Event Log in OMNeT++ 4.0

Levente Mészáros (levente.meszaros@gmail.com)

András Varga (andras@omnetpp.org)

July 9, 2007

1 Introduction

OMNeT++ 4.0 introduces the concept of event log as an extension of the log mechanism found in earlier versions. The simulation kernel is capable to log automatically various things during the simulation along with user level messages. These include among others module and connection creation and deletion, scheduling of self messages, sending of messages to other modules either through gates or directly and processing of messages. The result is an event log file which contains detailed information of the simulation and later can be used for various purposes.

Understanding a complex simulation model and making it correct might be a difficult task to do. One of the goals of the event log concept is to provide tools which help this process. The following chapters discuss how to set up the event log and what services do the tools provide to you.

2 Configuration

For details about the ini file format in general, see the document titled *Configuring Simulations in OMNeT++ 4.0*

2.1 Event Log File

Since writing an event log file might significantly decrease overall simulation performance therefore event log recording is turned off by default. To turn on event log recording enter a value for the following configuration key in the ini file:

```
eventlog-file
```

An example configuration might look like this:

```
eventlog-file = ieee-80211-omnetpp.log
```

The simulation kernel will write the event log file with the provided name into the working directory during the simulation. The event log file may be opened in Eclipse while the simulation is still running and wherever it makes sense tools will follow changes written to it.

Note: The event log file should have the file extension ‘log’ as the example shows above so that the Eclipse tools will be able to recognize it automatically.

2.2 Recording Message Data

Since recording message data might dramatically increase the size of the event log file and additionally slow down the simulation therefore it is turned off by default even if writing the event log is enabled. To turn on message data recording enter a value for the following configuration key in the ini file:

```
eventlog-message-detail-pattern
```

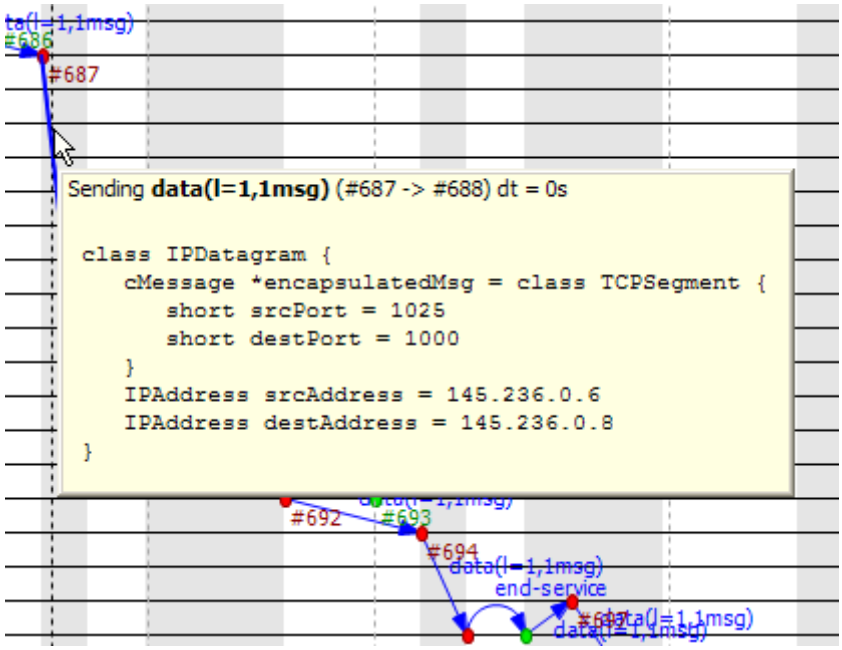
An example configuration for a IEEE 80211 model that records the field *encapsulationMsg* and all other fields which name ends with *Address* from messages which class name ends with *Frame* looks like this:

```
eventlog-message-detail-pattern = *Frame:encapsulatedMsg,*Address
```

An example configuration for a TCP/IP model that records the port and address fields looks like the following:

```
eventlog-message-detail-pattern = PPPFrame:encapsulatedMsg|
IPDatagram:encapsulatedMsg,*Address|TCPSegment:*Port
```

In general the configuration entry is a list of patterns separated by '|' character which will be used to write message detail information into the event log for each message sent during the simulation. The message detail will be automatically presented in the sequence chart and the event log table tool. Each pattern starts with an object pattern optionally followed by ':' character and a comma separated list of field name patterns. In the object pattern and/or/not/* and various field matcher expressions can be used. The field pattern contains wildcard expressions matched against field names.



