AgentTeamwork User Manual

Munehiro Fukuda*
Miriam Wallace*
Jumpei Miyauchi^

* Computing and Software Systems, University of Washington, Bothell
^ Department of Computer Science, Ehime University

Table of Contents

1. System Installation

1.1 Availability

The system currently supports bash on Linux, bash on Mac OS X, and DOS command prompt on Windows XP. All the class files have been generated with Java version 1.5. They can be also compiled and executed with Java version 1.6.

1.2 Downloading and Extraction

AgentTeamwork does not require root accesses at all. You can simply install it onto your home directory with an ordinary user account. To download the latest version of the AgentTeamwork system, go through the following steps:

1. Create a tripod account. If you belong to Distributed Systems Laboratory at UW Bothell or Multimedia Database Laboratory at Keio SFC, use the agentteamwork or the mdbl.sfc account, and skip to step 3.
2. Email us at dslab@u.washington.edu. Your email should include your name, affiliation, and tripod account. Wait for our response stating that you have been authorized to download the system.
3. Visit http://agentteamwork.tripod.com/release/ateam.tar.gz You will be asked to type your tripod account and password in order to download this zipped file.

To install the system, follow the instructions given below:

Linux/Mac OS X users:

Mac users need to start bash.

1. Move ateam.tar.gz to your home directory if it is not stored there: `mv ateam.tar.gz ~`
2. Unzip the file: `gzip -d ateam.tar.gz`
3. Extract all files: `tar -xvf - < ateam.tar`
4. Set up your ".bashrc" file:

```
ATeam=$HOME/agentteamwork
export ATeam
PATH=$PATH:$ATeam/scripts
```

5. Reinitialize ".bashrc": `source .bashrc`
6. Recompile the source if you want to revise AgentTeamwork to Java 1.6:
   $ATeam/scripts/`compileAndPackAll.sh`

Windows XP users:

1. Unzip and extract all files through 7-Zip, Explzh, or any available windows-based file archiving tool.
2. Move the ateam folder to C:\Documents and Setting\%USERNAME%.
3. Set up the ATeam and PATH environment variables by clicking "Environment Variables" in My Computer's Profile menu or using Rapid Environment Editor:

```
set ATeam="C:\Documents and Setting\%USERNAME%"
set path=%path%;%ATeam%\bat
```

4. Recompile the source if you want to revise AgentTeamwork to Java 1.6:
   %ATeam%\bat\compileAndPackAll.bat

Linux users at UW1-320, UW Bothell

No need to download and to recompile the system.

1. Simply set up your "~/.bashrc" as follows:

```
ATeam = /home/uwagent/agentteamwork
export ATeam
```

```
PATH=$PATH:$ATeam/scripts
```
2. Reinitialize ".bashrc": **`source ~/.bashrc`**

Mac24 users at Keio SFC

No need to download and to recompile the system. Note that Mac users need to run bash.

1. Simply set up your "~/.bashrc" as follows:
```
ATeam = /home/mfukuda/CNSiMac/agentteamwork
export ATeam
PATH=$PATH:$ATeam/scripts
```
2. Reinitialize ".bashrc": **`source ~/.bashrc`**

Note that this installation should be done only once as far as all the computers you will use are connected to the same NFS. Otherwise, repeat the above installation work at each non-NFS-connected computer accordingly.

2. System Invocation

AgentTeamwork runs on top of a network of UWAgent mobile-agent execution daemons. You need to start up this daemon (named UWPlace) at each of all computing nodes that you will use for your job execution.

Once a computing node has the UWPlace daemon running, it becomes an available location for agents to migrate and perform tasks.

2.1 Intra-Domain UWPlace Invocation and Shutdown

This invocation assumes that IP ports above 5000 are open to all computing nodes within the same IP domain (such as uw1-320-00 ~ uw1-320-31.uwb.edu, mnode0 ~ mnode31.uwb.edu or zmac000 ~ zmac159.sfc.keio.ac.jp). Two shell scripts are available to launch UWPlace: sshUWPlace.sh and runUWPlace.sh.

Use sshUWPlace.sh for all computing nodes that run bash and identify the ATeam environment variable with the same path, (e.g., Linux/Mac machines connected to the same NFS). Log in one of those computers and type as follows:

sshUWPlace.sh:

```
sshUWPlace.sh <port#> <-f filename | ipName {ipName}>
```

There, filename includes all remote IP names. Or you can enumerate them as arguments.

Example

sshUWPlace.sh 12345 uw1-320-00 uw1-320-01 uw1-320-02 uw1-320-03

sshUWPlace.sh 12345 zmac100 zmac101 zmac102 zmac103

sshUWPlace.sh –f nodes

where the nodes file includes uw1-320-00 uw1-320-01 uw1-320-02 uw1-320-03 in a separate line.

