

Distribution Panel

**Erik Klinskog
Anna Neiderud**

**Version 1.2.3
December 1, 2001**



Abstract

The Distribution Panel is a tool for monitoring the behavior of the distribution part of a Mozart engine. With its help it is possible to obtain an understanding of what is happening under the surface of your Mozart engine. It is useful to tune applications and detect unexpected behavior. To fully understand the information that the Distribution panel displays, good comprehension of the Mozart distribution model is required. See Seif and Peter's upcoming book.

Credits

Mozart logo by Christian Lindig

License Agreement

This software and its documentation are copyrighted by the German Research Center for Artificial Intelligence (DFKI), the Swedish Institute of Computer Science (SICS), and other parties. The following terms apply to all files associated with the software unless explicitly disclaimed in individual files.

The authors hereby grant permission to use, copy, modify, distribute, and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions. No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated on the first page of each file where they apply.

IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY DERIVATIVES THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. THIS SOFTWARE AND ITS DOCUMENTATION ARE PROVIDED ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

Contents

1 Basic Usage	1
1.1 Communication	1
1.1.1 Site list	2
1.1.2 Graphs	2
1.2 Exported Entities	3
1.3 Imported Entities	3
1.4 Message Statistics	3
1.5 Diff type Statistics	4
2 Monitoring remote Mozart engines	5
3 Interpretations of the Distribution Panel output	7
3.1 RTT fluctuations	7
3.2 Connection Port	7
3.3 Unexpected Exports	8
3.4 Mass importation of Variables	8

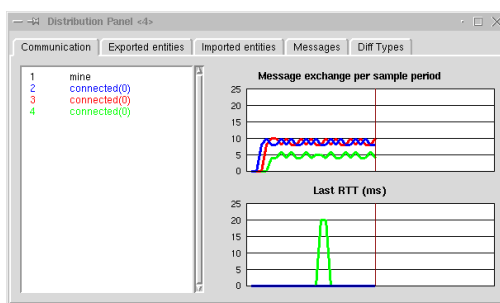
Basic Usage

The Distribution Panel is started in the OPI by feeding the line `{DistributionPanel.open}`. This opens a window that displays different data on the distributed work done by the current emulator.

Keep in mind that producing and displaying the data is a memory consuming task. Do not use the Distribution Panel with too much load, or the behaviour of the entire emulator will be highly affected and the results will not be the same as without the Distribution Panel.

The Distribution panel polls information at regular intervals. Some data is accumulative, e.g. Message exchange. For this data the number of events since the last poll is displayed. This is interpreted as events per sample period. Other data is transient, e.g. Last RTT (roundtrip time). Here the last known value is plotted.

Since the distribution behavior is affected by the garbage collector a red vertical bar is inserted in all graphs when garbage collection occurs. This allows garbage collection related behaviour to be understood.



The Distribution Panel has several different frames described in detail the remaining sections of this chapter.

1.1 Communication

The Communication frame displays the status and history of all Mozart sites that our site currently has references too. The information is composed of a list of all sites to the left and to graphs displaying the history of communication to and from each site.

1.1.1 Site list

In the list to the left, each site is numbered and given a separate color. By left clicking the site a detailed description of the site is shown in the browser. This contains the ip address, accepting port and the process id plus some more information. The output is taken from DPStatistics as defined in Chapter *Retriving statistical information from the Distribution layer: DPStatistics, (System Modules)*. An extra field, traffic shows the accumulated messagecounts.

The current state of the site is also printed in the site list. There is allways one site present with 'state' mine. That is a placeholder for the site representing the current emulator. The state of the other sites can be the following:

connected(N) A connection is established and the last measured round trip time to that site was N.

perm The site is permanently down.

passive or closed A reference is present but the emulator is currently not connected to that site since there is no need for communication.

problem There is no connection and there is some problem. It could be that it isn't possible to open a connection or that the connection was unexpectedly closed.

wait for handover, wait for remote, presentation or negotiate A connection is requested or is being initialized. Wait for handover means the local engine has been ordered to connect. Wait for remote means the remote site is responsible of reconnecting. Presentation and negotiate is part of an initialization phase.