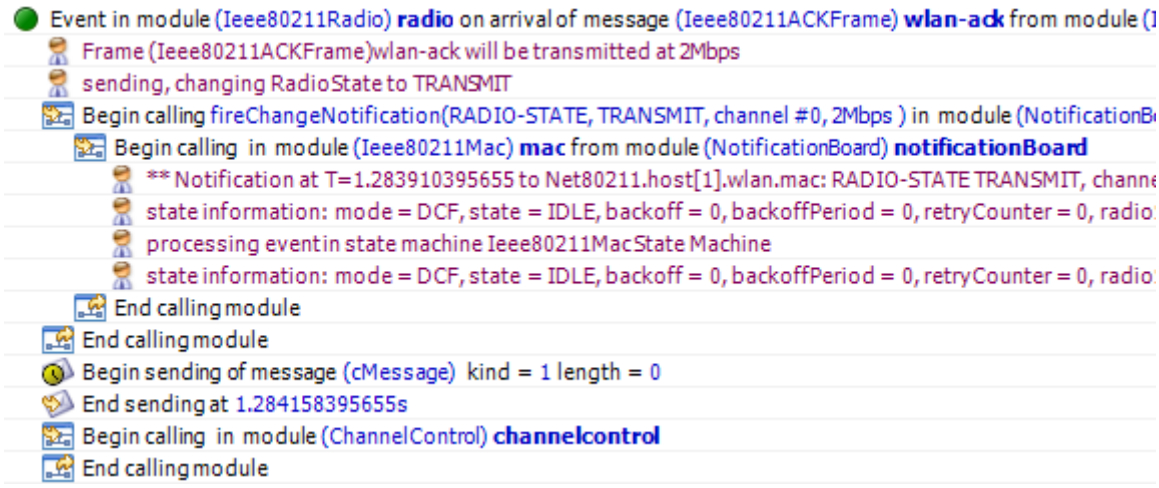
Message detail

Note: Take care about long running simulations because they might output event log files of several gigabytes and thus fail due to insufficient disk space. Both the event log table and the sequence chart are capable of efficiently displaying such huge event log files without requiring large amounts of physical memory to load the whole file at once.

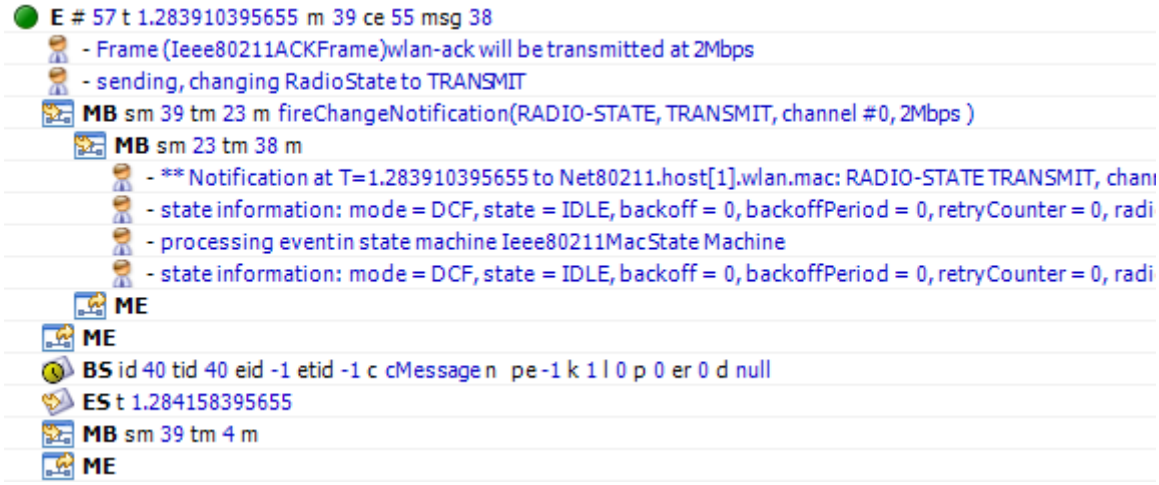
3 Event Log Table

The event log table is able to display the content of an event log file recorded during a simulation both as a view and as an editor in Eclipse. It is also capable to display filtered event log files whether they were created via the event log tool filter command or by custom filter tools or by on the fly in memory filtering.

