



MASS/MEME Simulation Server Setup Aid

Prepared by
Rajmund Bocsi, Gábor Szemes, Zsolt Kúti



July 2010, Budapest

Contents

Introduction	3
Agent-Based Modeling	3
MASS	3
MASS/MEME Simulation Server	3
About this document	4
1 Components of a system	5
1.1 Preliminary	5
1.2 Prerequisites on all hosts	5
1.2.1 Operation System (OS)	5
1.2.2 SSH	5
1.2.2.1 Key Exchange (Windows only)	5
1.2.3 Workspace	7
1.2.3.1 Default SFTP Directory (Windows only)	7
1.2.4 Install Java	8
1.2.5 ProActive 4.1.2 Library	8
1.3 Prerequisites on workers	8
1.3.1 MEME libraries	8
1.3.2 RepastJ 3.1 Library	8
1.3.3 Other libraries	8
1.3.4 Configuration files	8
1.3.5 Plugins folder	8
1.3.6 PlatformPlugins folder	9
1.4 Prerequisites on master	9
1.4.1 SSH (External Access)	9
1.4.2 MASS/MEME Simulation Server Port	9
1.4.3 Related Files	9
1.4.4 Configuration Files	9
1.4.4.1 The proactive.java.policy file	9
1.4.4.2 The proactive-log4j file	9
1.4.4.3 The logging.properties file	10
1.4.4.4 The server_config.xml file	10
1.4.4.5 The ProActiveConfiguration.xml file	10
1.4.4.6 The ProActive Descriptor (master.descriptor.xml)	10
1.4.5 Launcher Scripts	19
1.4.5.1 Modifying the Launcher Scripts	19
2 Installing Simulation Platforms	20
2.1 The MASS/MEME PlatformReg Application	20
2.2 Installing Platform for Custom Java Models	20
2.3 Installing NetLogo 4.0.4 Platform	21
2.3.1 Example Descriptor	21
3 After system setup	25
4 Good to know	26
5 References	27

Introduction

Agent-Based Modeling

Agent-based modeling is a branch of computer simulation. It models the individual, together with its imperfections (e.g., limited cognitive or computational abilities), its idiosyncrasies and personal interactions. The approach builds the model from 'the bottom-up', focusing mostly on micro rules and seeking to understand the emergence of macro behavior. Participatory simulation - a branch of agent-based simulation - is a methodology building on the synergy of human actors and artificial agents, excelling in the training and decision-making support areas. In participatory simulations some agents are controlled by users, while others are software governed.

MASS

The Multi-Agent Simulation Suite (MASS) is a software package intended to enable modelers to utilize the tools of agent-based simulation in various fields, without having to develop heavy programming skills.

MASS consists of four applications built around a simulation core. The simulation suite has its own core called the Multi-Agent Core (MAC), but it is also able to run on the popular Repast core. Being multi-core enables modelers to verify that results are core-independent, thus we plan to further develop this option. The Functional Agent-Based Language for Simulation (FABLES) is a programming language and its integrated modeling environment specially designed for creating agent-based simulations. The Model Exploration Module (MEME) is a tool that enables orchestrating experiments, managing results and has support for their analysis. The Participatory Extension (PET) is an optional web-based environment for multi-agent and participatory simulations. The fourth element of MASS, the Visualization Package does not translate into a standalone application. It consists of the various implementations of charts and visualizations used in all the other software.

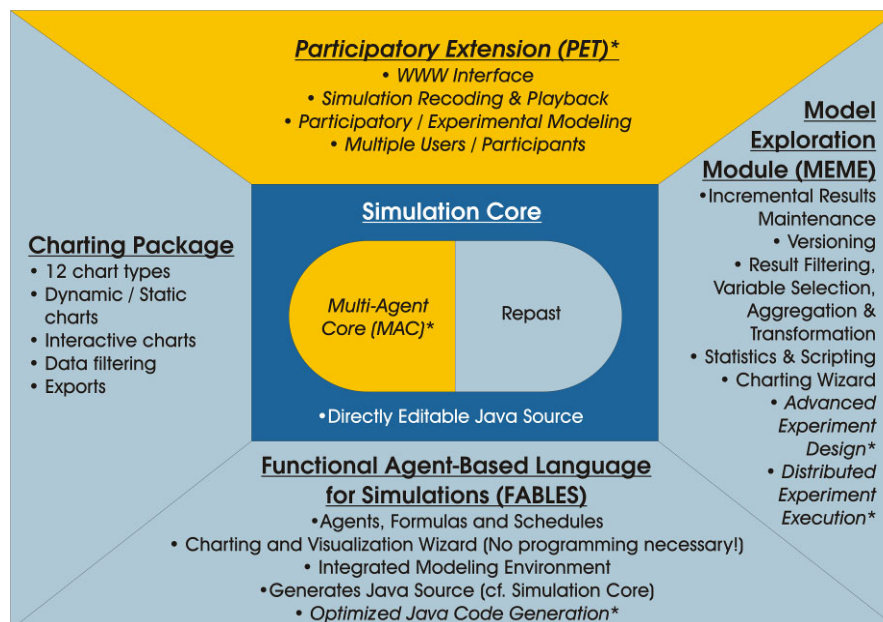


Figure 1 - Multi-Agent Simulation Suite

MASS/MEME Simulation Server

The MASS/MEME Simulation Server

- receives experiment execution request from MEME Parameter Sweep Wizard and schedules it;

- distributes the individual runs of the current experiment amongst its workers and collects the result;
- informs all connected monitors about the progress of the current experiment;
- allows to users to download the results of finished experiments.

About this document

This aid describes how to prepare your computer cluster before you start distributed parameter sweep with MEME Parameter Sweep Wizard. To learn more about the installation and usage of the MEME client, please consult its manual first ([1]).

Please note that this version of MASS/MEME Simulation Server (version 2.1.00714) can work with MEME v2.1.00713 or higher.

Running distributed simulation of experiments with MEME may require some further assistance in order to adopt our software to the specific requirements of your cluster. Had you encountered problems during execution of simulations after following this documentation, please, contact us at mass@aitia.ai for help or register on our MASS support mailing list (<https://lists.mass.aitia.ai/listinfo/mass-support>).

1 Components of a system

1.1 Preliminary

First, you should appoint a server host. Its purposes are twofold:

- runs MASS/MEME Simulation Server
- distributes tasks to *workers* as a master server

We will refer to it as *master*.

1.2 Prerequisites on all hosts

1.2.1 Operation System (OS)

The MASS/MEME Simulation Server is tested using Windows XP and Linux operation systems.

Important: all participating nodes should run the same operating system (Windows or Linux).

Some issues of cross-platform run are not resolved yet.

The node which runs the MEME may run arbitrary operating system (Windows or Linux).

1.2.2 SSH

Within the cluster all nodes should run a SSH daemon which guarantees (authenticated) login facility. (In addition the files between MEME and the MASS/MEME Simulation Server are transferred via SFTP channel).

