspeed;
        };
    submodules:
        host[100]: HostX;
    connections:
        host[0].pppPort++ <--> PPPLink <--> host[1].pppPort++;
```

```
CallCenter - compound-module type
ExampleNetworkModule - compound-module type
Host - compound-module type
RingQueue - compound-module type
SimpleQueue - compound-module type
TandemQueue - compound-module type
Terminal - compound-module type
like - keyword
```

The NED Editor in source editing mode

Text mode lets the user work with the NED source directly. When hitting Ctrl+Space, the editor offers context-aware completion of keywords and module type, parameter, gate, and submodule names. Template proposals to insert full compound module skeletons, submodules, various connection structures, etc. are also available. Documentation of referenced module types can be viewed by hovering over the NED type name. The NED source is continually parsed and validated as the user is typing, and errors are displayed in real time on the left margin. Syntax highlighting, automatic indentation, and automatic conversion from the OMNeT++ 3.x NED syntax are also provided.



```
network Terminal
    display("i=block/users")
    clerk : Server
    toOtherTerminal : Sink
    firstClass : PQueue
    businessClass : PQueue
    economyClass : PQueue
    delay1 : Delay
        display("p=222.0,39.0;i=block/b,yellow;i2=status/exclamation2")
    delay2 : Delay
    delay3 : Delay
    classifier : Classifier
    boarding : Sink
    arrival : Source
    arrival.out --> classifier.in++
    classifier.out++ --> delay1.in++
    classifier.out++ --> delay2.in++
```

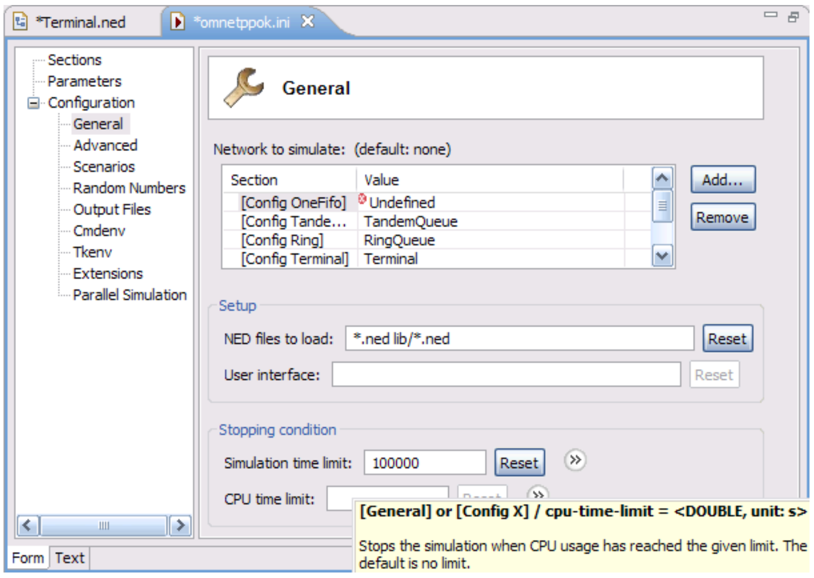
Outline View

The Outline View shows the structure of NED files in both graphical and text editing mode, and allows navigation, as well.

Other Views. The Module Hierarchy View and the NED Parameters View can also be used with the NED Editor (they will display the submodule hierarchy and the parameters of the selected module or submodule) – these views will be described with the Ini File Editor. Further related views are the Problems View (which displays errors and warnings in NED files and other files), and Tasks View (which shows a list of all FIXME and TODO comments found in NED files and other source files.)

The Ini File Editor

The Ini File Editor lets the user configure simulation models for execution. It features both form-based and source editing.



*Terminal.ned | *omnetppok.ini

- Sections
- Parameters
- Configuration
  - General
  - Advanced
  - Scenarios
  - Random Numbers
  - Output Files
  - Cmdenv
  - Tkenv
  - Extensions
  - Parallel Simulation

General

Network to simulate: (default: none)

Section	Value
[Config OneFifo]	Undefined
[Config Tande...	TandemQueue
[Config Ring]	RingQueue
[Config Terminal]	Terminal

Add... | Remove

Setup

NED files to load: *.ned lib/*.ned — Reset

User interface: — Reset

Stopping condition

Simulation time limit: 100000 — Reset

CPU time limit:

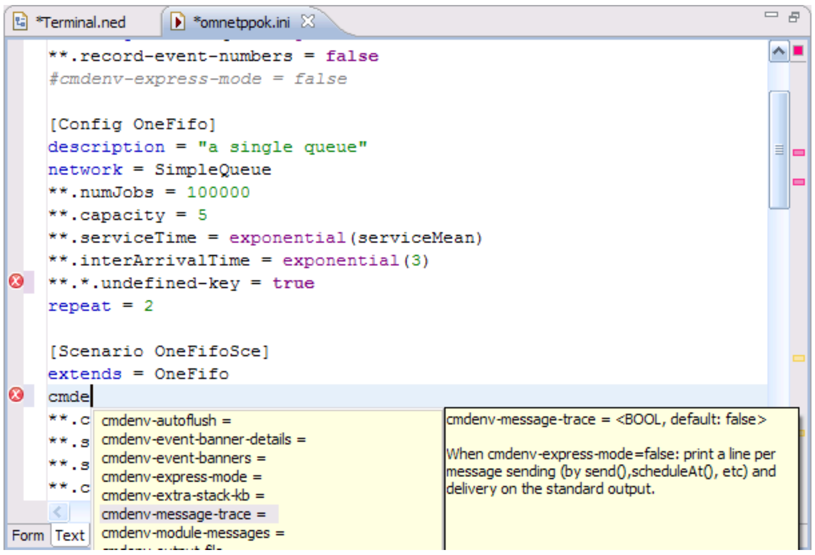
[General] or [Config X] / cpu-time-limit = <DOUBLE, unit: s>

Stops the simulation when CPU usage has reached the given limit. The default is no limit.

Form | Text

Form-based ini file editing

The Ini File editor considers all supported configuration options and offers them in several forms, organized by topics. Descriptions and default values are displayed in tooltips, which can be made persistent for easier reading. The structure of the ini file (sections and their inheritance tree) is also visualized and editable via drag&drop and dialogs. Validation and content assist (Ctrl+Space) are also provided where needed. The editor supports unlimited undo/redo and automatic conversion from OMNeT++ 3.x ini files.