The event log file content may be displayed in two different notations. The raw data notation displays exactly what was written to the file while the descriptive notation displays it after some preprocessing into a human readable form. Use the toolbar or the context menu to switch between them.



Descriptive notation



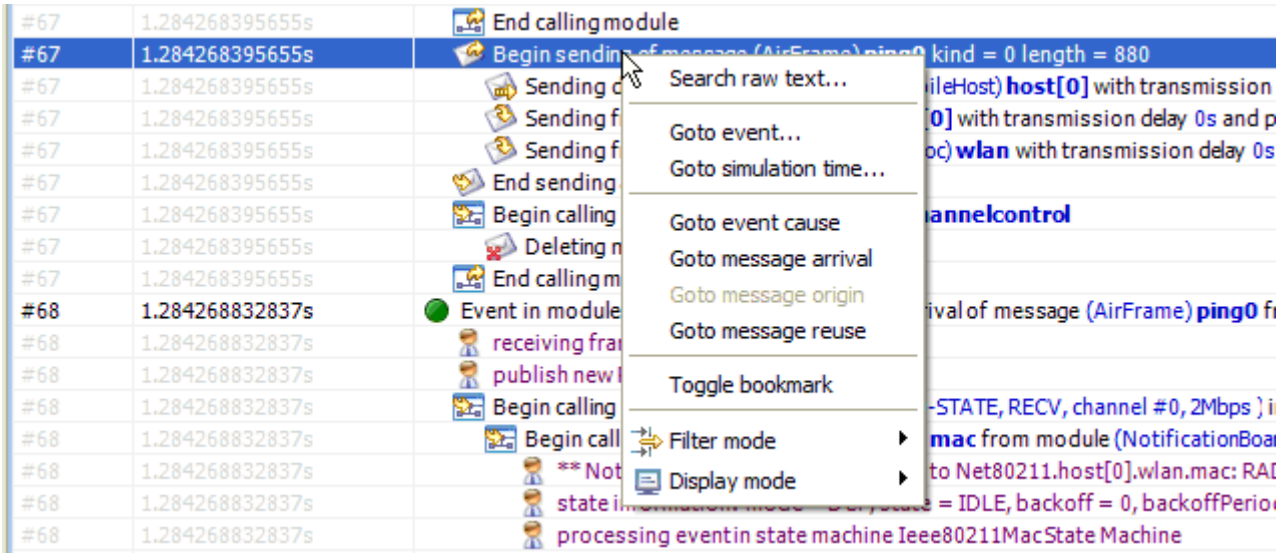
Raw notation

The event log table may be filtered by using the filter mode selector on the toolbar or from the context menu which allows filtering entries with specific types. There are some predefined filters but also custom filter patterns may be specified referring to fields present in the raw mode. The following is a custom filter example which shows message sends where the class of the message is AirFrame.

```
BS and c(AirFrame)
```

Note: Event lines marked with green circles are always shown in the event log table.