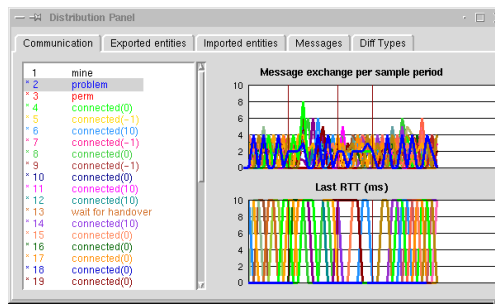
closing (hard), closing (weak) or wait for disconnect The connection is being closed. A hard close is due to lack of resources and will be reopened as soon as possible. A weak close is due to garbage collection when there is no need for the connection. Wait for disconnect is a passive wait for the remote party to close the channel.

1.1.2 Graphs

In the two graphs, the colors of the site list are used to show the communication between this site and each remote one.

Message exchange per sample period shows how many messages have been sent or received since the last poll for data.

Last RTT (ms) shows the last measured Round Trip Time at each poll for data. The RTT is defined as the time between when a message is transferred and when its acknowledgement is received back. This value includes the network RTT as well as any delays due to ongoing local or remote computation.

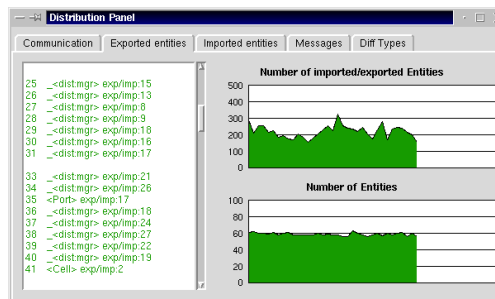


1.2 Exported Entities

The Exported Entities frame shows all entities that the current site has currently exported to the other sites. Notice that entities exported by the Connection module do not show up as exported until they are taken by another site.

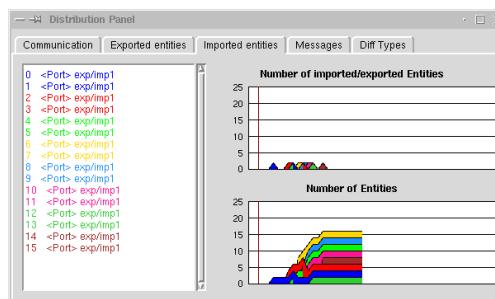
Number of imported/exported Entities shows how many times a reference to any exported entity has been sent or received by this site.

Number of Entities shows how many entities are currently exported.



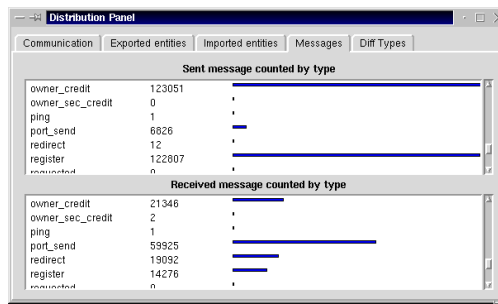
1.3 Imported Entities

The Imported Entities frame displays all entities currently imported to this site. The entities are colored with their origin site's color (see 1.1).



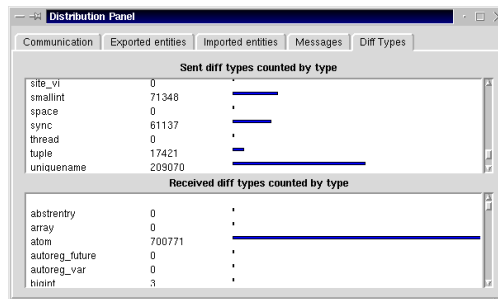
1.4 Message Statistics

The Message Statistics frame shows the total number of sent and received messages per message type. The Message type is the internal type used by the Message Passing Layer and the entity protocols.