*Terminal.ned | *omnetppok.ini

```
**.record-event-numbers = false
#cmdenv-express-mode = false

[Config OneFifo]
description = "a single queue"
network = SimpleQueue
**.numJobs = 100000
**.capacity = 5
**.serviceTime = exponential(serviceMean)
**.interArrivalTime = exponential(3)
**.*.undefined-key = true
repeat = 2

[Scenario OneFifoSce]
extends = OneFifo
cmde
**.c
**.s
**.s
**.c
```

```
cmdenv-autoflush =
cmdenv-event-banner-details =
cmdenv-event-banners =
cmdenv-express-mode =
cmdenv-extra-stack-kb =
cmdenv-message-trace =
cmdenv-module-messages =
```

cmdenv-message-trace = <BOOL, default: false>

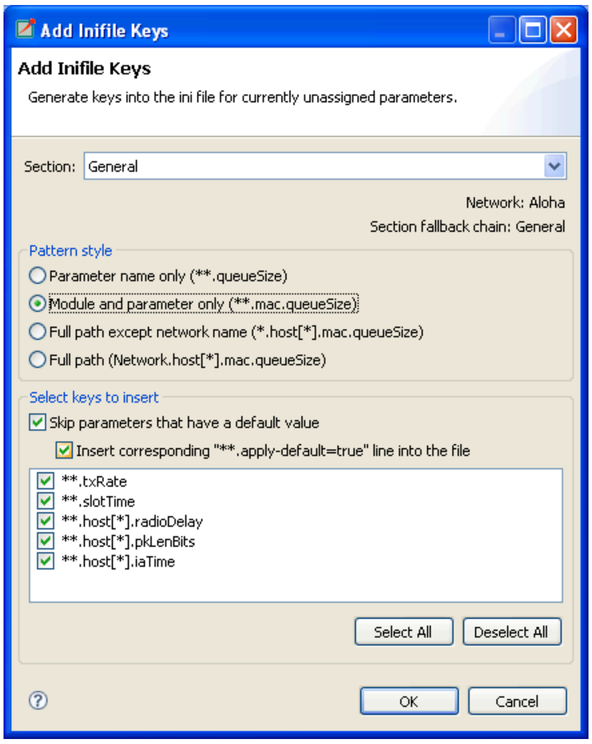
When cmdenv-express-mode=false: print a line per message sending (by send(),scheduleAt(), etc) and delivery on the standard output.

Form | Text

The ini file source editor

The text editor lets the user work with the ini file directly, which is more efficient for advanced users than forms. Unlimited undo/redo, syntax coloring, and completion of configuration keys, values, and module parameter patterns (on Ctrl+Space) are provided. As the user types, the file is continually analyzed and errors/warnings are displayed in real time.

The editor considers all NED declarations (simple modules, compound modules, channels, etc.) and can fully exploit this information to provide diagnostics and assistance to the user. Looking at the *network=* configuration key, it knows the top-level compound module NED type, and internally builds a static picture of the network by recursively enumerating the top-level module's submodules, sub-submodules, sub-sub-submodules, etc., and continually relates this information to the ini file contents. Thus, the editor always knows which ini file keys match which module parameters, which are the ini file keys that are unused (will not match anything), and which module parameters are still open (i.e. have no value assigned yet). It displays this information in tooltips, as warnings, and in various views.


Add Inifile Keys

Add Inifile Keys

Generate keys into the ini file for currently unassigned parameters.

Section: General

Network: Aloha

Section fallback chain: General

Pattern style

- ( ) Parameter name only (**.queueSize)
- (●) Module and parameter only (**.mac.queueSize)
- ( ) Full path except network name (*.host[*].mac.queueSize)
- ( ) Full path (Network.host[*].mac.queueSize)

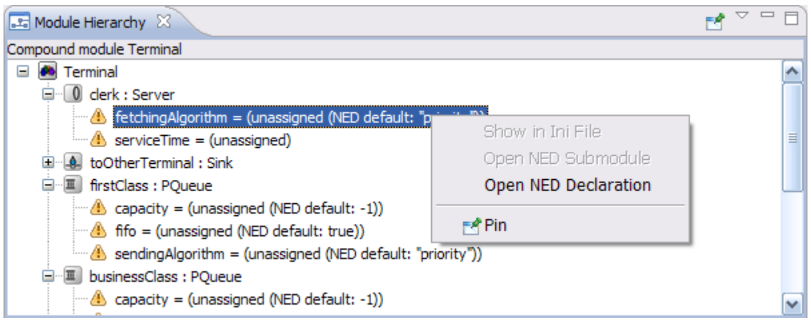
Select keys to insert

- [✓] Skip parameters that have a default value
  - [✓] Insert corresponding "**.apply-default=true" line into the file

- [✓] **.txRate
- [✓] **.slotTime
- [✓] **.host[*].radioDelay
- [✓] **.host[*].pkLenBits
- [✓] **.host[*].iaTime

The Add Missing Keys dialog

To speed up the process of creating ini files from scratch, the editor offers an *Add Missing Keys* dialog. After clicking OK, the editor will insert ini file keys for all open NED parameters. The user will only need to supply the values.


Module Hierarchy

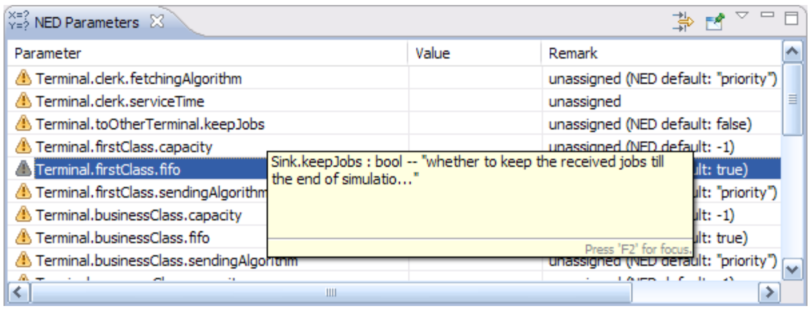
Compound module Terminal

```
Terminal
  clerk : Server
    fetchingAlgorithm = (unassigned (NED default: "p
    serviceTime = (unassigned)
  toOtherTerminal : Sink
  firstClass : PQueue
    capacity = (unassigned (NED default: -1))
    fifo = (unassigned (NED default: true))
    sendingAlgorithm = (unassigned (NED default: "priority"))
  businessClass : PQueue
    capacity = (unassigned (NED default: -1))
```

Module Hierarchy View

The Module Hierarchy View displays the submodule hierarchy of the current configuration's network, together with the module's parameters. It shows how module parameters obtain their values (from NED, by applying a NED default value, or from the ini file). It is able to resolve simple cases where a submodule type comes from a string parameter (NED's *like* keyword). By using the context menu, the user can go to the NED declaration of a module, submodule, parameter, gate, or to the place where a parameter gets its value (in NED or in the ini file).