Use runUWPlace.sh for computing nodes that do not run bash or do not see ATeam as the same path, (e.g., Windows clients or Linux/Mac not connected to the same NFS). Log in each of those computers and type as follows:

runUWPlace.sh for bash:

```
runUWPlace.sh <port#>
```

runUWPlace.bat for DOS prompt:

```
runUWPlace.bat <port#>
```

Example

```
runUWPlace.sh 12345
```

UWPlace daemons can be shut down with one of the following two actions. If you have invoked the daemons with sshUWPlace.sh at once, use sshKill.sh:

sshKill.sh:

```
sshKill.sh <-f filename | ipName {ipName}>
```

There, filename includes all remote IP names. Or you can enumerate them as arguments.

Example

```
sshKill.sh 12345 zmac100 zmac101 zmac102 zmac103

sshKill.sh –f nodes

where the nodes file includes zmac100 zmac101 zmac102 zmac103 in a separate line.
```

If you have started up each daemon with runUWPlace.sh, visit each terminal window where UWPlace is running, and type control-c to terminate the current runUWPlace.sh or runUWPlace.bat.

2.2 Inter-Domain UWPlace Invocation and Shutdown

This invocation assumes that only ssh port 22 is available to establish a socket across IP domains. In that case, your local and remote computers may need to use an SSH tunnel to establish inter-UWPlace communication.

Establish an SSH tunnel from your current system to the target remote computer using the following command:

SSH Tunnel Command:

```
ssh –l <accountName> -L <localOutPort#>:localhost:<targetInPort#> -R
    <targetOutPort#>:localhost:<localInPort#> <targetHostIP>
```

Example, SSH Tunnel Command:

```
ssh –l mickey -L 1000:localhost:12345 -R   2500:localhost:12345
    uw1-320-00.uwb.edu

This allows the local computer to establish a socket to mickey's at
uw1-320-00.uwb.edu so that a local process can write and read data to
port 1000 and from port 123455, whereas a remote process at uw1-320-20
can read and write data from port 12345 and to 2500.
```

Once an SSH tunnel is established, run the UWPlace command via the SHH tunnel on the remote computer (repeat this process as necessary for multiple remote systems):

UWPlace Command for bash:

```
java –cp $ATeam/jars/UWAgent.jar:. UWAgent.UWPlace –p <localInPort#>
    -<localOutPort#> <targetHostIP>
```

UWPlace Command for DOS prompt:

```
java –cp %ATeam%\jars\UWAgent.jar;. UWAgent.UWPlace –p <localInPort#>
    -<localOutPort#> <targetHostIP>
```

Example:

```
java –cp $ATeam/jars/UWAgent.jar:. UWAgent.UWPlace –p 12345 -1000 uw1-
320-00.uwb.edu
```

To shutdown daemons, visit each terminal window where UWPlace is running, and type control-c to terminate the current runUWPlace.sh or runUWPlace.bat

2.3 UWPlace Execution Test

To check if a UWPlace daemon is running at remote nodes as well as your local site, inject a sample agent as follows:

HopSkipJump.java

```
cd $ATeam/SampleAgents
java –cp $ATeam/jars/UWAgent.jar UWAgent.UWPlace –p port# -m 4
localhost HopSkipJump ipAddr0 ipAddr1 ipAddr2 ipAddr3 … ipAddrN
```

This agent visits each IP address in the same order as specified and prints out a greeting message from one to another place. Make sure that the your local computer is running UWPlae and that SampleAgent's last argument, (i.e., ipAddrN) is your local IP address. Do not use localhost for ipAddrN.

3. Job Injection

3.1 Compilation of Applications

You need to compile your Java application before submitting it to AgentTeamwork. Unless you have recompiled the AgentTeamwork system with Java1.6, make sure that you are going to compile your programs with Java1.5.

Compile your Java programs with two AgentTeamwork's jar files such as MPJ.jar and Ateam.jar:

Compile your program on bash:

```
javac –cp $ATeam/jars/MPJ.jar:$Ateam/jars/Ateam.jar:. *.java
```

Compile your program on DOS prompt:

```
javac –cp %ATeam%\jars\MPJ.jar;%Ateam%jars%Ateam.jar:. *.java
```

Thereafter, create .jar files for the user applications in the working directory by running the following command.

Create .jar files on both bash and DOS prompt:

```
jar cvf <name>.jar *.class
```

When program files are modified or to add new class files to the existing .jar file, run the following command to append the necessary changes to the existing .jar.

Create .jar files

```
jar uvf <name>.jar *.class
```

A sample java application was made available at $ATeam/applications/Sample/ for your convenience. This program repeats writing a greeting message 10 times to each remote node's terminal or /tmp log file as well as to AgentTeamwork's GUI window.