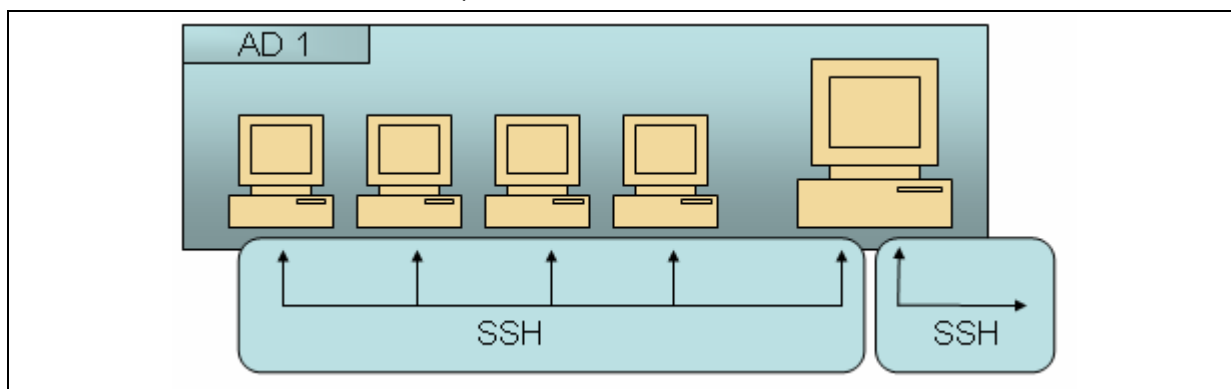


Figure 2 – The cluster

Using Linux SSH (client and daemon) is supported by default. Using Windows plink (an SSH client, see [4]) and FreeSSHd (an SSH daemon, see [3]) provides the required functionality.

If you use plink, please make sure that it can be started from the command line. This can be achieved by extending the PATH environment variable with the path of the directory where plink.exe is located.

Important: Asymmetric key-based authentication is available on both platforms (Windows and Linux, using `-l` argument), while password-based authentication is only available on Windows (using plink with `-pw` argument).

Please check if you have *ssh* and *sftp* servers on all hosts. Try to log on from *master* to all other machines. It is needed, as on first launch *ssh* initializes necessary data and requires user confirmation.

1.2.2.1 Key Exchange (Windows only)

SSH-based protocols periodically re-exchanges keys (e. g. at every hour), which may cause problems when using plink and FreeSSHd.

Fortunately, plink can load configurations saved by PuTTY using the „load“ command, so creating a configuration that disables key re-exchange with PuTTY on the master solves the problem:

- Start PuTTY and create a configuration (or *Saved Session*) named *UnlimitedKex* (see Figure 3).

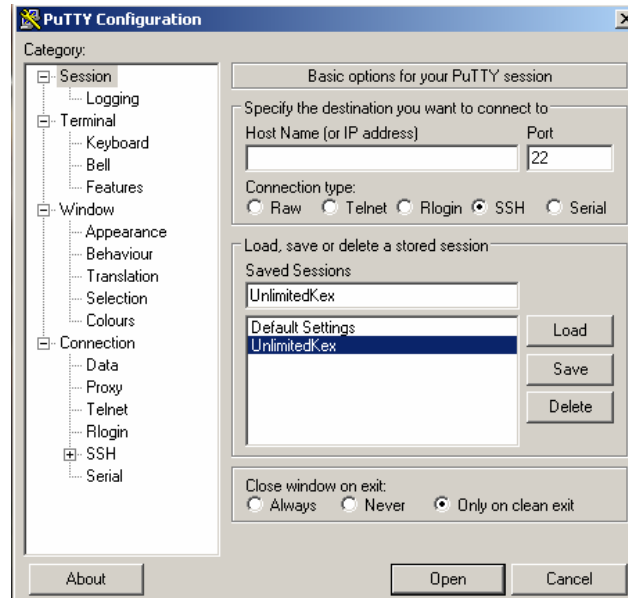


Figure 3 - Create the *UnlimitedKex* configuration

- For security purposes PuTTY automatically requests key re-exchange after a certain time or amount of exchanged data. To avoid this feature set the corresponding values to „unlimited“ by assigning 0 to them (see Figure 4).

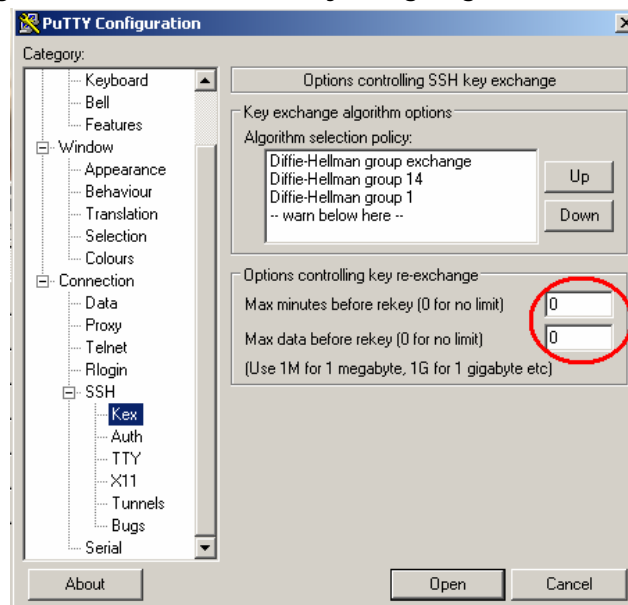


Figure 4 – Disable the automatic key re-exchange requests

- Erroneous key re-exchange is a known issue: set the configuration to handle the problem (see Figure 5).

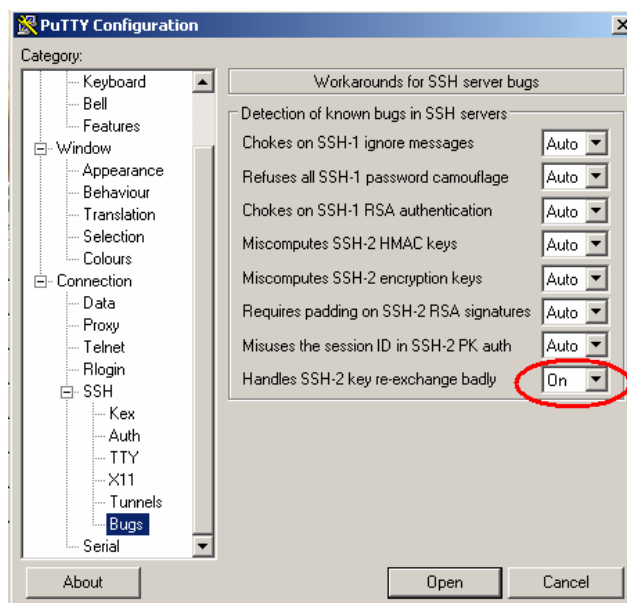


Figure 5 – Indicate key re-exchange error to PuTTY (plink)

- Finally, don't forget to save the configuration.

1.2.3 Workspace

Please define a workspace directory on every host, relative to the user's home (or default SFTP directory at Windows). For example (same on each host):

1. *username:* `user01`
2. *home:* `/home/user01`
3. *workspace:* `/home/user01/_cluster/_workspace`

Workspace directory is a parameter to set in the wizard. You need to set only the relative path. (Following the previous example: `_cluster/_workspace`).

1.2.3.1 Default SFTP Directory (Windows only)

This is the default directory after login to a node over SSH. Using FreeSSHd the default directory of the SSH server can be set on the *Settings* window:

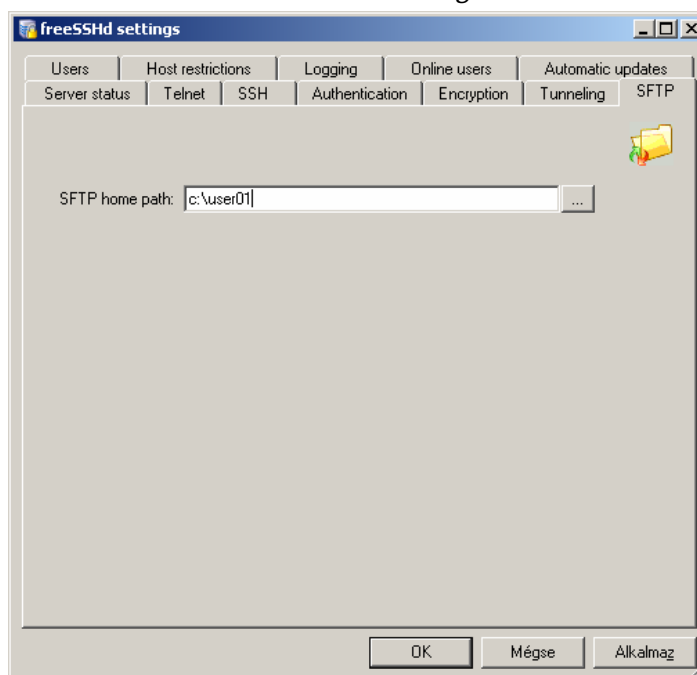


Figure 6 – Set the SSH server's home path (FreeSSHd)