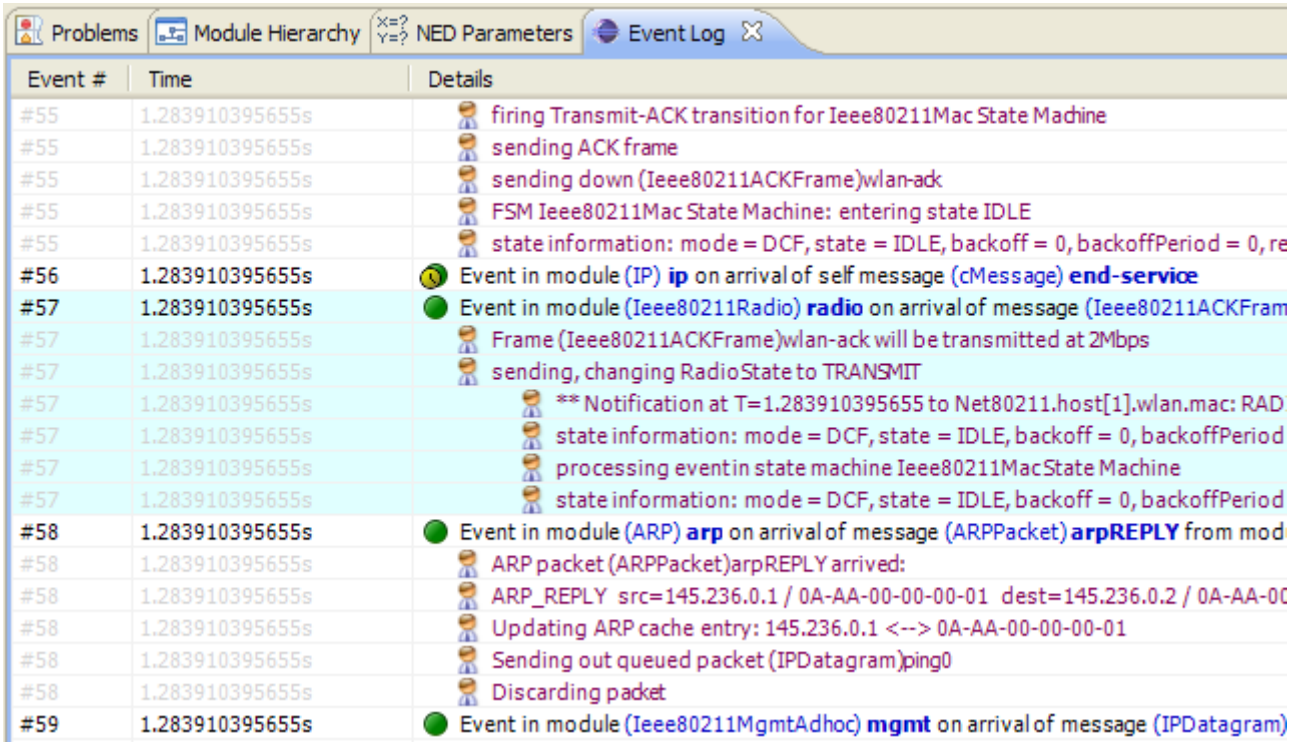
Standard navigation and selection within the event log table is done just like in any other table but there are extra navigation options in the context menu. These options focus on the causes and consequences of message sends.



Context menu

When viewing the very end (which means that the end key was actually pressed) of a running simulation's event log file then the event log table will follow the changes as new lines are written into the file.

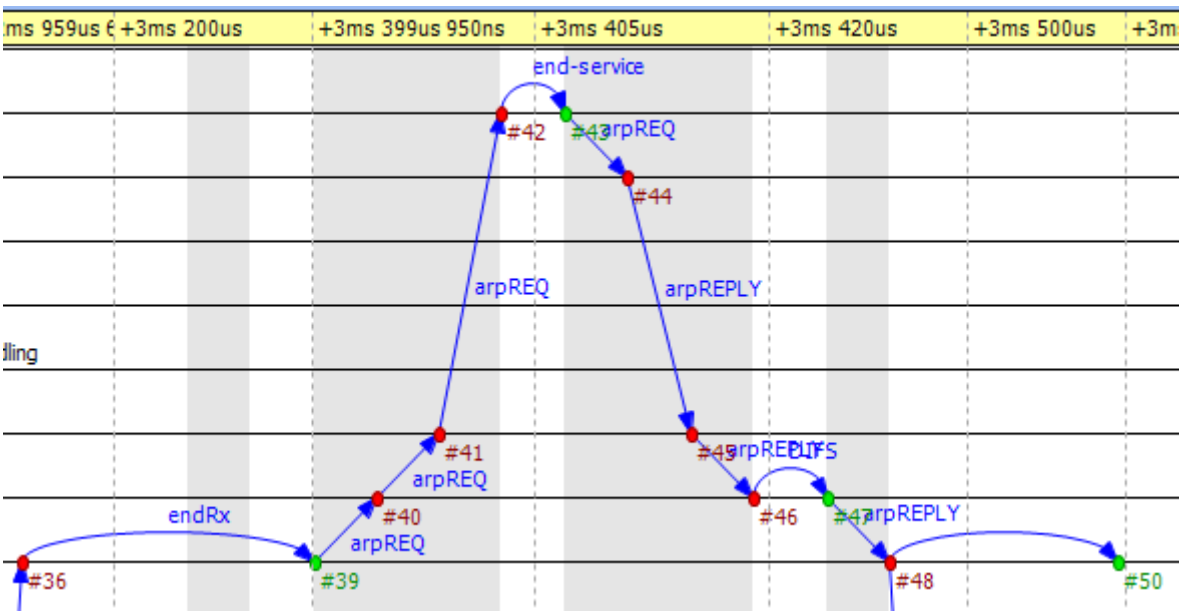
For easier navigation the Eclipse navigation history and bookmarks (from the context menu) are supported. Use the bookmark view to jump to bookmarks even after restarting eclipse.