Sample.java

```
mport AgentTeamwork.Ateam.*; // AgentTeamwork
import MPJ.*;                // mpijava
import java.net.*;           // for InetAddress

public class Sample extends AteamProg {
    private int cycle = 0;

    // blank const for Ateam
    public Sample( Ateam o ) { }

    public Sample( ) { }

    public void compute( ) {
        for ( int i = 0; i < 10; i++ )  {
            try {
                // write to each node-local terminal or /tmp's log file
                System.out.println( "Rank[" + MPJ.COMM_WORLD.Rank( ) +
                                    "]: Hello! @ " +
                                    InetAddress.getLocalHost( ).
                                    getHostName( ) );
                // write to SubmitGUI
                AteamProg.ateam.gridfile.
                    writeStdout( "Rank[" + MPJ.COMM_WORLD.Rank( ) +
                                 "]: Hello! @ " +
                                 InetAddress.getLocalHost( ).
                                 getHostName( ) + "\n" );
                Thread.currentThread( ).sleep( 1000 );
                ateam.takeSnapshot( cycle );
            } catch( Exception e ) {
                e.printStackTrace( );
            }
        }
    }

   /**
     * Sample.java is a simple program in that each rank prints out its own
     * greeting message 10 times.
     */
    public static void main( String[] args ) throws Exception
    {
```

```
        // Start the MPI library.
        MPJ.Init(args, ateam);

        // program instantiation or retrieval
        Sample program = null;
        if ( ateam.isResumed( ) ) {
            System.out.println( "isResumed" );
            program = ( Sample )ateam.retrieveLocalVar( "program" );
            program.cycle++;
        }
        else { // !ateam.isResumed( )
            System.out.println("!isResumed");

            // Compute a Sample object in both master and slaves.
            program = new Sample( );
            ateam.registerLocalVar( "program", program );

            System.out.println( "takeSnapshot main" );
            ateam.takeSnapshot( 0 );
        }

        // start the computation
        program.compute( );

        // Terminate the MPI library.
        MPJ.Finalize( );
    }
}
```

Its compilation should be done with the following scripts:

Compile Sample.java and create its jar on bash:

```
cd $ATeam/applications/Sample
javac –cp $ATeam/jars/MPJ.jar:$ATeam/jars/Ateam.jar:. Sample.java
jar cvf Sample.jar Sample.class
```

Compile Sample.java and create its jar on DOS prompt:

```
Cd %ATeam%\applications\Sample
javac –cp %ATeam%\jars\MPJ.jar:$ATeam\jars\Ateam.jar:. Sample.java
jar cvf Sample.jar Sample.class
```

3.2 Job Injection from submitGUI

If it is your very first time to submit a job to AgentTeamwork through its GUI, you have to set up your ".java.policy" file so as to run the GUI as an applet correctly. Cut and past the following policy example, and save it as the ".java.policy" file under your home directory, (i.e., ~/.java.policy on bash and C:\Documents and Settings\%USERNAME%\.java.policy on Windows).

Example, .java.policy on Mac OS X:

```
/* Replace file:///Volumes/Users/Users/mickey with your home directory */
/* For example on Windows: file:/C:/Documents and Settings/mickey      */
/* For example on Linux: file:///home/mickey                           */

grant codeBase "file:///Volumes/Users/Users/mickey/agentteamwork/GUI/-" {
```

```
  permission java.io.FilePermission "<<ALL FILES>>", "read, write, delete,
execute";
  permission java.util.PropertyPermission "user.home", "read";
  permission java.util.PropertyPermission "user.dir", "read";
  permission java.util.PropertyPermission "file.encoding", "read";
  permission java.lang.RuntimePermission "modifyThread";
  permission java.lang.RuntimePermission "modifyThreadGroup";
  permission java.net.SocketPermission "*", "accept";
};
grant codeBase "file:///Volumes/Users/Users/mickey/agentteamwork/jars/-" {
  permission java.io.FilePermission "<<ALL FILES>>", "read, write, delete,
execute";
  permission java.util.PropertyPermission "user.home", "read";
  permission java.util.PropertyPermission "user.dir", "read";
  permission java.util.PropertyPermission "file.encoding", "read";
  permission java.lang.RuntimePermission "modifyThread";
  permission java.lang.RuntimePermission "modifyThreadGroup";
};
grant codeBase "file:///Volumes/Users/Users/mickey/agentteamwork/jars/-" {
  permission java.net.SocketPermission "127.0.0.1", "connect";
};
grant codeBase "file:///Volumes/Users/Users/mickey/files/-" {
  permission java.io.FilePermission "<<ALL FILES>>", "read, write, delete,
execute";
};
grant codeBase "file:///Volumes/Users/Users/mickey/-" {
  permission java.io.FilePermission "<<ALL FILES>>", "read";
};
```

Now, run launchGUI.sh that internally starts AgentTeamwork's job-submission GUI:

Start launchGUI.sh

```
launchGUI.sh
```

Follows the GUI's menus as shown below:

(1) **Page 1:** enable/disable the resource agent.