1.2.4 Install Java

All of your computers in your cluster should have JRE 1.6 or later installed and runnable. Java is available at: <http://java.sun.com/javase/downloads/index.jsp>

Please make sure that java can be started from the command line. This can be achieved by extending the PATH environment variable with the path of the *bin* directory of JRE. You can verify this with the output of the `java -version` command. Here is an example for the output (the actual version numbers may be different):

```
java version "1.6.0_11"  
Java(TM) SE Runtime Environment (build 1.6.0_11-b03)  
Java HotSpot(TM) Client VM (build 11.0-b16, mixed mode, sharing)
```

Note: Some Windows platforms recognize new environment variables only after system restart.

1.2.5 ProActive 4.1.2 Library

All of your computers in your cluster should have the ProActive 4.1.2 ([2]) library somewhere. ProActive 4.1.2 is available at: <http://proactive.inria.fr/>.

Please note the MASS/MEME Simulation Server works with only this version; using previous or later versions may cause errors.

It is not mandatory however you should place ProActive-4.1.2 folder into the workspace folder if you want to use the any of the predefined launcher scripts. See details later.

1.3 Prerequisites on workers

1.3.1 MEME libraries

Copy the following files to the *lib* subdirectory of the workspace of all workers from the *meme_simulation_server.zip* archive file:

- *lib/MEME.jar*
- *lib/meme.custom.model.jar*

1.3.2 RepastJ 3.1 Library

All of your worker computers in your cluster should have the RepastJ 3.1 ([5]) library somewhere. RepastJ 3.1 is available at: http://repast.sourceforge.net/repast_3/index.html.

Important: The downloaded *lib/ProActive.jar* should be deleted; otherwise distributed simulations will cause errors.

1.3.3 Other libraries

If your cluster supports other simulation platforms (e.g. NetLogo 4.0.4) too additional libraries can be installed to all worker computers of your cluster. See in details in the section of the appropriate simulation platform.

1.3.4 Configuration files

Copy the following files to the *conf* subdirectory of the workspace of all workers from the *meme_simulation_server.zip* archive file:

- *conf/proactive.java.policy;*
- *conf/proactive-log4j;*
- *conf/ProActiveConfiguration.xml.*

See details about these files later.

1.3.5 Plugins folder

Copy the *Plugins* folder to the workspace of all workers from the *meme_simulation_server.zip* archive file.

1.3.6 PlatformPlugins folder

Create a *PlatformPlugins* directory in the workspace of all workers. At the first time it is an empty folder. Some platform-specific libraries is stored here if your cluster supports more than one simulation platforms.

1.4 Prerequisites on master

1.4.1 SSH (External Access)

The master should be able to handle external SSH requests (requests arriving outside of the network domain) to communicate with MEME Parameter Sweep Wizard.

1.4.2 MASS/MEME Simulation Server Port

The MEME Parameter Sweep Wizard uses Java socket to communicate with the server hence a listener port required on the master node. The MASS/MEME Simulation Server uses port 3000 by default (this can be changed at startup).

1.4.3 Related Files

Copy the content of the *meme_simulation_server.zip* into master's workspace directory.

- **conf (directory)**: contains all necessary configuration files used by the server. See details about them later.
- **documents (directory)**: the documentations of the server. At this time, it contains only this document.
- **lib (directory)**: Java libraries used by the server.
- **master (directory)**: An empty directory; the server uses this as a temp folder when performing a batch experiment.
- **PlatformPlugins (directory)**: it is also empty; the storage directory of plugins that supports other simulation platforms such as NetLogo 4.0.4.
- **Plugins (directory)**: stores dynamic IntelliSweep method plugins (at this time there is only one such plugin). See details about IntelliSweep method in [1].
- **results (directory)**: An empty directory; the server collects the results of all finished simulation here.
- **distributedServer.sh, distributedServer.bat**: predefined launcher scripts (Linux and Windows). Runs the MASS/MEME Simulation Server.
- **platformReg.sh, platformReg.bat**: predefined launcher scripts (Linux and Windows). Runs the MASS/MEME PlatformReg applications. See details in section 2.1 The MASS/MEME PlatformReg Application

1.4.4 Configuration Files

The configuration files are located in the *conf* subdirectory of the master's workspace. The following sections describe the meaning of each file.

1.4.4.1 The *proactive.java.policy* file

A ProActive policy file. See details in the section 2.2.3 of the ProActive Manual.

It is not recommended to modify the content of this file.

1.4.4.2 The *proactive-log4j* file

A logging policy file using by ProActive. See details in the section 2.2.4 of the ProActive Manual.

You should not modify the content of this file except in case of problems that need debugging, in order to provide us detailed information you can change the debug level in *proactive-log4j* file before sending us the output. You can achieve this by changing the line `log4j.rootLogger=INFO, A1` to `log4j.rootLogger=DEBUG, A1`.

1.4.4.3 The logging.properties file

A logging policy file using by MASS/MEME Simulation Server. See details about Java Logging in the following article: <http://java.sun.com/j2se/1.4.2/docs/guide/util/logging/overview.html>

It is not recommended to modify the content of this file.

1.4.4.4 The server_config.xml file

The MASS/MEME Simulation Server gets some of its settings from this file. You can set the following parameters in this file:

- **descriptor**: The location and name of the ProActive Descriptor file. It is not recommended to modify the default value of this parameter.
- **smtp-server**: If you want to send email notifications (at simulation end or on problems) to the uploader of the experiment, please install or use an SMTP server. The name and port number of the SMTP server can be configured here.
- **initialTaskPerWorker**: it is a technical parameter using by ProActive. See details in section 17.5.1.2 of the ProActive Manual. It is not recommended to modify the default value of this parameter.

1.4.4.5 The ProActiveConfiguration.xml file

This is the file where all ProActive related configuration is located. See details in chapter 22 of the ProActive Manual.

It is not recommended to modify the content of this file.

Important: If you change the following property in the *ProActiveConfiguration.xml* file the workers will not work: `<prop key="proactive.net.nolocal" value="true" />`.

1.4.4.6 The ProActive Descriptor (master.descriptor.xml)

This is the most important configuration file that describes the whole cluster. In this documentation we provide some example descriptors (they can also be found in the *conf/example_descriptors* subdirectory of the master's workspace) with some explanation. For further details please consult with the ProActive Manual (Chapter 23).