Bookmark

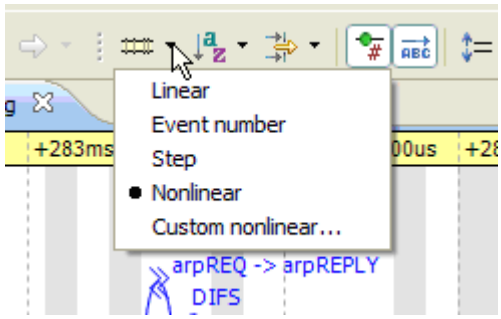
4 Sequence Chart

The sequence chart also displays the content of an event log file focusing on the causes and consequences of events and message sends recorded during the simulation. It is also capable to display filtered event log files just like the event log table.



Sequence chart

The horizontal axis of the sequence chart corresponds to the simulation time in four different ways. Linear and event number mean that the simulation time and event number is proportional to the distance measured in pixels respectively. While non linear and step mean that there is a non linear transformation between distance measured in pixels and simulation time. The goal of this non linear transformation is to get a compact figure even if there are longer and shorter periods of simulation time intervals between events. You can switch between timeline modes from the toolbar or the context menu.

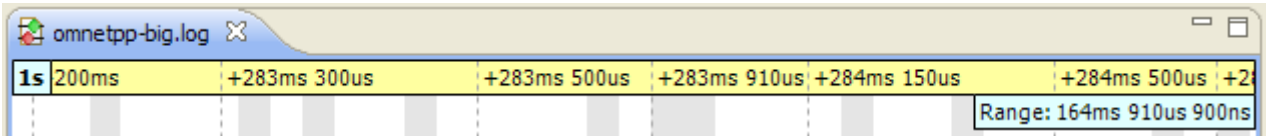


Timeline mode

It is quite common in simulation models that multiple events occur at the very same simulation time. This is shown as grey areas indicating that the simulation time does not change along the horizontal axis within the area.

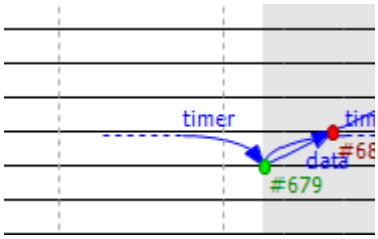
The vertical axis of the sequence chart corresponds to modules in the simulation. By default all modules are displayed as a single solid line and events occurred in that module are shown as filled circles. A green circle represents the processing of a self message while a red circle is an event caused by receiving a message from another module. Axes can be reordered manually, by module name, by module id or by (approximately) minimizing the number of crossing arrows between them.

Both the top and the bottom of the sequence chart display the simulation times corresponding to the currently visible part. The left side of the top simulation time gutter displays the common simulation time prefix which should be added to each individual simulation time shown at the vertical hairlines. The right side of the figure displays the simulation time range currently visible within the sequence chart window.