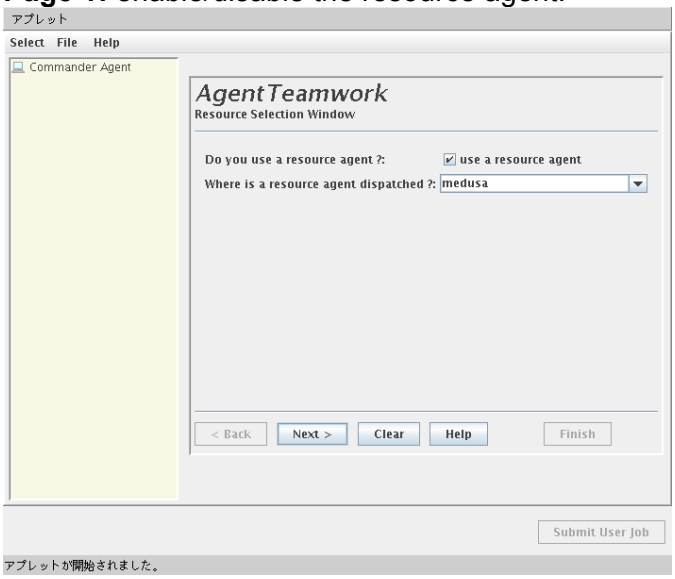

Figure 0. Resource agent selection window

If you have invoked AgentTeamwork's XML resource database and use a resource agent, check the "use a resource agent" check box and specify the IP address where the XML database is running. If you don't use the resource database, don't check the box to skip to **page 4**.

(2) **Page 2:** decide resource requirements such as OS, CPU, disk, memory, etc.

If you have checked the "use a resource agent" check box on page 1, decide OS type, CPU architecture, disk space, memory space, #CPU cores per each node, percentage of CPU idle, a choice of cluster computers or individual public computers, CPU speed, total #computing nodes, when to run the user job, (e.g., 855pm34sec=205534, now=0) and extra nodes for resumption purposes (RN). If necessary, you may specify IP addresses of computing node you would like to use.

The minimum requirement is total #computing nodes (denoted as "total").

Recommendations:

- Fill only this "total" field if you would like to maximize the opportunity of finding the best computing resources from an XML resource database.
- Choose "public" in the "choice" field if you use computing nodes within the same IP domain as you are working.

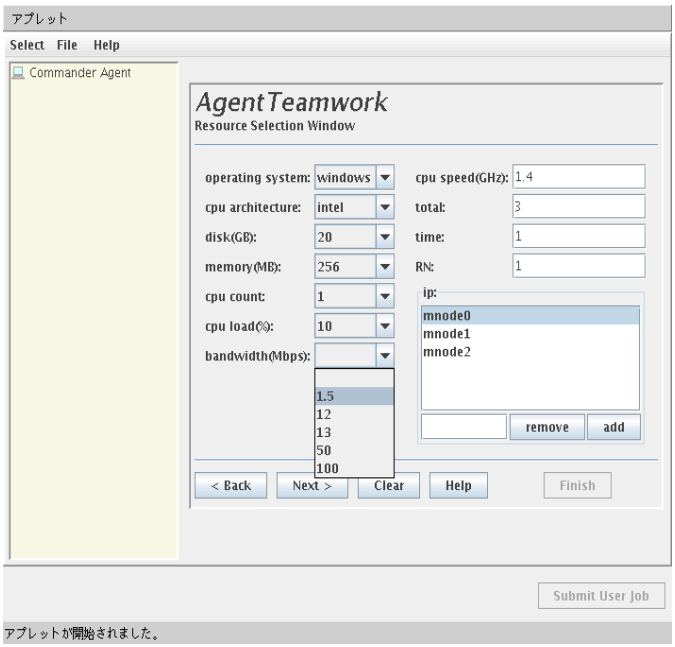

Figure 0. Specification selection window

(3) **Page 3:** specify an FTP server.

If you have filled out resource requirements on page 2, you will come to this page to specify an FTP server that your resource agent will contact.

Users at UW Bothell:

FTP server	ftp.tripod.com
Account	agentteamwork

Users at Keio SFC:

FTP server	ftp.tripod.com
Account	mdbl.sfc

Ask the corresponding password to us at dslab@u.washington.edu.

For time being, fill out -1 in the “Monitoring Period”.