Example 1 (2 Windows-workers, only RepastJ support)

Let's assume the followings:

- The workspace is the *c:\DistributedParamSweep* folder.
- The JRE 1.6 is installed to the *c:\Program Files\Java\jdk1.6.0_11* folder.
- The ProActive 4.1.2 is in the *c:\DistributedParamSweep\ProActive-4.1.2* folder.
- The RepastJ 3.1 is in the *c:\DistributedParamSweep\RepastJ* folder.

```
<?xml version="1.0" encoding="UTF-8"?>
<ProActiveDescriptor xmlns="urn:proactive:deployment:3.3"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:proactive:deployment:3.3 http://www-
sop.inria.fr/oasis/ProActive/schemas/deployment/3.3/deployment.xsd">

  <!-- Example descriptor with two Windows worker (supports RepastJ platform only) -->

  <variables>
    <!-- The absolute path of java on the workers -->
    <descriptorVariable name="REMOTE_JAVA"
      value="C:\\Program Files\\Java\\jdk1.6.0_11\\bin\\java"/>
  </variables>

  <componentDefinition>
    <virtualNodesDefinition>
      <!-- Virtual Node for worker 1 -->
      <virtualNode name="remote" property="multiple"/>

      <!-- Virtual Node for worker 1 -->
```

```

    <virtualNode name="remote2" property="multiple"/>
  </virtualNodesDefinition>
</componentDefinition>

<deployment>
  <mapping>
    <!-- mapping Virtual Nodes to JVMs -->
    <map virtualNode="remote">
      <jvmSet>
        <vmName value="jvm1"/>
      </jvmSet>
    </map>

    <map virtualNode="remote2">
      <jvmSet>
        <vmName value="jvm2"/>
      </jvmSet>
    </map>
  </mapping>

  <jvms>
    <!-- Creating JVMs -->

    <!-- JVM for worker 1 -->
    <jvm name="jvm1">
      <creation>
        <processReference refid="worker_1"/>
      </creation>
    </jvm>

    <!-- JVM for worker 2 -->
    <jvm name="jvm2">
      <creation>
        <processReference refid="worker_2"/>
      </creation>
    </jvm>
  </jvms>
</deployment>

<infrastructure>
  <processes>
    <!-- Specifying JVM process for worker 1 -->
    <processDefinition id='worker_1_JVM'>
      <jvmProcess class="org.objectweb.proactive.core.process.JVMNodeProcess">
        <jvmParameters>
          <!-- Maximum heap size for the JVM -->
          <parameter value="-Xmx512M"/>

          <!-- Worker's directory (with cluster-unique name) within the workspace.
              You have to use absolute path here. -->
          <parameter value="-Duser.dir=c:\DistributedParamSweep\worker_1"/>
        </jvmParameters>

        <!-- The absolute path of java on worker 1 -->
        <javaPath>
          <absolutePath value="\${REMOTE_JAVA}"/>
        </javaPath>

        <!-- The classpath of the JVM -->
        <classpath>
          <!-- ProActive-related JARs -->
          <absolutePath value='c:\DistributedParamSweep\ProActive-4.1.2\dist\lib\*' />
          <!-- MEME-related JARs -->
          <absolutePath value='c:\DistributedParamSweep\lib\MEME.jar' />
          <absolutePath value='c:\DistributedParamSweep\lib\meme.custom.model.jar' />
          <!-- RepastJ-related JARs -->
          <absolutePath value='c:\DistributedParamSweep\RepastJ\repast.jar' />
          <absolutePath value='c:\DistributedParamSweep\RepastJ\lib\*' />

          <absolutePath value='c:\DistributedParamSweep\worker_1\' />
        </classpath>

        <!-- ProActive policy file -->
        <policyFile>
          <absolutePath value='c:\DistributedParamSweep\conf\proactive.java.policy' />
        </policyFile>
      </jvmProcess>
    </processDefinition>
  </processes>
</infrastructure>

```

```

<!-- ProActive logging properties -->
<log4jpropertiesFile>
  <absolutePath value='c:\DistributedParamSweep\conf\proactive-log4j' />
</log4jpropertiesFile>

<!-- ProActive configuration file -->
<ProActiveUserPropertiesFile>
  <absolutePath
    value='c:\DistributedParamSweep\conf\ProActiveConfiguration.xml' />
</ProActiveUserPropertiesFile>
</jvmProcess>
</processDefinition>

<!-- Specifying SSH process for worker 1 -->
<processDefinition id='worker_1'>
<!-- myhostname is the hostname of the computer on which worker 1 will run -->
<!-- myusername is the login required for an SSH connection -->
  <sshProcess class="org.objectweb.proactive.core.process.ssh.SSHProcess"
    hostname="myhostname"
    username="myusername">
    <processReference refid="worker_1_JVM"></processReference>

    <!-- plink command. mypassword is the password of myusername user -->
    <commandPath value="plink -ssh -load UnlimitedKex -pw mypassword"/>
  </sshProcess>
</processDefinition>

<!-- Specifying JVM process for worker 2 -->
<processDefinition id='worker_2_JVM'>
  <jvmProcess class="org.objectweb.proactive.core.process.JVMNodeProcess">
    <jvmParameters>
      <parameter value='-Xmx512M' />
      <parameter value='-Duser.dir=c:\DistributedParamSweep\worker_2' />
    </jvmParameters>

    <javaPath>
      <absolutePath value="{REMOTE_JAVA}" />
    </javaPath>

    <classpath>
      <!-- ProActive-related JARs -->
      <absolutePath value='c:\DistributedParamSweep\ProActive-4.1.2\dist\lib\*' />
      <!-- MEME-related JARs -->
      <absolutePath value='c:\DistributedParamSweep\lib\MEME.jar' />
      <absolutePath value='c:\DistributedParamSweep\lib\meme.custom.model.jar' />
      <!-- RepastJ-related JARs -->
      <absolutePath value='c:\DistributedParamSweep\RepastJ\repast.jar' />
      <absolutePath value='c:\DistributedParamSweep\RepastJ\lib\*' />

      <absolutePath value='c:\DistributedParamSweep\worker_2\*' />
    </classpath>

    <policyFile>
      <absolutePath value='c:\DistributedParamSweep\conf\proactive.java.policy' />
    </policyFile>

    <log4jpropertiesFile>
      <absolutePath value='c:\DistributedParamSweep\conf\proactive-log4j' />
    </log4jpropertiesFile>

    <ProActiveUserPropertiesFile>
      <absolutePath
        value='c:\DistributedParamSweep\conf\ProActiveConfiguration.xml' />
    </ProActiveUserPropertiesFile>
  </jvmProcess>
</processDefinition>

<!-- Specifying SSH process for worker 2 -->
<processDefinition id='worker_2'>
  <sshProcess class="org.objectweb.proactive.core.process.ssh.SSHProcess"
    hostname="myhostname"
    username="myusername">
    <processReference refid="worker_2_JVM"></processReference>
    <commandPath value="plink -ssh -load UnlimitedKex -pw mypassword"/>
  </sshProcess>
</processDefinition>
</processes>

```

```
</infrastructure>
</ProActiveDescriptor>
```

Notes

- It is very important that every worker JVM has own virtual node in cluster. Although ProActive allows multiple JVMs per virtual node using this feature may cause problems in the simulation runs.
- Every worker needs a cluster-unique name and a subdirectory with the same name in the workspace. However you don't have to create these subdirectories manually; the MASS/MEME Simulation Server will create them at the first experiment performing.
- In order to achieve optimal performance the number of defined workers should be lesser than or equal to the number of CPU cores on every machine of the cluster.
- The example descriptor specifies 512 Mb maximum heap size to all workers. If your computers have more RAM feel free to modify the maximum heap size.
- As you may see the * wildcard can be used in the class path definition which means all JAR files of the referenced directory will be part of the class path of the worker's JVM.
- The descriptor file of this example can be found in the *conf/example_descriptors* subdirectory of the master's workspace. Its name is *master.descriptor.win.2workers.repast.xml*.