Gutter

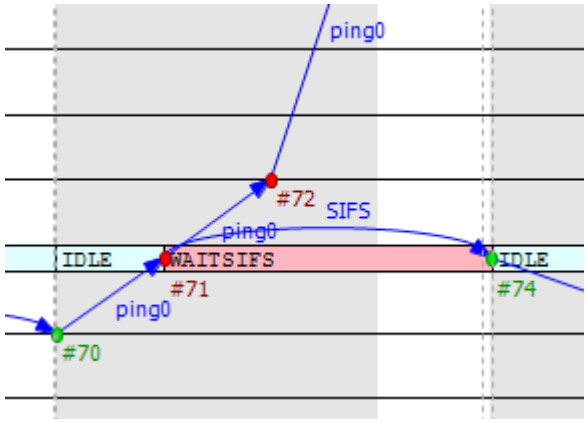
The sequence chart represents message sends via blue arrows. The arrow starts at the module which is sending the message and ends at the module which processed that message. The simulation times corresponding to the start and end points are also important. Sometimes a message send goes far away on the figure in which case the line is broken into two small parts displayed at the two ends pointing towards each other but without actually linked with a continuous line.



A message arriving from far away

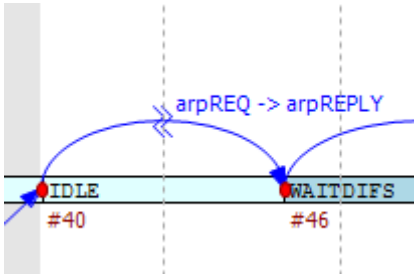
There is a green arrow representing message reuses. When a message arrives to a module which neither deletes nor sends but rather stores it and later on sends the very same message to another module then the events where the message arrived and where the message was actually sent are in message reuse relationship.

It is possible to attach Vector data to individual axes from the context menu by right clicking on the axis. The vector to be displayed must be of type enum and be registered with the C++ macro Register_Enum. In this case the solid line of the axis will be turned into a thick colored bar based on the enum values present in the vector.



Vector data with enum type displayed on module axis

Filtering the sequence chart is also possible. This actually means that some of the events are not displayed on the chart so that the user can focus on the relevant parts. When filtering is turned on (displayed in the status line) some of the message arrows might be decorated with a filter sign (a double zigzag crossing the arrow line's center). Such a message arrow means that there is a message going out from the source module which after some processing in some other filtered modules reaches the target module. The message names correspond to the first and the last message in the chain of filtered message sends.

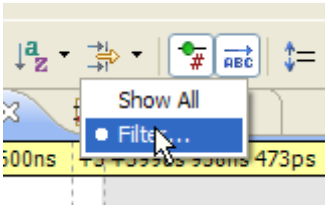


A chain of filtered message sends represented as a single arrow

The sequence chart supports documentation purposes by being able to export continuous parts into SVG format.

5 On The Fly Filter Tool

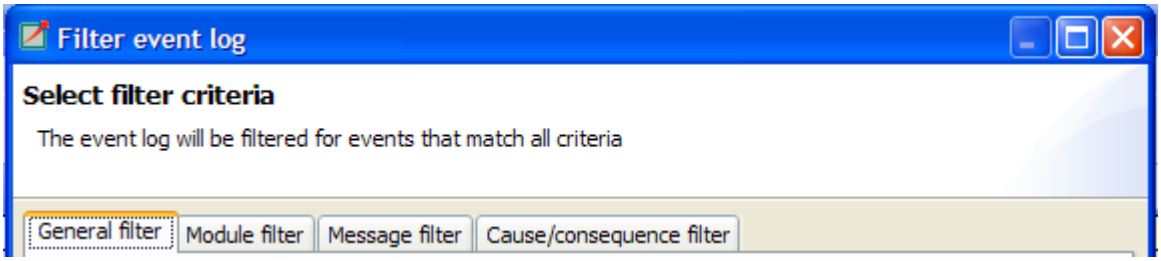
The content of the event log file may be filtered on the fly by opening the filter configuration.



Opening the filter configuration

On the fly filtering means that the result of the filter is not saved into a file but it is rather lazily computed and stored within memory. This allows rapid switching between various views on the same event log data within both the sequence chart and the event log table.

The filter configuration dialog has four tabs each restricting the content of the event log on its own. The individual tabs may be turned on and off independently of each other.