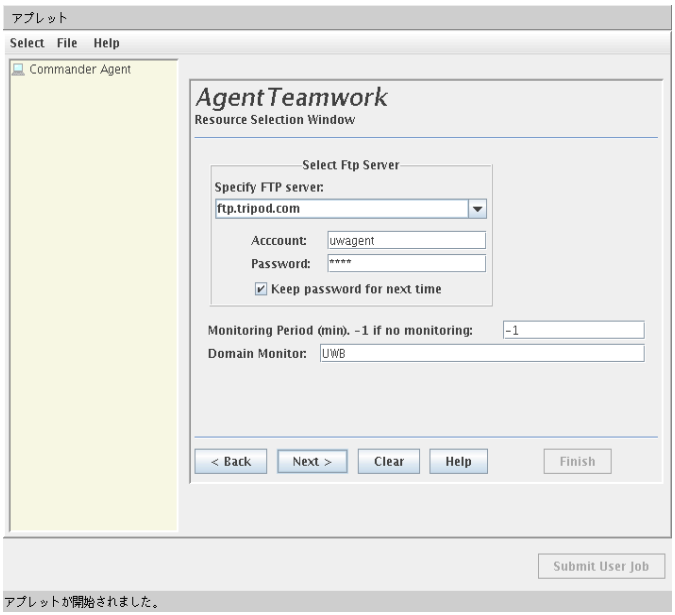

Figure 0. FTP Server selection window

(4) **Page 4:** describe your application.

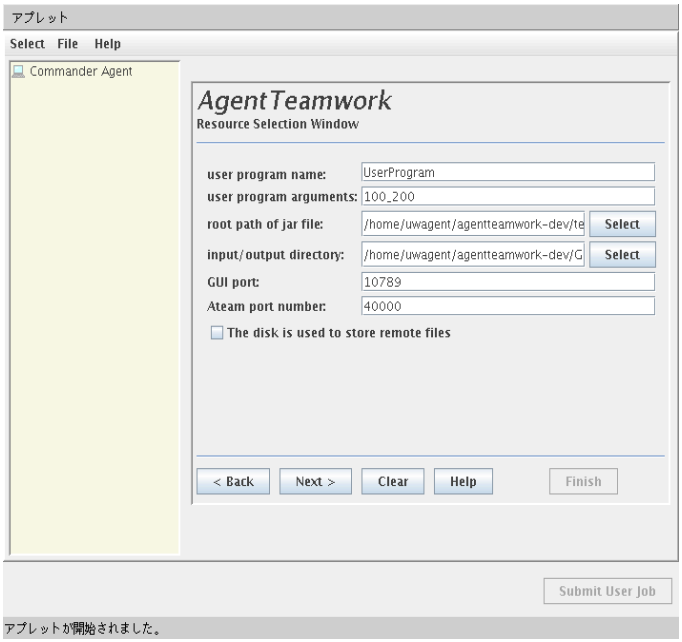

Figure 0. User program selection window

Specify your application program, arguments passed to it, the absolute path to its “jar” file, a directory to input/output files, the GUI port, and Ateam port. Note that arguments

must be delimited with "_" rather than a space " ". For Sample.java, fill out only the user program name: **Sample**

(5) **Page 5:** specify if a commander agent should be submitted remotely or locally.

If you click the "remote submission" box, specify where a commander agent should be submitted. You must also tell the GUI of your computer's IP address in the canonical form (rather than localhost).

(6) **Page 6:** describe about your bookkeeper agents.

If you use an XML resource database, you may specify the number of bookkeeper agents, (each of which will be dispatched to a different computing node automatically). If you do not use the database (thus skipping pages 2 and 3), specify computing nodes to accept a bookkeeper agent. Such a list of bookkeeper agents must be delimited with "_" rather than a comma or a space.

Click **"finish"** if you use the database, otherwise click **"next".**

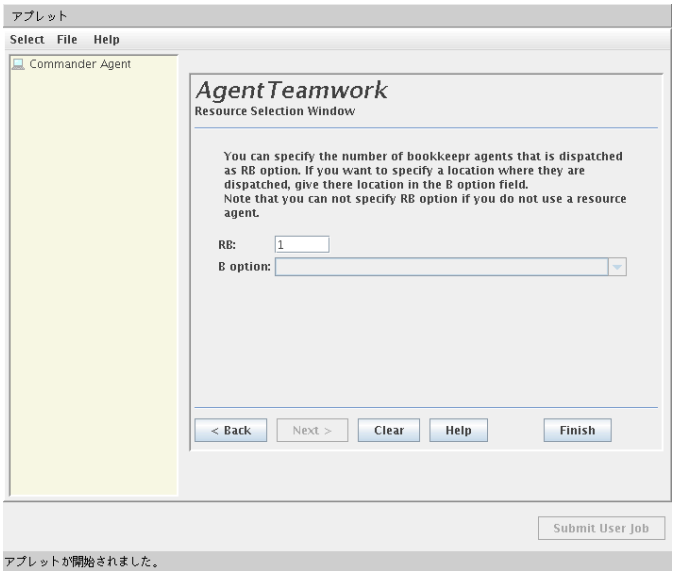

Figure 0. Bookkeeper agent selection window

(7) **Page 7:** specify the number of cluster systems to allocate to your job.

The user can input the number of clusters used for job execution. Optionally, the user may also specify the number of extra clusters for cluster resumption.