The View can be pinned to a particular editor to prevent its contents from changing when the user switches across NED and ini file editors.



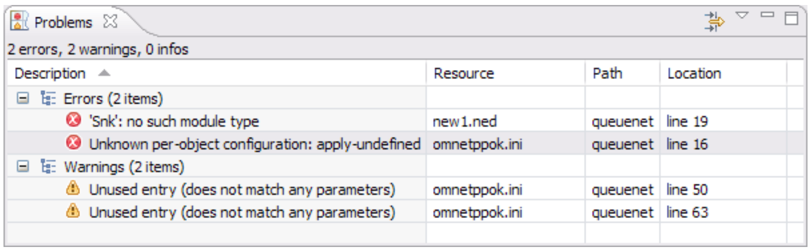
Parameter	Value	Remark
Terminal.clerk.fetchingAlgorithm		unassigned (NED default: "priority")
Terminal.clerk.serviceTime		unassigned
Terminal.toOtherTerminal.keepJobs		unassigned (NED default: false)
Terminal.firstClass.capacity		unassigned (NED default: -1)
Terminal.firstClass.fifo		lt: true)
Terminal.firstClass.sendingAlgorithm		lt: "priority")
Terminal.businessClass.capacity		lt: -1)
Terminal.businessClass.fifo		lt: true)
Terminal.businessClass.sendingAlgorithm		unassigned (NED default: "priority")

Sink.keepJobs : bool -- "whether to keep the received jobs till the end of simulatio..."

Press 'F2' for focus.

The NED Parameters View

By default, the NED Parameters View displays the table of all unassigned NED parameters in the current configuration's network and it can be switched over to show all parameters, assigned or unassigned. For assigned parameters, it shows how they obtain their values (from NED, by applying a NED default value, or from the ini file). The user can open the relevant places in the NED files or the ini file. This View can also be pinned.



2 errors, 2 warnings, 0 infos

Description	Resource	Path	Location
Errors (2 items)			
'Snk': no such module type	new1.ned	queuenet	line 19
Unknown per-object configuration: apply-undefined	omnetppok.ini	queuenet	line 16
Warnings (2 items)			
Unused entry (does not match any parameters)	omnetppok.ini	queuenet	line 50
Unused entry (does not match any parameters)	omnetppok.ini	queuenet	line 63

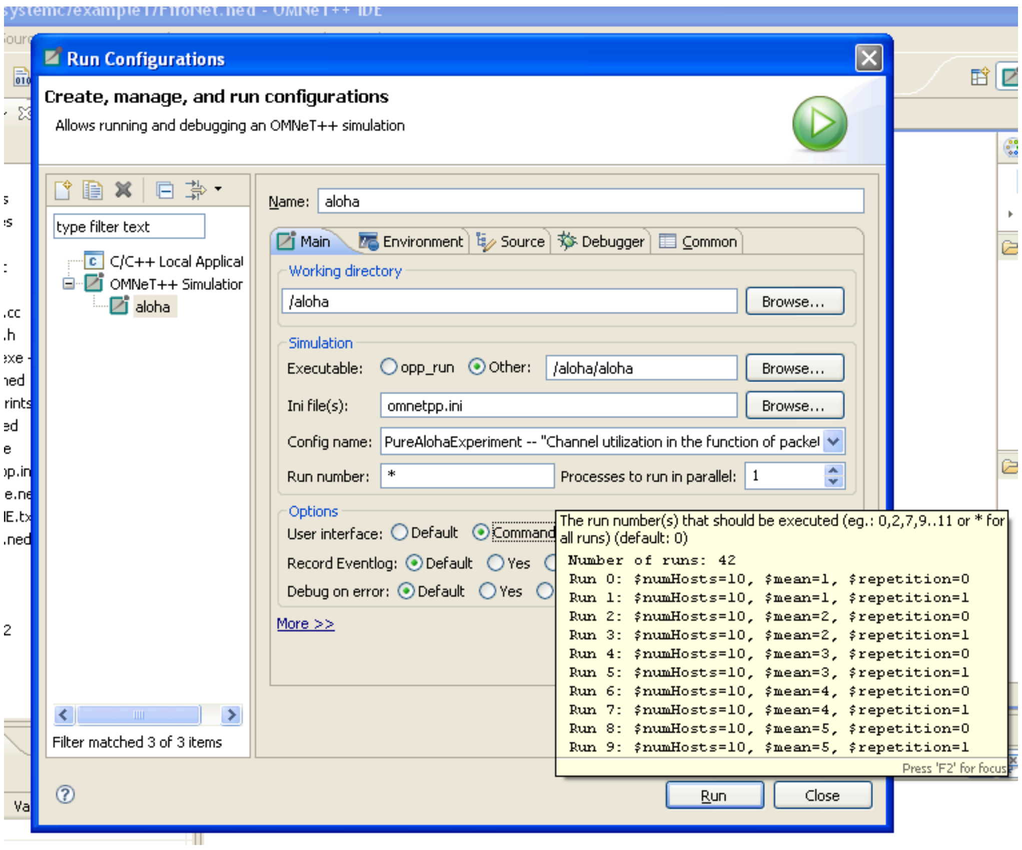
The Problems View

The Problems View presents errors, warnings, and info messages in NED files, ini files, and other source files in a unified manner. Double-clicking on an item opens the corresponding file and goes to the error's location. The view's contents can be filtered in various ways (current file, current project, all projects, by severity, etc).

Other Views. The Outline View displays the list of sections in the current ini file and can be used for navigation as well.

Simulation Launcher

The OMNeT++ IDE makes it possible to run simulations directly from the integrated environment. It is possible to run a simulation as a normal C/C++ application and perform C++ source-level debugging on it. The user can also run it as a standalone application (under Tkenv or Cmdenv) or run batches of simulations where runs differ in module parameter settings or random number seeds (NOTE: an Eclipse-based runtime environment [“IDEEnv”] does not exist yet; it is planned for future releases).