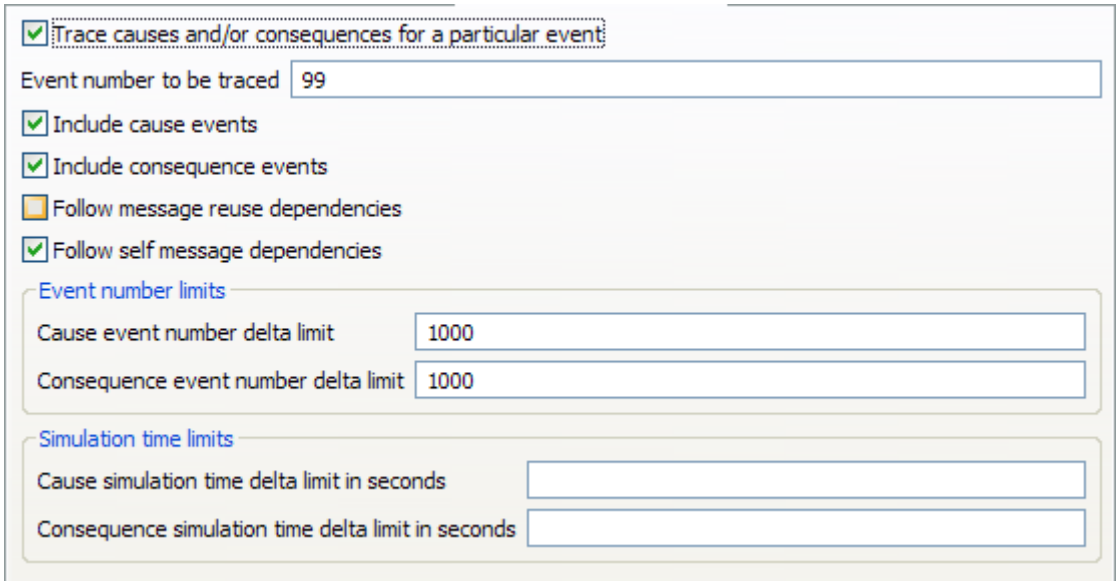
Filter configuration dialog

The general filter is the simplest one which filters out events from the beginning and from the end of the event log file. This might help to reduce the computation time when defining filters which otherwise would be very expensive to compute for the whole event log.

The module and message filter tabs each have three inner tabs. These inner tabs restrict the content to events matching any condition present in them. Both the module and message filters allow selecting modules and messages by expression, type, name and id.

The trace filter allows filtering for causes and consequence of particular event specified by its event number. The cause/consequence relation between two events means that there is message send path from the cause event to the consequence event.

Note: Since computing the causes and consequences far away in the log file from the traced event might be a time consuming task therefore there are extra limits around the traced event to be set.



Trace tab

6 Event Log Tool

The event log tool is a command line tool to process event log files. Invoking it without parameters will display usage information such as available commands and options. The following are the commands most useful for users.

6.1Filter

The event log tool provides off line filtering that is usually applied to the event log file after the simulation has been finished and before actually opening it in Eclipse. Use the *filter* command and the various options to specify what should be present in the result file.

Some of the filter options are also available using the Eclipse platform for on the fly in memory filtering but may result in long running computations. In such cases pre-filtering the result file might give better performance.

6.2Echo

Since the event log file format is text based and users are encouraged to implement their own filters therefore there needs to be a way to check whether an event log file is correct. The echo command provides a way to check this and help users creating custom filters. Anything not echoed back by the event log tool will not be taken into consideration by the other tools.

Note: custom filter tools should filter out whole events only otherwise the consequences are undefined

7 Event Log Format

The event log file format is a quite simple text and line based format. The event log doesn't use a binary format to let users write their own filters and still be able to use the provided tools.

Each line in the event log file describes an event log entry. The line starts with an entry type identifier which is followed by a list of name, value parameter pairs. The various event log entry types have different parameters some of which contain booleans, simulation times, identifiers, class names, user specified names, quoted strings, etc.

For the sake of easy human recognition the simulation kernel writes an empty line before each *E* line. Although this is not mandatory it makes reading the file easier for humans.

See the list of event log entry type parameters and the BNF grammar for event log file format in the Appendix.