If you use computing nodes in the same domain as you are working, fill these fields with 0 and skip to **page 9.**

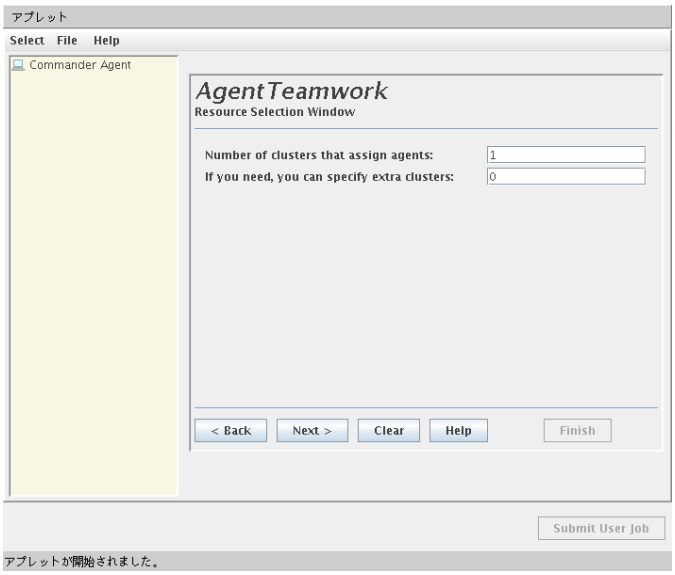

Figure 0. Number of clusters selection window

(8) **Page 8:** specify clusters to allocate to your job.

The cluster options include: cluster name, gateway IP name, and each computing node's IP name. If the user would like to provide additional nodes of a given cluster to assist with job resumption in case of node crashes, the cluster name and gateway IP with the additional nodes much match the cluster name and gateway IP of the given cluster.

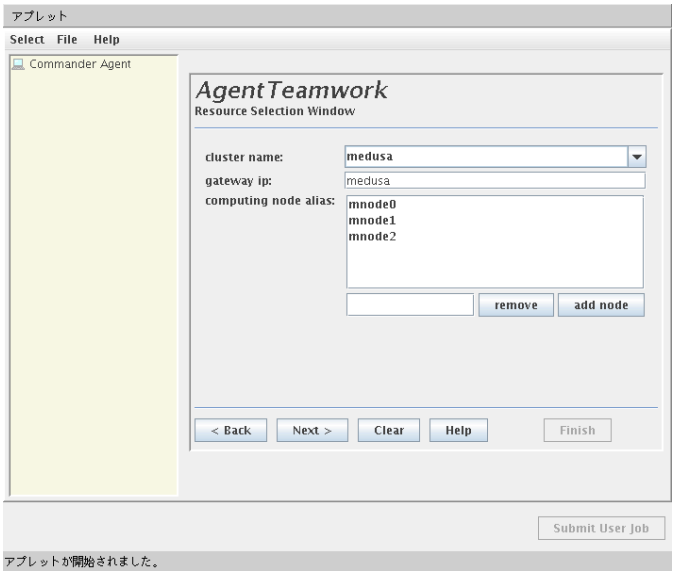

Figure 0. Cluster Selection Window

(9) **Page 9:** specify all individual computing nodes to allocate to sentinels, thus to your job.

The sentinel agent options include: root sentinel agent IP name, desktop computing node IP names, and optionally extra desktop computing node IP names for resumption purposes. At least two sentinel agent IP names are required. (The first is for the root sentinel and the second for the rank-0 sentinel.) Use '_' to delaminate between IP names.

After filling out those fields, click the "finish" button.

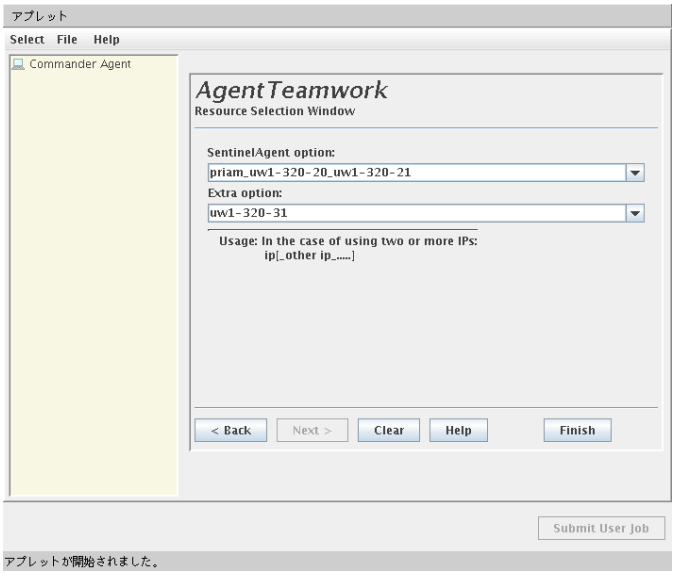

Figure 0. Root Sentinel agent and desktop computing node selection window

3.3 File Transfer with SubmitGUI

SubmitGUI has file transfer function by which input files are transferred automatically to the Commander agent after job submission. Similarly, output files that are created by each computing agent are received by the Commander agent and then written to the user-specified input/output directory.