Create, manage, and run configurations

Allows running and debugging an OMNeT++ simulation

type filter text

C/C++ Local Applicat
OMNeT++ Simulatior
aloha

Filter matched 3 of 3 items

Name: aloha

Working directory: /aloha

Simulation

Executable: opp_run [✓ Other:] /aloha/aloha

Ini file(s): omnetpp.ini

Config name: PureAlohaExperiment -- "Channel utilization in the function of packet

Run number: *

Processes to run in parallel: 1

Options

User interface: Default [✓ Command]

Record Eventlog: [✓ Default] Yes

Debug on error: [✓ Default] Yes

The run number(s) that should be executed (eg.: 0,2,7,9..11 or * for all runs) (default: 0)

```
Number of runs: 42
Run 0: $numHosts=10, $mean=1, $repetition=0
Run 1: $numHosts=10, $mean=1, $repetition=1
Run 2: $numHosts=10, $mean=2, $repetition=0
Run 3: $numHosts=10, $mean=2, $repetition=1
Run 4: $numHosts=10, $mean=3, $repetition=0
Run 5: $numHosts=10, $mean=3, $repetition=1
Run 6: $numHosts=10, $mean=4, $repetition=0
Run 7: $numHosts=10, $mean=4, $repetition=1
Run 8: $numHosts=10, $mean=5, $repetition=0
Run 9: $numHosts=10, $mean=5, $repetition=1
```

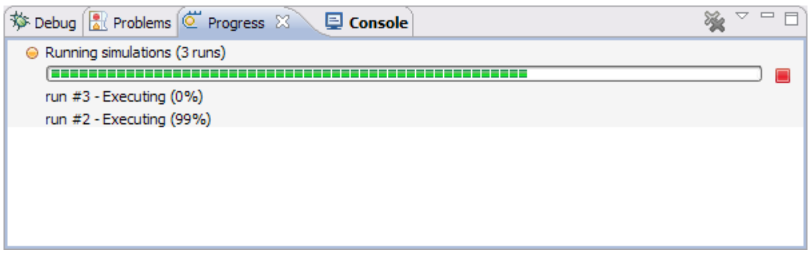
Press 'F2' for focus

The Run dialog showing a simulation launch configuration

In the Eclipse IDE, the various ways of launching a program under development is described by launch configurations. There are various launch configuration types (C/C++ Application, Java Application, etc.). To launch a program for the first time, the user creates an instance of a launch configuration, fills in a form (program name, command-line arguments, etc.), and hits the Run button. OMNeT++ provides launch configuration types for simulations: one extends the normal “C/C++ Local Application” launch type with a *Simulation* tab (for convenient editing of simulation-related command-line options), and a standalone launch type for launching a simulation or running simulation batches. The former can be used for C++ debugging, as well.

The dialog shows a standalone launch type for the QueueNet example simulation model. Batch simulations can be launched by specifying more than one run number in the dialog. Ini files allow the user to define various scenarios iterating over a list of values for certain NED parameters, and/or do multiple runs with different seeds. The launch dialog helps the user select the desired run numbers by showing which runs correspond to which parameters and seed settings.

Multi-processor or multi-core computers can be exploited by specifying that more than one process may be launched (i.e. may be running) at a time.



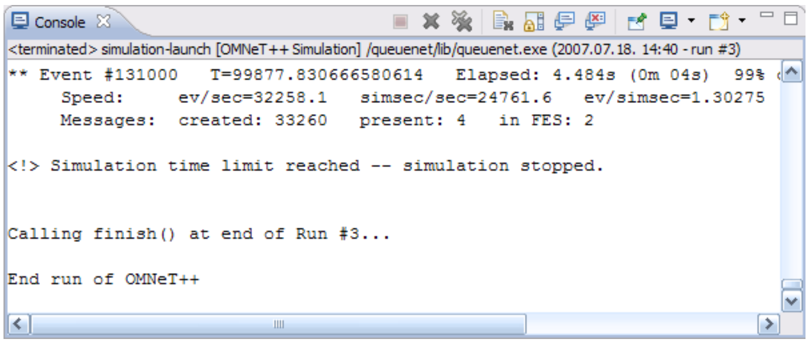
Running simulations (3 runs)

run #3 - Executing (0%)

run #2 - Executing (99%)

Progress View

The Progress View reports the status of simulation execution when you have a long-running simulation, or you are executing several runs in a batch. It is possible to cancel the whole batch operation with a single click, if necessary. Simulations run in separate processes that do not block the IDE, so users can keep working while their simulations run in the background. In the above screenshot, the number of processes to run in parallel was set to 2; run #1 has already terminated (and is no longer shown), run #3 has already launched, and run #2 is still running.

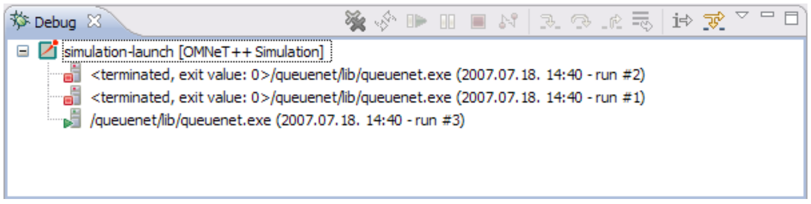


<terminated> simulation-launch [OMNeT++ Simulation] /queuenet/lib/queuenet.exe (2007.07.18. 14:40 - run #3)

```
** Event #131000   T=99877.830666580614   Elapsed: 4.484s (0m 04s)   99%
     Speed:     ev/sec=32258.1   simsec/sec=24761.6   ev/simsec=1.30275
     Messages:  created: 33260   present: 4   in FES: 2

<!> Simulation time limit reached -- simulation stopped.


Calling finish() at end of Run #3...

End run of OMNeT++
```

Console View

Each running process sends its output to a separate console buffer within the Console View, so the user can review the output after a simulation has finished. One can switch between console buffers using the Console View's menu or toolbar, or by double-clicking on a process in the Debug View.