8 Appendix

8.1Event Log Configuration

eventlog-file = <file-name>

<file-name> ::= <text> **DOT 'log'**

eventlog-message-detail-pattern = <pattern> (**PIPE** <pattern>)*

<pattern> ::= <message-object-matcher> [**COLON** <message-field-patterns>]

<message-field-patterns> ::= <message-field-pattern> (**COMMA** <message-field-pattern>)*

<message-object-matcher> ::= <object-matcher-expression>

<message-field-pattern> ::= <wildcard-pattern>

8.2Event Log Format

<file> ::= <line>*

<line> ::= <empty-line> | <user-log-message> | <event-log-entry>

<empty-line> ::= **CR LF**

<user-log-message> ::= **-** **SPACE** <text> **CR LF**

<event-log-entry> ::= <event-log-entry-type> **SPACE** <parameters> **CR LF**

<event-log-entry-type> ::= **BU** | **MB** | **ME** | **MC** | **MD** | **MR** | **CC** | **CD** | **CS** | **MS** | **E** | **CE** | **BS** | **ES** | **SD** | **SH** | **DM**

<parameters> ::= (<name> **SPACE** <value>)*

<name> ::= <text>

<value> ::= <integer> | <text> | <quoted-text>

8.3Event Log Entry Type Parameters

Name	Type	Description
BU		display a bubble message
id	integer	id of the module which printed the bubble message
txt	string	displayed message text
MB		beginning of a call to another module
sm	integer	id of the caller module
tm	integer	id of the module being called
m	string	C++ method name
ME		end of a call to another module
MC		creating a module
id	integer	id of the new module
c	string	C++ class name of the module

pid	integer	id of the parent module
n	string	full dotted hierarchic module name
MD		deleting a module
id	integer	id of the module being deleted
MR		reparenting a module
id	integer	id of the module being reparented
p	integer	id of the new parent module
CC		creating a connection
sm	integer	id of the source module identifying the connection
sg	integer	id of the gate at the source module identifying the connection
sn	string	full name of the gate at the source module
dm	integer	id of the destination module
dg	integer	id of the gate at the destination module
dn	string	full name of the gate at the destination module
CD		deleting a connection
sm	integer	id of the source module identifying the connection
sg	integer	id of the gate at the source module identifying the connection
CS		a connection display string change
sm	integer	id of the source module identifying the connection
sg	integer	id of the gate at the source module identifying the connection
d	string	the new display string
MS		a module display string change
id	integer	id of the module
d	string	the new display string
E		an event that is processing of a message
#	integer	unique event number
t	simulation time	simulation time when the event occurred
m	integer	id of the processing module
ce	integer	event number from which the message being processed was sent or -1 if the message was sent from initialize
msg	integer	life time unique id of the message being processed
CE		cancelling an event caused by self message
id	integer	id of the message being removed from the FES
pe	integer	event number from which the message being cancelled was sent or -1 if the message was sent from initialize
BS		beginning to send a message
id	integer	life time unique id of the message being sent
tid	integer	id of the message inherited by dup
eid	integer	id of the message inherited by encapsulation
etid	integer	id of the message inherited by both dup and encapsulation
c	string	C++ class name of the message
n	string	message name
pe	integer	event number from which the message being sent was processed or -1

		if the message has not yet been processed before
k	integer	message kind
l	integer	message length in bits
p	integer	message priority
er	boolean	true indicates the message has bit errors
d	string	detailed information of message content when recording message data is turned on
ES		prediction of the arrival of a message
t	simulation time	when the message will arrive to its destination module
SD		sending a message directly to a destination gate
sm	integer	id of the source module from which the message is being sent
dm	integer	id of the destination module to which the message is being sent
dg	integer	id of the gate at the destination module to which the message is being sent
pd	simulation time	propagation delay that is while the message is propagated through the connection
td	simulation time	transmission delay that is while the whole message is sent from the source gate
SH		sending a message through a connection identified by its source module and gate id
sm	integer	id of the source module from which the message is being sent
sg	integer	id of the gate at the source module from which the message is being sent
pd	simulation time	propagation delay that is while the message is propagated through the connection
td	simulation time	transmission delay that is while the whole message is sent from the source gate
DM		deleting a message
id	integer	id of the message being deleted
pe	integer	event number from which the message being deleted was sent or -1 if the message was sent from initialize