Example 2 (2 Linux workers in a grid, only RepastJ support)

Let's assume the followings:

- The master and its two workers are parts of a grid. They can connect each other via SSH without any authentication. However reaching the master via SSH from outside still needs a username/password pair.
- There is a user named *paramsweeper* on all machines of the cluster.
- The workspace is the */home/paramsweeper/_cluster_workspace/dps* folder. (In MEME Parameter Sweep Wizard you have to specify a relative path of the workspace which is in this case *_cluster_workspace/dps*).
- The JRE 1.6 is installed to the */usr/bin* folder.
- The ProActive 4.1.2 is in the */home/paramsweeper/_cluster_workspace/dps/ProActive-4.1.2* folder.
- The RepastJ 3.1 is in the */home/paramsweeper/_cluster_workspace/dps/RepastJ* folder.

```
<?xml version="1.0" encoding="UTF-8"?>
<ProActiveDescriptor xmlns="urn:proactive:deployment:3.3"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:proactive:deployment:3.3 http://www-
sop.inria.fr/oasis/ProActive/schemas/deployment/3.3/deployment.xsd">

  <!-- Example descriptor with two Linux workers (supports RepastJ platform only)
  The master and its two workers are parts of a grid. They can connect each other
  via SSH without any authentication. -->

  <variables>
    <descriptorVariable name="REMOTE_JAVA" value="/usr/bin/java"/>
  </variables>

  <componentDefinition>
    <virtualNodesDefinition>
      <!-- Virtual Node for worker 1 -->
      <virtualNode name="vn1" property="multiple"/>

      <!-- Virtual Node for worker 2 -->
      <virtualNode name="vn2" property="multiple"/>
    </virtualNodesDefinition>
  </componentDefinition>
```

```

<deployment>
  <mapping>
    <!-- mapping Virtual Nodes to JVMs -->
    <map virtualNode="vn1">
      <jvmSet>
        <vmName value="grid3"/>
      </jvmSet>
    </map>

    <map virtualNode="vn2">
      <jvmSet>
        <vmName value="grid4"/>
      </jvmSet>
    </map>
  </mapping>

  <jvms>
    <!-- Creating JVMs -->

    <!-- JVM for worker 1 -->
    <jvm name="grid3">
      <creation>
        <processReference refid="worker_1"/>
      </creation>
    </jvm>

    <!-- JVM for worker 2 -->
    <jvm name="grid4">
      <creation>
        <processReference refid="worker_2"/>
      </creation>
    </jvm>
  </jvms>
</deployment>

<infrastructure>
  <processes>
    <!-- Specifying JVM process for worker 1 -->
    <processDefinition id="worker_1_JVM">
      <jvmProcess class="org.objectweb.proactive.core.process.JVMNodeProcess">
        <jvmParameters>
          <!-- Maximum heap size for the JVM -->
          <parameter value="-Xmx512M"/>
          <!-- Worker's directory (with cluster-unique name) within the workspace. You
          have to use absolute path here. -->
          <parameter
            value="-Duser.dir=/home/paramsweeper/_cluster_workspace/dps/worker_1"/>
        </jvmParameters>

        <!-- The absolute path of java on worker 1 -->
        <javaPath>
          <absolutePath value="{REMOTE_JAVA}"/>
        </javaPath>

        <!-- The classpath of the JVM -->
        <classpath>
          <!-- ProActive-related JARs -->
          <absolutePath
            value='/home/paramsweeper/_cluster_workspace/dps/ProActive-4.1.2/dist/lib/*' />
          <!-- MEME-related JARs -->
          <absolutePath value='/home/paramsweeper/_cluster_workspace/dps/lib/MEME.jar' />
          <absolutePath
            value='/home/paramsweeper/_cluster_workspace/dps/lib/meme.custom.model.jar' />
          <!-- RepastJ-related JARs -->
          <absolutePath
            value='/home/paramsweeper/_cluster_workspace/dps/RepastJ/repast.jar' />
          <absolutePath
            value='/home/paramsweeper/_cluster_workspace/dps/RepastJ/lib/*' />

          <absolutePath value='/home/paramsweeper/_cluster_workspace/dps/worker_1' />
        </classpath>

        <!-- ProActive policy file -->
        <policyFile>
          <absolutePath
            value='/home/paramsweeper/_cluster_workspace/dps/conf/proactive.java.policy' />
        </policyFile>
      </jvmProcess>
    </processDefinition>
  </processes>
</infrastructure>

```

```

<!-- ProActive logging properties -->
<log4jpropertiesFile>
  <absolutePath
    value='/home/paramsweeper/_cluster_workspace/dps/conf/proactive-log4j' />
</log4jpropertiesFile>

<!-- ProActive configuration file -->
<ProActiveUserPropertiesFile>
  <absolutePath
    value='/home/paramsweeper/_cluster_workspace/dps/conf/ProActiveConfiguration.xml' />
</ProActiveUserPropertiesFile>
</jvmProcess>
</processDefinition>

<!-- Specifying SSH process for worker 1 -->
<processDefinition id="worker_1">
  <!-- Only hostname and username have to be specified. -->
  <sshProcess class="org.objectweb.proactive.core.process.ssh.SSHProcess
    hostname="grid3"
    username="paramsweeper">
    <processReference refid="worker_1_JVM"></processReference>
  </sshProcess>
</processDefinition>

<!-- Specifying JVM process for worker 2 -->
<processDefinition id="worker_2_JVM">
  <jvmProcess class="org.objectweb.proactive.core.process.JVMNodeProcess">
    <jvmParameters>
      <parameter value="-Xmx512M" />
      <parameter
        value="-Duser.dir=/home/paramsweeper/_cluster_workspace/dps/worker_2/" />
    </jvmParameters>

    <javaPath>
      <absolutePath value="{REMOTE_JAVA}" />
    </javaPath>

    <classpath>
      <!-- ProActive-related JARs -->
      <absolutePath
        value='/home/paramsweeper/_cluster_workspace/dps/ProActive-4.1.2/dist/lib/*' />
      <!-- MEME-related JARs -->
      <absolutePath value='/home/paramsweeper/_cluster_workspace/dps/lib/MEME.jar' />
      <absolutePath
        value='/home/paramsweeper/_cluster_workspace/dps/lib/meme.custom.model.jar' />
      <!-- RepastJ-related JARs -->
      <absolutePath
        value='/home/paramsweeper/_cluster_workspace/dps/RepastJ/repast.jar' />
      <absolutePath
        value='/home/paramsweeper/_cluster_workspace/dps/RepastJ/lib/*' />

      <absolutePath value='/home/paramsweeper/_cluster_workspace/dps/worker_2' />
    </classpath>

    <policyFile>
      <absolutePath
        value='/home/paramsweeper/_cluster_workspace/dps/conf/proactive.java.policy' />
    </policyFile>

    <log4jpropertiesFile>
      <absolutePath
        value='/home/paramsweeper/_cluster_workspace/dps/conf/proactive-log4j' />
    </log4jpropertiesFile>

    <ProActiveUserPropertiesFile>
      <absolutePath
        value='/home/paramsweeper/_cluster_workspace/dps/conf/ProActiveConfiguration.xml' />
    </ProActiveUserPropertiesFile>
  </jvmProcess>
</processDefinition>

<!-- Specifying SSH process for worker 2 -->
<processDefinition id="worker_2">
  <sshProcess class="org.objectweb.proactive.core.process.ssh.SSHProcess"
    hostname="grid4"
    username="paramsweeper">

```

```

        <processReference refid="worker_2_JVM"></processReference>
    </sshProcess>
</processDefinition>
</processes>
</infrastructure>
</ProActiveDescriptor>

```

Notes

- The descriptor file of this example can be found in the *conf/example_descriptors* subdirectory of the master's workspace. Its name is *master.descriptor.linux.2workers.repast.noauth.xml*.