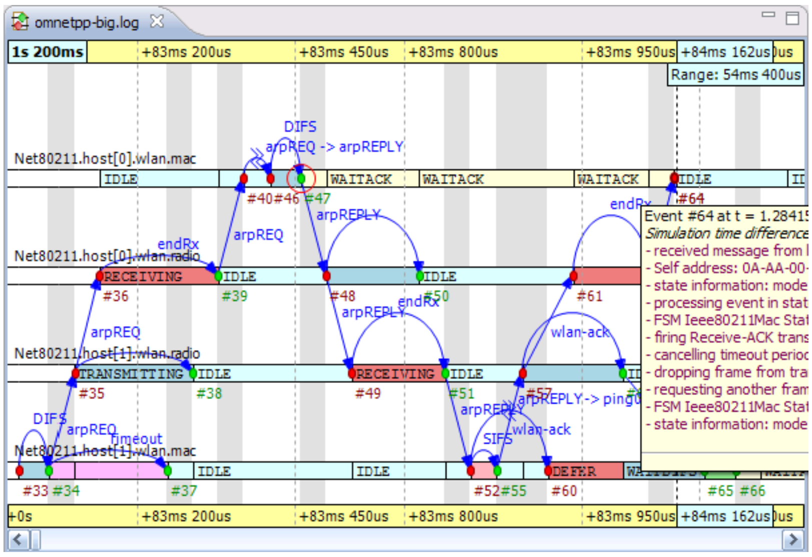
simulation-launch [OMNeT++ Simulation]
<terminated, exit value: 0>/queuenet/lib/queuenet.exe (2007.07.18. 14:40 - run #2)
<terminated, exit value: 0>/queuenet/lib/queuenet.exe (2007.07.18. 14:40 - run #1)
/queuenet/lib/queuenet.exe (2007.07.18. 14:40 - run #3)

The Debug View showing three runs in a simulation batch

The Debug View is another one of Eclipse's standard Views, but it is not only useful for debugging. While the Progress View only shows currently executing processes, the Debug View displays the ones already terminated, as well, together with their exit codes. Processes are marked with run numbers and launch times for easier identification. Double-clicking an item reveals the process output in the Console View.

Sequence Chart

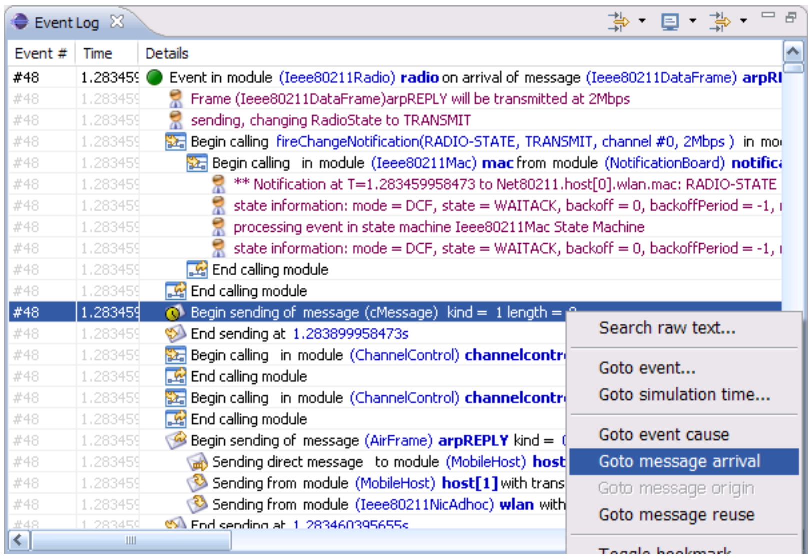
The OMNeT++ simulation kernel is capable of logging various events during simulation: scheduling and canceling self-messages, sending messages, display changes, module and connection creation and deletion, user log messages, etc. The result is an event log file which contains detailed information of the simulation, and later can be used for various purposes, such as visualizing message exchanges among modules on a sequence chart. The file format is line-based text to facilitate processing by third party tools.


omnetpp-big.log
1s 200ms
+83ms 200us
+83ms 450us
+83ms 800us
+83ms 950us
+84ms 162us
Range: 54ms 400us
Net80211.host[0].wlan.mac
Net80211.host[0].wlan.radio
Net80211.host[1].wlan.radio
Net80211.host[1].wlan.mac
IDLE
WAITACK
RECEIVING
TRANSMITTING
DEFER
DIFS
arpREQ -> arpREPLY
arpREQ
arpREPLY
endRx
wlan-ack
SIFS
timeout
arpREPLY -> ping
+0s
Event #64 at t = 1.28415
Simulation time difference
- received message from
- Self address: 0A-AA-00-
- state information: mode
- processing event in stat
- FSM Ieee80211Mac Stat
- firing Receive-ACK trans
- cancelling timeout perio
- dropping frame from tra
- requesting another fram
- FSM Ieee80211Mac Stat
- state information: mode

A Sequence Chart showing ARP on a wireless network

An event log can be visualized with the Sequence Chart in the IDE. On the Sequence Chart, events are represented as nodes, and message send/schedule/reuse operations as arrows. The chart can be filtered according to modules, message types, and various other criteria. The chart can also be filtered according to the causes/consequences of a particular event (this works by following message arrows backward and forward in simulation time). The chart timeline can be displayed in various modes, such as linear (simulation time), compact (non-linear), and event number-based. Other features include sorting of module axes, displaying state vector data on axes (using output vector files), zooming, and special navigation options (bookmarks, zoom to message send, follow message send), and exporting the chart in SVG format.

Both the Event Log Table and the Sequence Chart are capable of efficiently displaying event log files of several gigabytes without requiring large amounts of memory.



Event #	Time	Details
#48	1.283459	Event in module (Ieee80211Radio) **radio** on arrival of message (Ieee80211DataFrame) **arpR**
#48	1.283459	Frame (Ieee80211DataFrame)arpREPLY will be transmitted at 2Mbps
#48	1.283459	sending, changing RadioState to TRANSMIT
#48	1.283459	Begin calling fireChangeNotification(RADIO-STATE, TRANSMIT, channel #0, 2Mbps) in mo
#48	1.283459	Begin calling in module (Ieee80211Mac) **mac** from module (NotificationBoard) **notifica**
#48	1.283459	** Notification at T=1.283459958473 to Net80211.host[0].wlan.mac: RADIO-STATE
#48	1.283459	state information: mode = DCF, state = WAITACK, backoff = 0, backoffPeriod = -1,
#48	1.283459	processing event in state machine Ieee80211Mac State Machine
#48	1.283459	state information: mode = DCF, state = WAITACK, backoff = 0, backoffPeriod = -1,
#48	1.283459	End calling module
#48	1.283459	End calling module
#48	1.283459	Begin sending of message (cMessage) kind = 1 length =
#48	1.283459	End sending at 1.283899958473s
#48	1.283459	Begin calling in module (ChannelControl) **channelcontr**
#48	1.283459	End calling module
#48	1.283459	Begin calling in module (ChannelControl) **channelcontr**
#48	1.283459	End calling module
#48	1.283459	Begin sending of message (AirFrame) **arpREPLY** kind =
#48	1.283459	Sending direct message to module (MobileHost) **host**
#48	1.283459	Sending from module (MobileHost) **host[1]** with trans
#48	1.283459	Sending from module (Ieee80211NicAdhoc) **wlan** with
#48	1.283459	End sending at 1.283460395655s

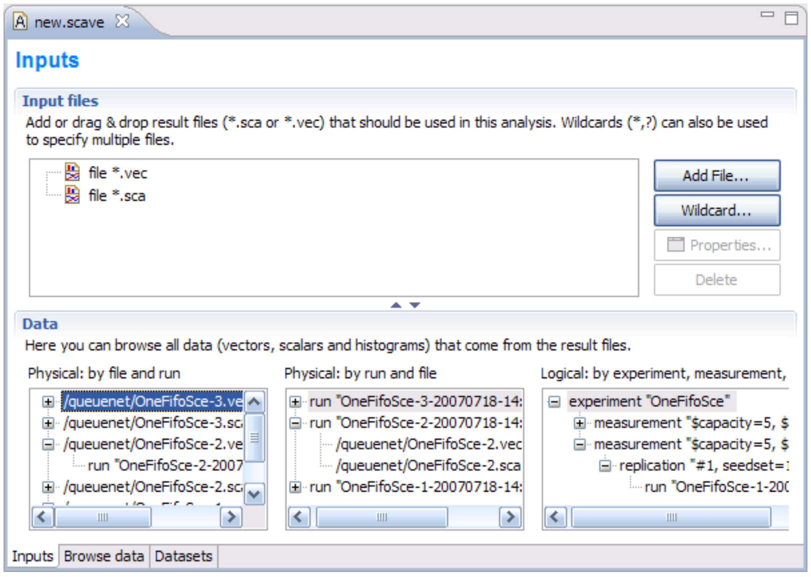
The Event Log View

The Event Log Table displays the content of an event log file recorded during a simulation. It supports raw and descriptive notations, line-based filtering for event log entry types and various parameters, search for free text, bookmarks, and special navigation options such as "go to simulation time" or message arrival. The Event Log Table supports the same filtering as the Sequence Chart.