4. Job Monitoring and Termination

4.1 Job monitoring and Standard In/Output

After selecting resources, the user inputs the port number for UWPlace as follows:

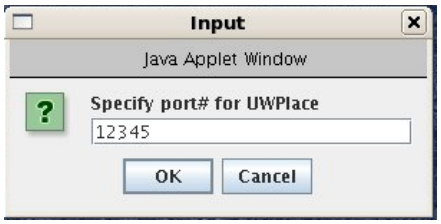

Figure 1. Port input window

Then, a job monitoring display is shown (10).

As shown Figure 10, SubmitGUI's job monitoring display is constructed from (1) agent selection window, (2) standard output display, (3) standard input field, and (4) remote node information display.

(1) Agent selection window

The agent selection window allows the user to see the standard output from an agent, send standard input to an agent, and monitor the status of all agents. All gateway agents, public computing nodes, and extra public computing nodes can be found under the root Sentinel agent.

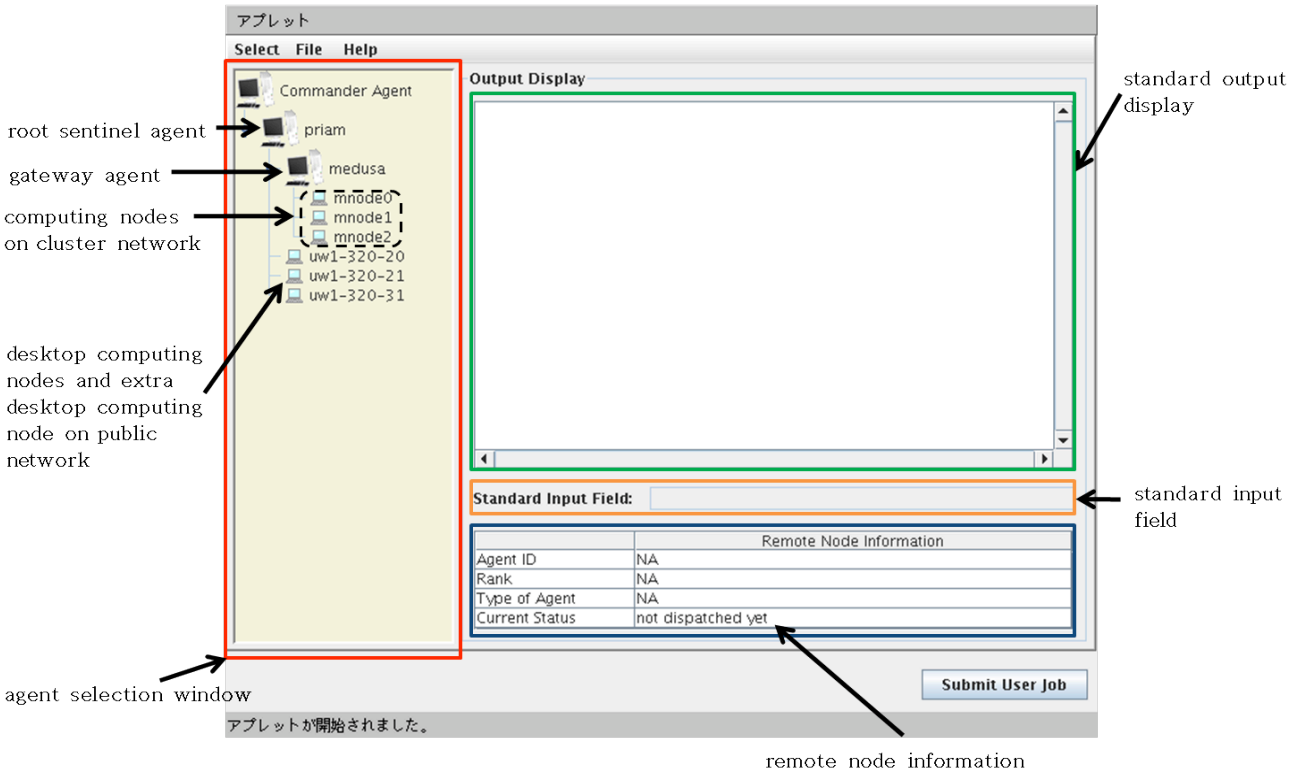

Figure 10. Job monitoring display

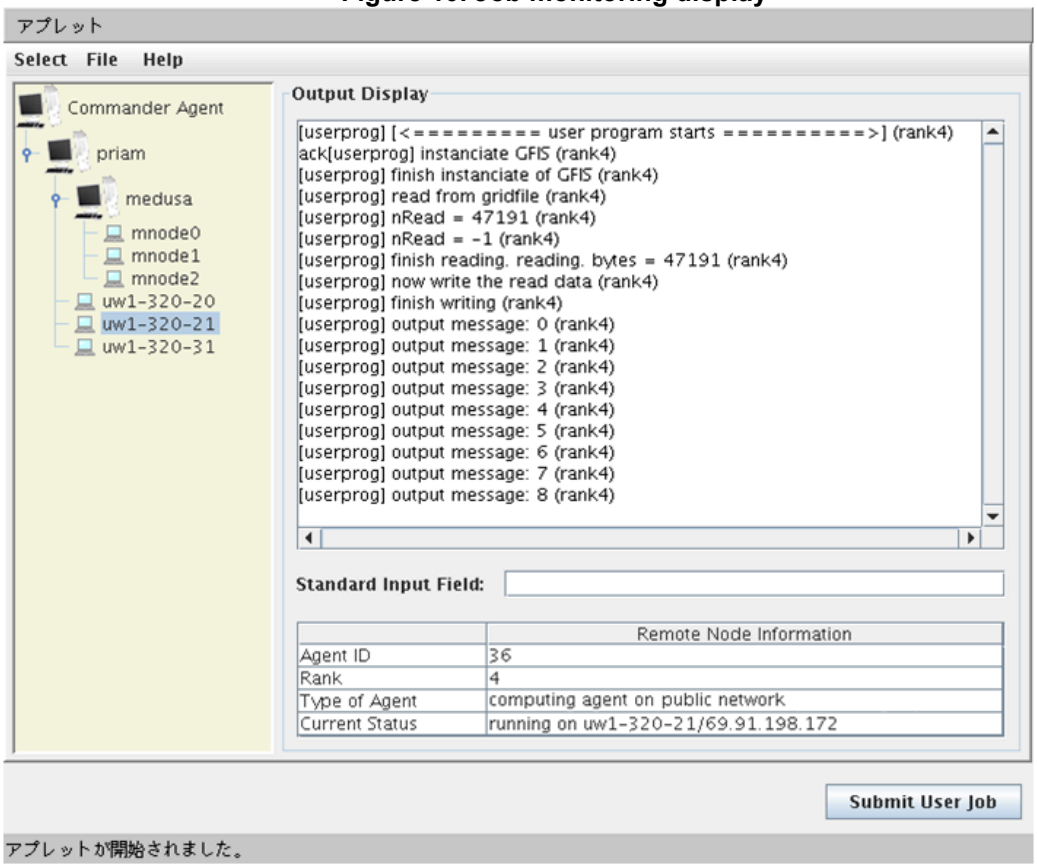

Figure 11. Standard output display

(2) Standard output display

Since standard output is sent only from computing nodes, the output display of the Commander agent and gateway agents show no output.

(3) Standard input field

Standard input is sent to a corresponding agent immediately when the user types data to a standard input field.

(4) Remote node information display

Remote node information display shows the dispatched agent id, agent rank, type of agent, and current status. The type of agent that is shown in remote node information display is “Commander agent”, “Root Sentinel agent”, “Gateway agent”, “Computing agent on cluster network”, and “Computing agent on public network”. If these agents are extra agents intended for resumption, “extra” is added to the type information as a prefix.

Current status varies with an agent status. When each agent has not been dispatched to remote node, the status is “not dispatched yet” as shown in Figure . Agent id and rank are calculated prior to dispatch unless the agent type includes the prefix “extra”, in which case agent id, rank and status information is shown as in Figure . When an agent does crash the “extra” agent will take over the id, rank (and ultimately status) of the crashed agent.