Example 3 (2 Linux workers, only RepastJ support)

Let's assume the followings:

- The master and its two workers connect each other via SSH using key-based authentication.
- There is a user named *paramsweeper* on all machines of the cluster.
- The workspace is the */home/paramsweeper/_cluster_workspace/dps* folder. (In MEME Parameter Sweep Wizard you have to specify a relative path of the workspace which is in this case *_cluster_workspace/dps*).
- The JRE 1.6 is installed to the */usr/bin* folder.
- The ProActive 4.1.2 is in the */home/paramsweeper/_cluster_workspace/dps/ProActive-4.1.2* folder.
- The RepastJ 3.1 is in the */home/paramsweeper/_cluster_workspace/dps/RepastJ* folder.

```

<?xml version="1.0" encoding="UTF-8"?>
<ProActiveDescriptor xmlns="urn:proactive:deployment:3.3"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:proactive:deployment:3.3 http://www-
sop.inria.fr/oasis/ProActive/schemas/deployment/3.3/deployment.xsd">

<!-- Example descriptor with two Linux workers (supports RepastJ platform only) using key-
based authentication -->

<variables>
<descriptorVariable name="REMOTE_JAVA" value="/usr/bin/java"/>
<descriptorVariable name="PRIVATE_KEY_FILE"
    value="/home/paramsweeper/.ssh/private_keys/key"/>
</variables>

<componentDefinition>
<virtualNodesDefinition>
<!-- Virtual Node for worker 1 -->
<virtualNode name="vn1" property="multiple"/>

<!-- Virtual Node for worker 2 -->
<virtualNode name="vn2" property="multiple"/>
</virtualNodesDefinition>
</componentDefinition>

<deployment>
<mapping>
<!-- mapping Virtual Nodes to JVMs -->
<map virtualNode="vn1">
<jvmSet>
<vmName value="jvm1"/>
</jvmSet>
</map>

<map virtualNode="vn2">
<jvmSet>
<vmName value="jvm2"/>
</jvmSet>
</map>

```

```

</mapping>

<jvms>
  <!-- Creating JVMs -->

  <!-- JVM for worker 1 -->
  <jvm name="jvm1">
    <creation>
      <processReference refid="worker_1"/>
    </creation>
  </jvm>

  <!-- JVM for worker 2 -->
  <jvm name="jvm2">
    <creation>
      <processReference refid="worker_2"/>
    </creation>
  </jvm>
</jvms>
</deployment>

<infrastructure>
  <processes>
    <!-- Specifying JVM process for worker 1 -->
    <processDefinition id="worker_1_JVM">
      <jvmProcess class="org.objectweb.proactive.core.process.JVMNodeProcess">
        <jvmParameters>
          <!-- Maximum heap size for the JVM -->
          <parameter value="-Xmx512M"/>
          <!-- Worker's directory (with cluster-unique name) within the workspace. You
            have to use absolute path here. -->
          <parameter
            value="-Duser.dir=/home/paramsweeper/_cluster_workspace/dps/worker_1"/>
        </jvmParameters>

        <!-- The absolute path of java on worker 1 -->
        <javaPath>
          <absolutePath value="{REMOTE_JAVA}"/>
        </javaPath>

        <!-- The classpath of the JVM -->
        <classpath>
          <!-- ProActive-related JARs -->
          <absolutePath
            value='/home/paramsweeper/_cluster_workspace/dps/ProActive-4.1.2/dist/lib/*' />
          <!-- MEME-related JARs -->
          <absolutePath value='/home/paramsweeper/_cluster_workspace/dps/lib/MEME.jar' />
          <absolutePath
            value='/home/paramsweeper/_cluster_workspace/dps/lib/meme.custom.model.jar' />
          <!-- RepastJ-related JARs -->
          <absolutePath
            value='/home/paramsweeper/_cluster_workspace/dps/RepastJ/repast.jar' />
          <absolutePath
            value='/home/paramsweeper/_cluster_workspace/dps/RepastJ/lib/*' />

          <absolutePath value='/home/paramsweeper/_cluster_workspace/dps/worker_1' />
        </classpath>

        <!-- ProActive policy file -->
        <policyFile>
          <absolutePath
            value='/home/paramsweeper/_cluster_workspace/dps/conf/proactive.java.policy' />
        </policyFile>

        <!-- ProActive logging properties -->
        <log4jpropertiesFile>
          <absolutePath
            value='/home/paramsweeper/_cluster_workspace/dps/conf/proactive-log4j' />
        </log4jpropertiesFile>

        <!-- ProActive configuration file -->
        <ProActiveUserPropertiesFile>
          <absolutePath
            value='/home/paramsweeper/_cluster_workspace/dps/conf/ProActiveConfiguration.xml' />
        </ProActiveUserPropertiesFile>
      </jvmProcess>
    </processDefinition>
  </processes>
</infrastructure>

```

```

<!-- Specifying SSH process for worker 1 -->
<processDefinition id="worker_1">
  <!--Hostname, username and private key file have to be specified. -->
  <sshProcess class="org.objectweb.proactive.core.process.ssh.SSHProcess
    hostname="host1"
    username="paramsweeper">
    <processReference refid="worker_1_JVM"></processReference>
    <commandPath value="ssh -i ${PRIVATE_KEY_FILE}"/>
  </sshProcess>
</processDefinition>

<!-- Specifying JVM process for worker 2 -->
<processDefinition id="worker_2_JVM">
  <jvmProcess class="org.objectweb.proactive.core.process.JVMNodeProcess">
    <jvmParameters>
      <parameter value="-Xmx512M"/>
      <parameter
        value="-Duser.dir=/home/paramsweeper/_cluster_workspace/dps/worker_2"/>
    </jvmParameters>

    <javaPath>
      <absolutePath value="${REMOTE_JAVA}"/>
    </javaPath>

    <classpath>
      <!-- ProActive-related JARs -->
      <absolutePath
        value='/home/paramsweeper/_cluster_workspace/dps/ProActive-4.1.2/dist/lib/*' />
      <!-- MEME-related JARs -->
      <absolutePath value='/home/paramsweeper/_cluster_workspace/dps/lib/MEME.jar' />
      <absolutePath
        value='/home/paramsweeper/_cluster_workspace/dps/lib/meme.custom.model.jar' />
      <!-- RepastJ-related JARs -->
      <absolutePath
        value='/home/paramsweeper/_cluster_workspace/dps/RepastJ/repast.jar' />
      <absolutePath
        value='/home/paramsweeper/_cluster_workspace/dps/RepastJ/lib/*' />

      <absolutePath value='/home/paramsweeper/_cluster_workspace/dps/worker_2' />
    </classpath>

    <policyFile>
      <absolutePath
        value='/home/paramsweeper/_cluster_workspace/dps/conf/proactive.java.policy' />
    </policyFile>

    <log4jpropertiesFile>
      <absolutePath
        value='/home/paramsweeper/_cluster_workspace/dps/conf/proactive-log4j' />
    </log4jpropertiesFile>

    <ProActiveUserPropertiesFile>
      <absolutePath
        value='/home/paramsweeper/_cluster_workspace/dps/conf/ProActiveConfiguration.xml' />
    </ProActiveUserPropertiesFile>
  </jvmProcess>
</processDefinition>

<!-- Specifying SSH process for worker 2 -->
<processDefinition id="worker_2">
  <sshProcess class="org.objectweb.proactive.core.process.ssh.SSHProcess"
    hostname="host2"
    username="paramsweeper">
    <processReference refid="worker_2_JVM"></processReference>
    <commandPath value="ssh -i ${PRIVATE_KEY_FILE}"/>
  </sshProcess>
</processDefinition>
</processes>
</infrastructure>
</ProActiveDescriptor>

```

Notes

- The descriptor file of this example can be found in the *conf/example_descriptors* subdirectory of the master's workspace. Its name is *master.descriptor.linux.2workers.repast.auth.xml*.

1.4.5 Launcher Scripts

MASS/MEME Simulation Server can be launched via script files (*distributedServer.bat/distributedServer.sh*) found in master's workspace directory. These script files assume that the master's workspace is *c:\DistributedParamSweep (/home/paramsweeper/distributedParamSweeper* on Linux) folder and the ProActive-4.1.2 folder is located in the workspace folder. If any of these assumptions are wrong, you have to modify the launcher script (see details below).

Those scripts contain no port parameters and the server uses default port 3000. This can be changed as desired (by specifying the port number at the end of the script), but then do not forget setting it on client side among MEME preferences, too. The port used should be opened in intervening firewalls, if any.