Scave (Result Analysis)

Scave is the result analysis tool of OMNeT++ and its task is to help the user process and visualize simulation results saved into vector and scalar files. Scave is designed so that the user can work equally well on the output of a single simulation run (one or two files) and the result of simulation batches (which may be several hundred files, possibly in multiple directories). Ad-hoc browsing of the data is supported in addition to systematic and repeatable processing. With the latter, all processing and charts are stored as “recipes”. For example, if simulations need to be re-run due to a model bug or misconfiguration, existing charts do not need to be drawn all over again. Simply replacing the old result files with the new ones will result in the charts being automatically displayed with the new data.

Scave is implemented as a multi-page editor. What the editor edits is the “recipe,” which includes what files to take as inputs, what data to select from them, what (optional) processing to apply, and what kind of charts to create from them. The pages (tabs) of the editor roughly correspond to these steps. You will see that Scave is much more than just a union of the OMNeT++ 3.x Scalars and Plove tools.



Inputs

Input files

Add or drag & drop result files (*.sca or *.vec) that should be used in this analysis. Wildcards (*,?) can also be used to specify multiple files.

file *.vec
file *.sca

Data

Here you can browse all data (vectors, scalars and histograms) that come from the result files.

Physical: by file and run
/queuenet/OneFifoSce-3.ve
/queuenet/OneFifoSce-3.sc
/queuenet/OneFifoSce-2.ve
run "OneFifoSce-2-2007
/queuenet/OneFifoSce-2.sc

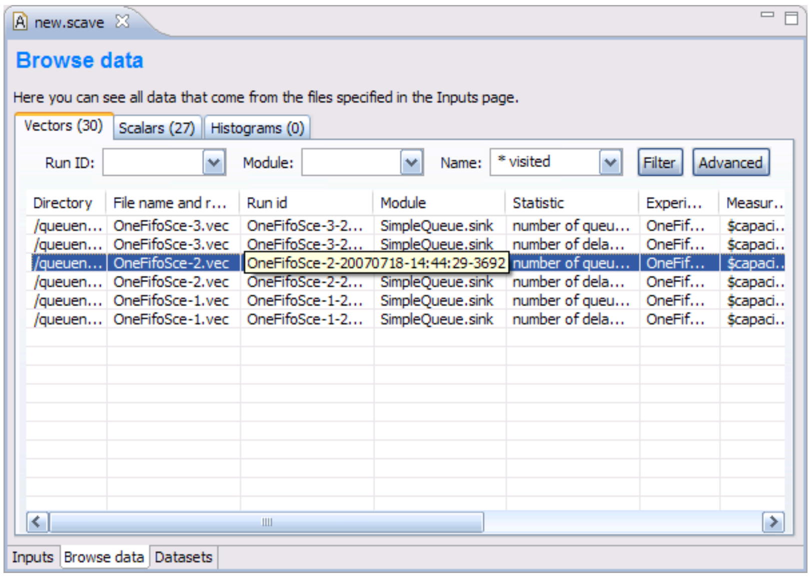
Physical: by run and file
run "OneFifoSce-3-20070718-14:
run "OneFifoSce-2-20070718-14:
/queuenet/OneFifoSce-2.vec
/queuenet/OneFifoSce-2.sca
run "OneFifoSce-1-20070718-14:

Logical: by experiment, measurement,
experiment "OneFifoSce"
measurement "$capacity=5, $
measurement "$capacity=5, $
replication "#1, seedset=
run "OneFifoSce-1-20

Specifying input files for data analysis

The first page displays the result files that serve as input for the analysis. The upper half specifies what files to select, by explicit filenames or by wildcards. The lower half shows what files actually matched the input specification and what runs they contain. Note that OMNeT++ result files contain a unique run ID and several metadata annotations in addition to the actual recorded data. The third tree organizes simulation runs according to their experiment—measurement—replication labels.

The underlying assumption is that users will organize their simulation-based research into various “experiments”. An experiment will consist of several “measurements”, which are typically (but not necessarily) simulations done with the same model but with different parameter settings; that is, the user will explore the parameter space with several simulation runs. To gain statistical confidence in the results, each measurement will be possibly repeated several times with different random number seeds. It is easy to set up such scenarios with the improved ini files of OMNeT++. Then, the experiment-measurement-replication labels will be assigned more-or-less automatically – please refer to the Inifile document ("Configuring Simulations in OMNeT++") for more discussion.



