Java MPI Simulator 2001

Rok Preskar

EPCC
University of Edinburgh
email: rok.preskar@kiss.uni-lj.si.

Introduction

The **Message Passing Interface (MPI)** [1] is a standard for communication between processes. The Java MPI simulator was designed as an applet to help people learning MPI understand communications between processes and learn different MPI commands by visualising the message passing process.

History

1. This applet was first developed in 1997 as a simulator for blocking point-to-point communications, visualised as envelopes traveling from process to process.
2. In 1998 it was extended by an SSP project to include collective communications. At this stage the user interface was completely revised.
3. Another SSP project in 2000 dealt mostly with documentation, refactoring, and making the applet easier to understand.

Old version of the applet

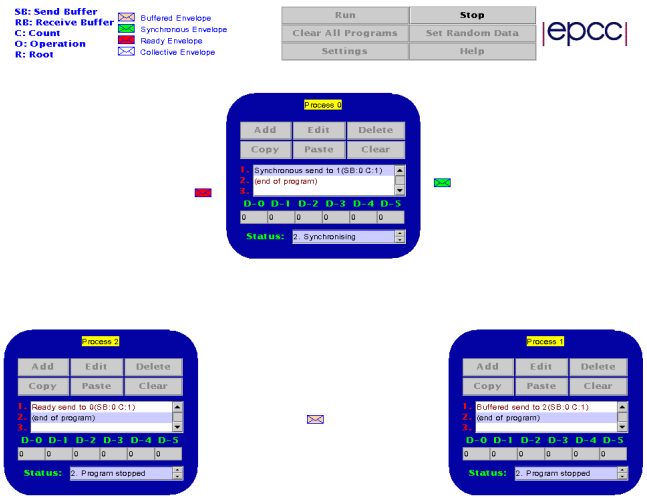
The applet consisted of several modules (coded as Java Beans), which were not completely independent. These were:

- **Miscutils** - Part of a different package, the utilities were scarcely used.
- **Envelope box** - Canvas capable of drawing the travelling envelopes.
- **Process** - Component representing one process involved in communications.
- **Controller** - Panel of buttons providing general applet commands.
- **Communicator** - Hidden component that supervised the collective communications.
- **MPI** - The runnable applet containing the other components listed above.

There were several problems, as well as gaps in functionality within the existing applet. This prompted the current project, which changed some of the existing functionality and extended it.

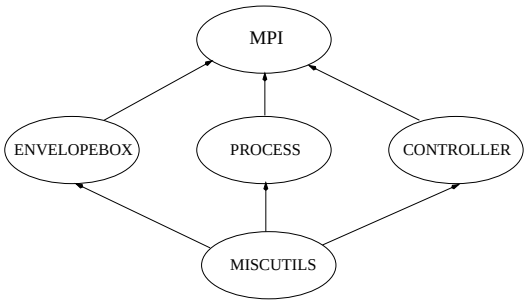
New version of the applet

On the figure below three processes exchanging messages via envelopes can be seen.

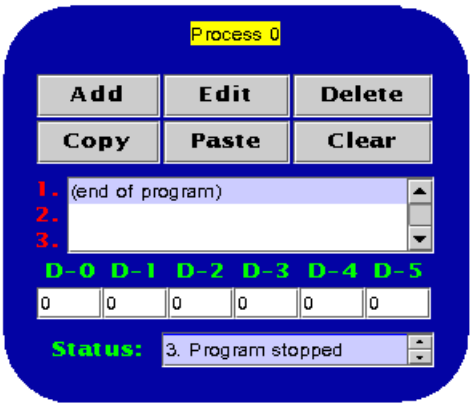


The modifications of the applet included:

- Removing inconsistencies between collective and point-to-point communications. Both have been changed to include changing the data buffers as well as displaying travelling envelopes.



- Removing most of the dependencies (some of them circular) among the different Beans in the applet. The communicator was merged into the controller and process modules for better consistency and understandability of the code. The remaining dependencies are shown on the figure above.
- Implementing Java Swing instead of AWT to provide consistent cross-platform appearance.
- Adding the ability for saving and restoring programmes through the applet [2].
- Allowing the user to set the number of processes within the applet.
- Making the applet more intuitive for the user.
- Adding help windows.



- Altering the appearance of each process to remove scaling problems. The appearance of a process is shown in the figure above.
- Adding numbers beside the programme list to indicate which commands the user is seeing.
- Improving error messages.

Acknowledgements

I would like to thank my supervisors Mario Antonioletti, Lindsay Pottage and Neil Chue Hong for providing help and guidance throughout the course of the project. I would also like to thank Roger Hare, one of the organisers of the Summer Scholarship Programme at EPCC, for providing me with valuable end-user comments.

Bibliography

1. Marc Snir, Steve W. Otto, Steven Huss-Lederman, David W. Walker and Jack Dongarra. MPI The Complete reference. The MIT Press. 1996.
2. Mary Dageforde, Security in Java 2 SDK 1.2, http://java.sun.com/docs/books/tutorial/security1.2/index.html.