1.4.5.1 Modifying the Launcher Scripts

Here's the content of the *distributedServer.bat* script. It is marked where you should changed the command line arguments:

```
java -Djava.security.policy=conf/proactive.java.policy
-Dlog4j.configuration=file:conf/proactive-log4j
-Dproactive.configuration=conf/ProActiveConfiguration.xml
-Dparamsweep.distr.master.workspace=/DistributedParamSweep
-Djava.util.logging.config.file=conf/logging.properties
-cp lib/dps.jar;ProActive-4.1.2/dist/lib/*;lib/MEME.jar;
    lib/meme.custom.model.jar;Plugins/intellisweepPlugin.jar
ai.aitia.meme.paramsweep.server.MEMESimulationServer
```

The *paramsweep.distr.master.workspace* argument specifies the absolute path of the master's workspace.

The other arguments describe the locations of the configuration files.

If you installed ProActive 4.1.2 to an other location you must modify the corresponding class path entry as well.

2 Installing Simulation Platforms

This chapter describes in detail how to install a new simulation platform such as NetLogo 4.0.4 under the MASS/MEME Simulation Server.

2.1 The MASS/MEME PlatformReg Application

During the installation process you have to register the new simulation platform with the MASS/MEME PlatformReg command-line application. This section shows the usage of this program.

MASS/MEME PlatformReg program can be launched via the *platformReg.bat* (*platformReg.sh* on Linux) script file.

Running without any command line arguments prints the help to the standard output:

```
MASS/MEME PlatformReg application v0.9.90326
Copyright 2009 - 2010 Aitia International, Inc.

USAGE:

platformReg.bat [<command>]*

where <command> can be one of the following:
  list: lists the currently registered platforms with their installation
        directories
  +platform=<platform-id>;<directory>: add a new registered platforms or
        edit an existing one
  -platform=<platform-id>: remove a registered platform

Notes:

1. The <platform-id> can be one of the following strings:
   CUSTOM, NETLOGO
2. The <directory> is the installation directory of the platform. If the path
   contains space, it must be marked with double quotes.
3. You can execute more than one +platform and/or -platform command
   simultaneously by separating them with spaces.
```

Please note that on Linux operation system you have to use colon (:) instead of semi-colon (;) in the `+platform` command. For example: `platformReg.sh +platform=NETLOGO:/usr/bin/netlogo_4.0.4`

2.2 Installing Platform for Custom Java Models

In this section we discuss the installation process on Windows operation system. Installing to Linux is very similar, there are only slight differences (e.g. you have to use *platformReg.sh* launcher script instead of *platformReg.bat*, etc.).

- **Step 1:** If you download this platform, a simple ZIP file named *customjavaplugin.zip* to be given. Extract it.
- **Step 2:** Copy *CustomJavaPlugin.jar* to the *PlatformPlugins* directory located in the master's workspace. Repeat this step on all workers.
- **Step 3:** Use the MASS/MEME PlatformReg application on the master machine to register the new simulations platform by using the following command:

```
platformReg.bat +platform=CUSTOM;.
```

Since there is no installation directory of the simulation platform the current directory (.) is used.

- **Step 4:** Restart the MASS/MEME Simulation Server.

2.3 Installing NetLogo 4.0.4 Platform

In this section we discuss the installation process on Windows operation system. Installing to Linux is very similar, there are only slight differences (e.g. you have to use *platformReg.sh* launcher script instead of *platformReg.bat*, etc.).

- **Step 0:** To use MASS/MEME Simulation Server with NetLogo, you have to be installed NetLogo 4.0.4 on all of the computers of the cluster. So if you don't have one on all of your machines, install it. You can download NetLogo 4.0.4 from the official NetLogo website (see [6]). In the followings we assume that the NetLogo 4.0.4 is installed to the *c:\NetLogo_4.0.4* folder on every machine.
- **Step 1:** Extract the *netlogoplugin.zip* (it can be downloaded from our site: <http://mass.aitia.ai>) file to a temporary folder. The only file you will need is *NetLogoPlugin.jar* (the other files are used only when you installing the simulation platform on client side).
- **Step 2:** Copy *NetLogoPlugin.jar* to the *PlatformPlugins* directory located in the master's workspace. Repeat this step on all workers.
- **Step 3:** Edit *platformReg.bat* on the master computer to extend the class path with *NetLogo.jar*. If the NetLogo 4.0.4 is installed to the *c:\NetLogo_4.0.4* folder the content of *platformReg.bat* is changed to the followings:

```
@echo OFF
java -cp
lib/dps.jar;lib/MEME.jar;lib/meme.custom.model.jar;c:/NetLogo_4.0.4/NetLogo.
jar ai.aitia.meme.paramsweep.utils.PlatformReg %*
@echo ON
```

- **Step 4:** Use MASS/MEME PlatformReg application to register the new simulation platform by using the following command:

```
platformReg.bat +platform=NETLOGO;c:\NetLogo_4.0.4
```

- **Step 5:** Edit *distributedServer.bat* to extend the class path with *NetLogo.jar*. If the NetLogo 4.0.4 is installed to the *c:\NetLogo_4.0.4* folder the content of *distributedServer.bat* is changed to the followings:

```
java -Djava.security.policy=conf/proactive.java.policy
-Dlog4j.configuration=file:conf/proactive-log4j
-Dproactive.configuration=conf/ProActiveConfiguration.xml
-Dparamsweep.distr.master.workspace=/DistributedParamSweep
-Djava.util.logging.config.file=conf/logging.properties
-cp lib/dps.jar;ProActive-4.1.2/dist/lib/*;lib/MEME.jar;
lib/meme.custom.model.jar;Plugins/intellisweepPlugin.jar;
c:/NetLogo_4.0.4/NetLogo.jar
ai.aitia.meme.paramsweep.server.MEMESimulationServer
```

- **Step 6:** You have to modify the ProActive descriptor file (*master.descriptor.xml*) by extending the class path of all workers with the JAR files related to NetLogo 4.0.4. The following section shows an example descriptor that supports NetLogo 4.0.4 simulation platform too besides RepastJ.
- **Step 7:** Restart MASS/MEME Simulation Server.

2.3.1 Example Descriptor

(2 Windows-workers, supports RepastJ and NetLogo 4.0.4 platforms)

Let's assume the followings:

- The workspace is the *c:\DistributedParamSweep* folder.
- The JRE 1.6 is installed to the *c:\Program Files\Java\jdk1.6.0_11* folder.
- The ProActive 4.1.2 is in the *c:\DistributedParamSweep\ProActive-4.1.2* folder.
- The RepastJ 3.1 is in the *c:\DistributedParamSweep\RepastJ* folder.
- The NetLogo 4.0.4 is installed to the *c:\NetLogo_4.0.4* folder.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

<ProActiveDescriptor xmlns="urn:proactive:deployment:3.3"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:schemaLocation="urn:proactive:deployment:3.3 http://www-
sop.inria.fr/oasis/ProActive/schemas/deployment/3.3/deployment.xsd">

  <!-- Example descriptor with two Windows worker (supports RepastJ and NetLogo 4.0.4
platforms) -->

  <variables>
    <!-- The absolute path of java on the workers -->
    <descriptorVariable name="REMOTE_JAVA"
                        value="C:\\Program Files\\Java\\jdk1.6.0_11\\bin\\java"/>
  </variables>

  <componentDefinition>
    <virtualNodesDefinition>
      <!-- Virtual Node for worker 1 -->
      <virtualNode name="remote" property="multiple"/>

      <!-- Virtual Node for worker 1 -->
      <virtualNode name="remote2" property="multiple"/>
    </virtualNodesDefinition>
  </componentDefinition>