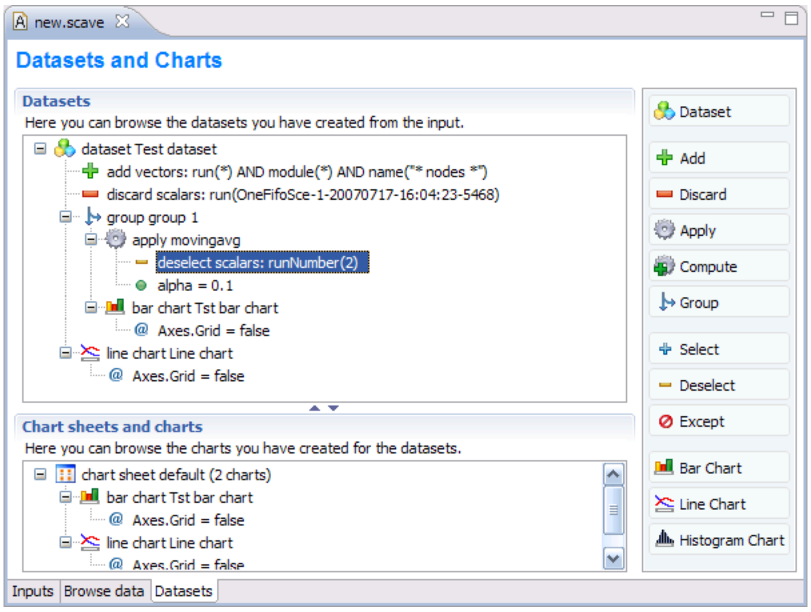
Here you can see all data that come from the files specified in the Inputs page.

Run ID: | Module: | Name: * visited

Directory	File name and r...	Run id	Module	Statistic	Experi...	Measur..
/queuen...	OneFifoSce-3.vec	OneFifoSce-3-2...	SimpleQueue.sink	number of queu...	OneFif...	$capaci..
/queuen...	OneFifoSce-3.vec	OneFifoSce-3-2...	SimpleQueue.sink	number of dela...	OneFif...	$capaci..
/queuen...	OneFifoSce-2.vec	OneFifoSce-2-20070718-14:44:29-3692		number of queu...	OneFif...	$capaci..
/queuen...	OneFifoSce-2.vec	OneFifoSce-2-2...	SimpleQueue.sink	number of dela...	OneFif...	$capaci..
/queuen...	OneFifoSce-1.vec	OneFifoSce-1-2...	SimpleQueue.sink	number of queu...	OneFif...	$capaci..
/queuen...	OneFifoSce-1.vec	OneFifoSce-1-2...	SimpleQueue.sink	number of dela...	OneFif...	$capaci..

Browsing vector and scalar data generated by the simulation

The second page displays results (vectors, scalars, and histograms) from all files in tables and lets the user browse them. Results can be sorted and filtered. Simple filtering is possible with combo boxes, or when that is not enough, the user can write arbitrarily complex filters using a generic pattern matching expression language. Selected or filtered data can be immediately plotted, or remembered in named datasets *for further processing*.



Datasets

Here you can browse the datasets you have created from the input.

```
dataset Test dataset
    add vectors: run(*) AND module(*) AND name("* nodes *")
    discard scalars: run(OneFifoSce-1-20070717-16:04:23-5468)
    group group 1
        apply movingavg
            deselect scalars: runNumber(2)
            alpha = 0.1
        bar chart Tst bar chart
            Axes.Grid = false
    line chart Line chart
        Axes.Grid = false
```

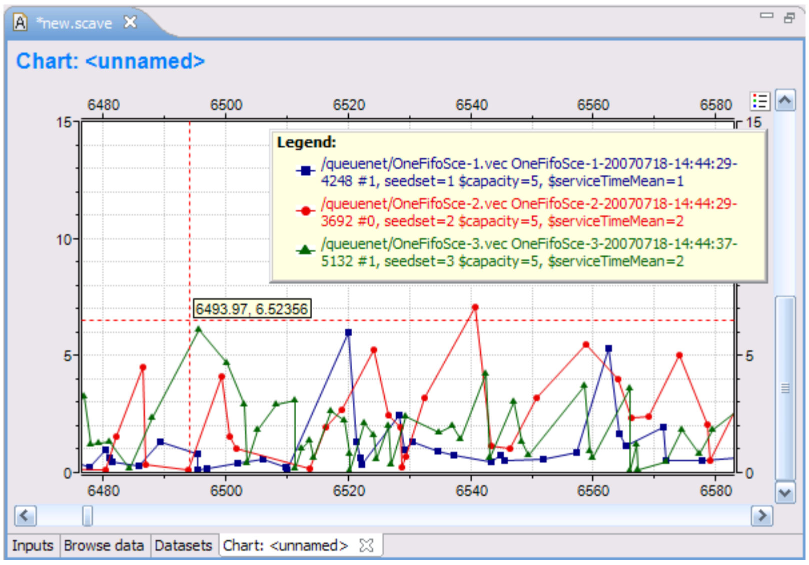
Chart sheets and charts

Here you can browse the charts you have created for the datasets.

```
chart sheet default (2 charts)
    bar chart Tst bar chart
        Axes.Grid = false
    line chart Line chart
        Axes.Grid = false
```

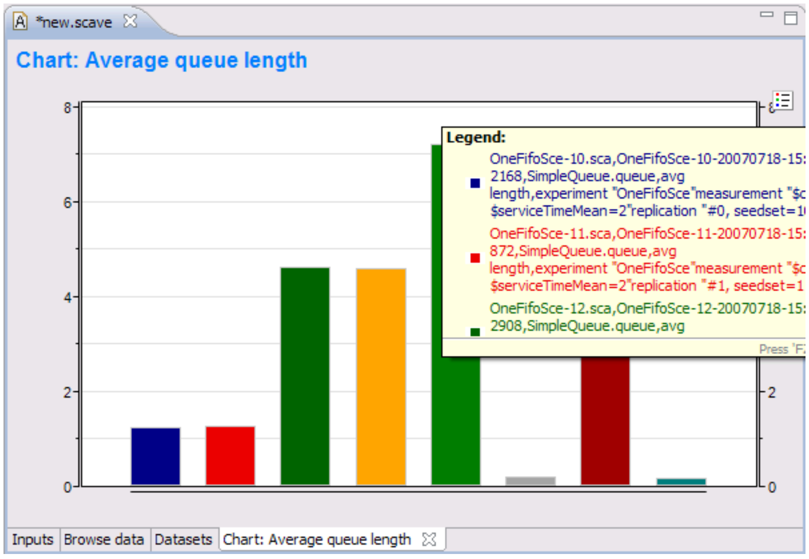
Defining datasets to be analyzed

It is possible to define reusable datasets that are basically recipes on how to select and process data received from the simulation. You can add selection and data processing nodes to a dataset. Chart drawing is possible at any point in the processing tree.