1.5 Diff type Statistics

The Diff type Statistics shows the total number of imported and exported Mozart data structures.



Monitoring remote Mozart engines

In Chapter 1 we have seen how the Distribution Panel can be used to monitor communication to and from its own site. The Distribution Panel can also be used to monitor communication between remote Mozart engines. The communication frame can be opened locally for especially connected remote engines. This is a feature useful in distributed debugging, but must be used with care. Connecting creates extra communication that must be taken into account when viewing the statistics and may alter the behaviour of the application.

To connect Distribution Panels do the following:

```
{DistributionPanel.server ?P}
```

This call prepares the DistributionPanel to act as a server able to monitor connected engines. A port is returned and is to be used for such connections.

```
{DistributionPanel.client P}
```

If *P* is a port created by `{DistributionPanel.server?P}` this will connect the calling ozengine to the ozengine who created *P*.

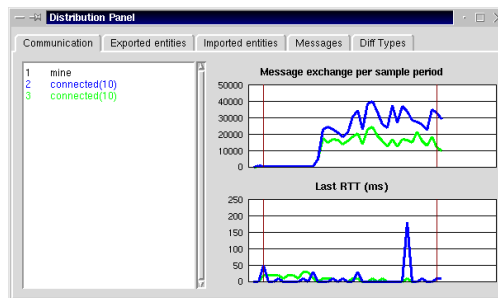
In the Distribution Panel of the server, an asterix (*) will appear next to the site of connected clients. By right clicking such a site, the communication frame of that client will be opened in a separate window at the server and the client will be told to start sending information on its communication. When the new window is closed the reports will be stopped.

Interpretations of the Distribution Panel output

Here we give a couple of examples of the output of the Distribution Panel. We describe a program with a special property and then we show how it is possible to see that property in the Distribution Panel.

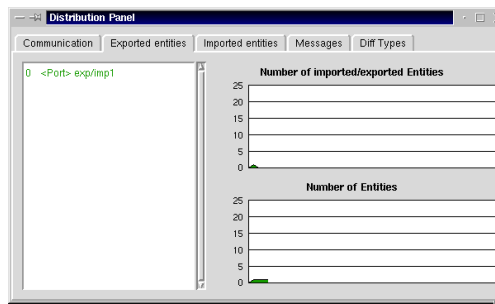
3.1 RTT fluctuations

In a client server application where the clients and the servers are involved in heavy communication even the smallest RTT fluctuations can totally ruin the performance. We monitor the server and observe strange transients in the RTT graph. The transients seem to come with regular intervals. An intelligent guess is that garbage collection at the clients and the server causes inaccessability and thus raises the RTT.



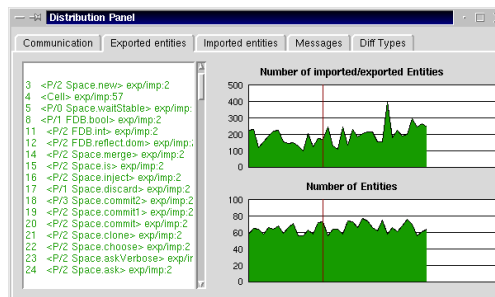
3.2 Connection Port

Inspecting the Exported entities frame on a disconnected Mozart emulator shows a port in the scroll window. This is the port that the connection mechanism uses and not a bug. It is allways present when some distribution has been initialized since this always initializes the Connection module.



3.3 Unexpected Exports

Unintentional exportation of Modules is a common bug in Mozart applications. With help from the Exported frame one can see all the entities that are exported.



3.4 Mass importation of Variables

When doing synchronone distributed calls using Variables for passing the answer a lot of temporary variables are created. Binding all the extra variables creates a lot of protocoll action and implies a lot of messages.