  <deployment>
    <mapping>
      <!-- mapping Virtual Nodes to JVMs -->
      <map virtualNode="remote">
        <jvmSet>
          <vmName value="jvm1"/>
        </jvmSet>
      </map>

      <map virtualNode="remote2">
        <jvmSet>
          <vmName value="jvm2"/>
        </jvmSet>
      </map>
    </mapping>

    <jvms>
      <!-- Creating JVMs -->

      <!-- JVM for worker 1 -->
      <jvm name="jvm1">
        <creation>
          <processReference refid="worker_1"/>
        </creation>
      </jvm>
      <!-- JVM for worker 2 -->
      <jvm name="jvm2">
        <creation>
          <processReference refid="worker_2"/>
        </creation>
      </jvm>
    </jvms>
  </deployment>

  <infrastructure>
    <processes>
      <!-- Specifying JVM process for worker 1 -->
      <processDefinition id='worker_1_JVM'>
        <jvmProcess class="org.objectweb.proactive.core.process.JVMNodeProcess">
          <jvmParameters>
            <!-- Maximum heap size for the JVM -->
            <parameter value='-Xmx512M' />
            <!-- Worker's directory (with cluster-unique name) within the workspace. You
have to use absolute path here. -->
            <parameter value='-Duser.dir=c:\\DistributedParamSweep\\worker_1' />
          </jvmParameters>

          <!-- The absolute path of java on worker 1 -->
          <javaPath>
            <absolutePath value="{REMOTE_JAVA}" />
          </javaPath>

          <!-- The classpath of the JVM -->

```

```

<classpath>
  <absolutePath value='c:\NetLogo_4.0.4\lib\asm-3.0.jar' />
  <!-- ProActive-related JARs -->
  <absolutePath value='c:\DistributedParamSweep\ProActive-4.1.2\dist\lib\*' />
  <!-- MEME-related JARs -->
  <absolutePath value='c:\DistributedParamSweep\lib\MEME.jar' />
  <absolutePath value='c:\DistributedParamSweep\lib\meme.custom.model.jar' />
  <!-- RepastJ-related JARs -->
  <absolutePath value='c:\DistributedParamSweep\RepastJ\repast.jar' />
  <absolutePath value='c:\DistributedParamSweep\RepastJ\lib\*' />
  <!-- NetLogo 4.0.4-related JARs -->
  <absolutePath value='c:\NetLogo_4.0.4\NetLogo.jar' />
  <absolutePath value='c:\NetLogo_4.0.4\lib\*' />

  <absolutePath value='c:\DistributedParamSweep\worker_1\' />
</classpath>

<!-- ProActive policy file -->
<policyFile>
  <absolutePath value='c:\DistributedParamSweep\conf\proactive.java.policy' />
</policyFile>

<!-- ProActive logging properties -->
<log4jpropertiesFile>
  <absolutePath value='c:\DistributedParamSweep\conf\proactive-log4j\' />
</log4jpropertiesFile>

<!-- ProActive configuration file -->
<ProActiveUserPropertiesFile>
  <absolutePath
    value='c:\DistributedParamSweep\conf\ProActiveConfiguration.xml' />
</ProActiveUserPropertiesFile>
</jvmProcess>
</processDefinition>

<!-- Specifying SSH process for worker 1 -->
<processDefinition id='worker_1'>
  <!-- myhostname is the hostname of the computer on which worker 1 will run -->
  <!-- myusername is the login required for an SSH connection -->
  <sshProcess class="org.objectweb.proactive.core.process.ssh.SSHProcess"
    hostname="myhostname"
    username="myusername">
    <processReference refid="worker_1_JVM"></processReference>

    <!-- plink command. mypassword is the password of myusername user -->
    <commandPath value="plink -ssh -load UnlimitedKex -pw mypassword"/>
  </sshProcess>
</processDefinition>

<!-- Specifying JVM process for worker 2 -->
<processDefinition id='worker_2_JVM'>
  <jvmProcess class="org.objectweb.proactive.core.process.JVMNodeProcess">
    <jvmParameters>
      <parameter value='-Xmx512M' />
      <parameter value='-Duser.dir=c:\DistributedParamSweep\worker_2\' />
    </jvmParameters>

    <javaPath>
      <absolutePath value="{REMOTE_JAVA}" />
    </javaPath>

    <classpath>
      <absolutePath value='c:\NetLogo_4.0.4\lib\asm-3.0.jar' />
      <!-- ProActive-related JARs -->
      <absolutePath value='c:\DistributedParamSweep\ProActive-4.1.2\dist\lib\*' />
      <!-- MEME-related JARs -->
      <absolutePath value='c:\DistributedParamSweep\lib\MEME.jar' />
      <absolutePath value='c:\DistributedParamSweep\lib\meme.custom.model' />
      <!-- RepastJ-related JARs -->
      <absolutePath value='c:\DistributedParamSweep\RepastJ\repast.jar' />
      <absolutePath value='c:\DistributedParamSweep\RepastJ\lib\*' />
      <!-- NetLogo 4.0.4-related JARs -->
      <absolutePath value='c:\NetLogo_4.0.4\NetLogo.jar' />
      <absolutePath value='c:\NetLogo_4.0.4\lib\*' />

      <absolutePath value='c:\DistributedParamSweep\worker_2\' />
    </classpath>

```

```
<policyFile>
  <absolutePath value='c:\DistributedParamSweep\conf\proactive.java.policy' />
</policyFile>

<log4jpropertiesFile>
  <absolutePath value='c:\DistributedParamSweep\conf\proactive-log4j' />
</log4jpropertiesFile>

<ProActiveUserPropertiesFile>
  <absolutePath
    value='c:\DistributedParamSweep\conf\ProActiveConfiguration.xml' />
</ProActiveUserPropertiesFile>
</jvmProcess>
</processDefinition>

<!-- Specifying SSH process for worker 2 -->
<processDefinition id='worker_2'>
  <sshProcess class="org.objectweb.proactive.core.process.ssh.SSHProcess"
    hostname="myhostname"
    username="myusername">
    <processReference refid="worker_2_JVM"></processReference>
    <commandPath value="plink -ssh -load UnlimitedKex -pw mypassword" />
  </sshProcess>
</processDefinition>
</processes>
</infrastructure>
</ProActiveDescriptor>
```

Notes

- Please note that *asm-3.0.jar* from the NetLogo 4.0.4 installation folder come **before** the ProActive library in the class path of the workers. It is necessary because both NetLogo 4.0.4 and ProActive 4.1.2 use asm and the recent version of asm can be found in the NetLogo 4.0.4 installation folder (in its lib subdirectory).
- The descriptor file of this example can be found in the *conf/example_descriptors* subdirectory of the master's workspace. Its name is *master.descriptor.win.2workers.netlogo.xml*.

3 After system setup

After server has successfully been started, the MEME Parameter Sweep Wizard can be used for setting up client side parameters. Please consult the MEME manual ([1]) for details.

4 Good to know

Despite our efforts to make clean shutdown of every JVMs that are dynamically launched during simulation, there might be circumstances under which this is not possible and as a result JVMs might be hanging around unwanted. If after killing of simulation server other JVMs appears to be running on any workers, their manual shutdown may be necessary. Be sure not to kill on-going simulations!

5 References

- [1] MEME User Manual <http://mass.aitia.ai>
- [2] ProActive 4.1.2 (INRIA) <http://proactive.inria.fr/>
- [3] FreeSSHd (Kresimir Grofelnik) 1.2.4 <http://www.freesshd.com/>
- [4] Putty beta (Simon Tatham) 0.60
<http://www.chiark.greenend.org.uk/~sgtatham/putty/>
- [5] Repast 3.1 Java (Argonne National Laboratory) <http://repast.sourceforge.net>
- [6] NetLogo 4.0.4 <http://ccl.northwestern.edu/netlogo/>