*new.scave
Chart: <unnamed>
Legend:
/queuenet/OneFifoSce-1.vec OneFifoSce-1-20070718-14:44:29-4248 #1, seedset=1 $capacity=5, $serviceTimeMean=1
/queuenet/OneFifoSce-2.vec OneFifoSce-2-20070718-14:44:29-3692 #0, seedset=2 $capacity=5, $serviceTimeMean=2
/queuenet/OneFifoSce-3.vec OneFifoSce-3-20070718-14:44:37-5132 #1, seedset=3 $capacity=5, $serviceTimeMean=2
6493.97, 6.52366
Inputs
Browse data
Datasets
Chart: <unnamed>

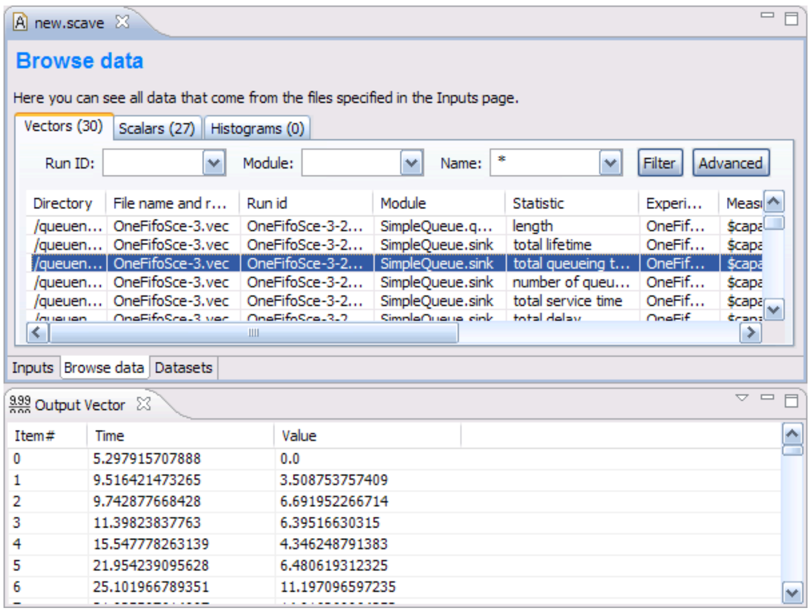
A Line Chart

Line charts are typically drawn from time-series data stored in vector files. Pre-processing of the data is possible in the dataset. The line chart component can be configured freely to display the vector data according to your needs.


*new.scave
Chart: Average queue length
Legend:
OneFifoSce-10.sca,OneFifoSce-10-20070718-15:2168,SimpleQueue.queue,avg length,experiment "OneFifoSce"measurement "$c $serviceTimeMean=2"replication "#0, seedset=1
OneFifoSce-11.sca,OneFifoSce-11-20070718-15:872,SimpleQueue.queue,avg length,experiment "OneFifoSce"measurement "$c $serviceTimeMean=2"replication "#1, seedset=1
OneFifoSce-12.sca,OneFifoSce-12-20070718-15:2908,SimpleQueue.queue,avg
Press 'F
Inputs
Browse data
Datasets
Chart: Average queue length

A Bar Chart

Bar charts are created from scalar results and histograms. Relevant data can be grouped and displayed via the Bar chart component. Colors, chart type, and other display attributes can be set on the component.



Browse data

Here you can see all data that come from the files specified in the Inputs page.

Vectors (30) | Scalars (27) | Histograms (0)

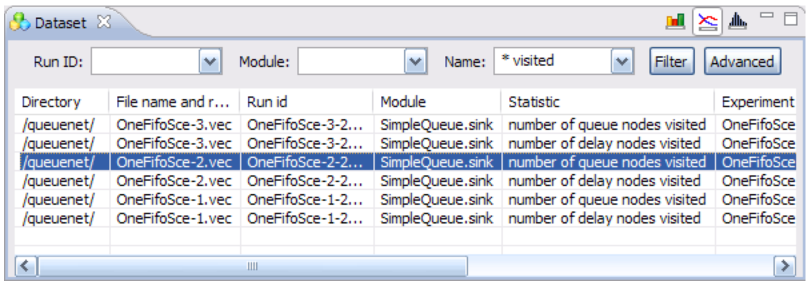
Run ID: Module: Name: *

Directory	File name and r...	Run id	Module	Statistic	Experi...	Meas
/queuen...	OneFifoSce-3.vec	OneFifoSce-3-2...	SimpleQueue.q...	length	OneFif...	$capa
/queuen...	OneFifoSce-3.vec	OneFifoSce-3-2...	SimpleQueue.sink	total lifetime	OneFif...	$capa
/queuen...	OneFifoSce-3.vec	OneFifoSce-3-2...	SimpleQueue.sink	total queueing t...	OneFif...	$capa
/queuen...	OneFifoSce-3.vec	OneFifoSce-3-2...	SimpleQueue.sink	number of queu...	OneFif...	$capa
/queuen...	OneFifoSce-3.vec	OneFifoSce-3-2...	SimpleQueue.sink	total service time	OneFif...	$capa
/queuen	OneFifoSce-3.vec	OneFifoSce-3-2	SimpleQueue.sink	total delay	OneFif	$capa

Item#	Time	Value
0	5.297915707888	0.0
1	9.516421473265	3.508753757409
2	9.742877668428	6.691952266714
3	11.39823837763	6.39516630315
4	15.547778263139	4.346248791383
5	21.954239095628	6.480619312325
6	25.101966789351	11.197096597235

Output Vector View

The Output Vector View can be used to inspect the raw numerical data when required. It can show the original data read from the vector file, or the result of a computation. The user can select a point on the line chart or a vector in the Dataset View and its content will be displayed.



Run ID: Module: Name: * visited

Directory	File name and r...	Run id	Module	Statistic	Experiment
/queuenet/	OneFifoSce-3.vec	OneFifoSce-3-2...	SimpleQueue.sink	number of queue nodes visited	OneFifoSce
/queuenet/	OneFifoSce-3.vec	OneFifoSce-3-2...	SimpleQueue.sink	number of delay nodes visited	OneFifoSce
/queuenet/	OneFifoSce-2.vec	OneFifoSce-2-2...	SimpleQueue.sink	number of queue nodes visited	OneFifoSce
/queuenet/	OneFifoSce-2.vec	OneFifoSce-2-2...	SimpleQueue.sink	number of delay nodes visited	OneFifoSce
/queuenet/	OneFifoSce-1.vec	OneFifoSce-1-2...	SimpleQueue.sink	number of queue nodes visited	OneFifoSce
/queuenet/	OneFifoSce-1.vec	OneFifoSce-1-2...	SimpleQueue.sink	number of delay nodes visited	OneFifoSce

Dataset View

The Dataset View is used to show the result items contained in the dataset. The content of the view corresponds to the state of the dataset after the selected processing is performed.