	Remote Node Information
Agent ID	36
Rank	4
Type of Agent	computing agent on public network
Current Status	not dispatched yet

Figure 12. Remote node information in computing agent (public network)

	Remote Node Information
Agent ID	NA
Rank	NA
Type of Agent	extra computing agent on public network
Current Status	not dispatched yet

Figure 13. Remote node information in extra computing agent (public network)

If a certain agent crashed, its agent’s job monitoring display is shown (as in Figure 14) and the crashed agent hops to extra computing node. The user can discover where the agent moved by checking job monitoring display of extra computing nodes. The status of resumed agent is shown in Figure 15.

(4) Automatic Job Abortion

A job may be aborted automatically when AgentTeamwork can no longer continue to run it because of the following reasons:

(a) You have specified wrong computing nodes for a job injection. In most case, nodes specified for you're a job may not have yet started UWPlace. Make sure that UWPlace is running at each of the nodes specified.

(b) Your application caused an exception. Debug your program.

(c) When detecting a node crash, AgentTeamwork was not able to find the latest snapshot corresponding to this node. Your application did not include checkpoints at all or a node was crashed before your application took the very first snapshot.

(d) When detecting a node crash, AgentTeamwork was not able to find another node to resume the application. Check the following cases: you didn't use a resource database and didn't specify extra nodes for resumption purposes; all extra nodes you specified were used up; the resource database couldn't available computing nodes; and the database itself was crashed.

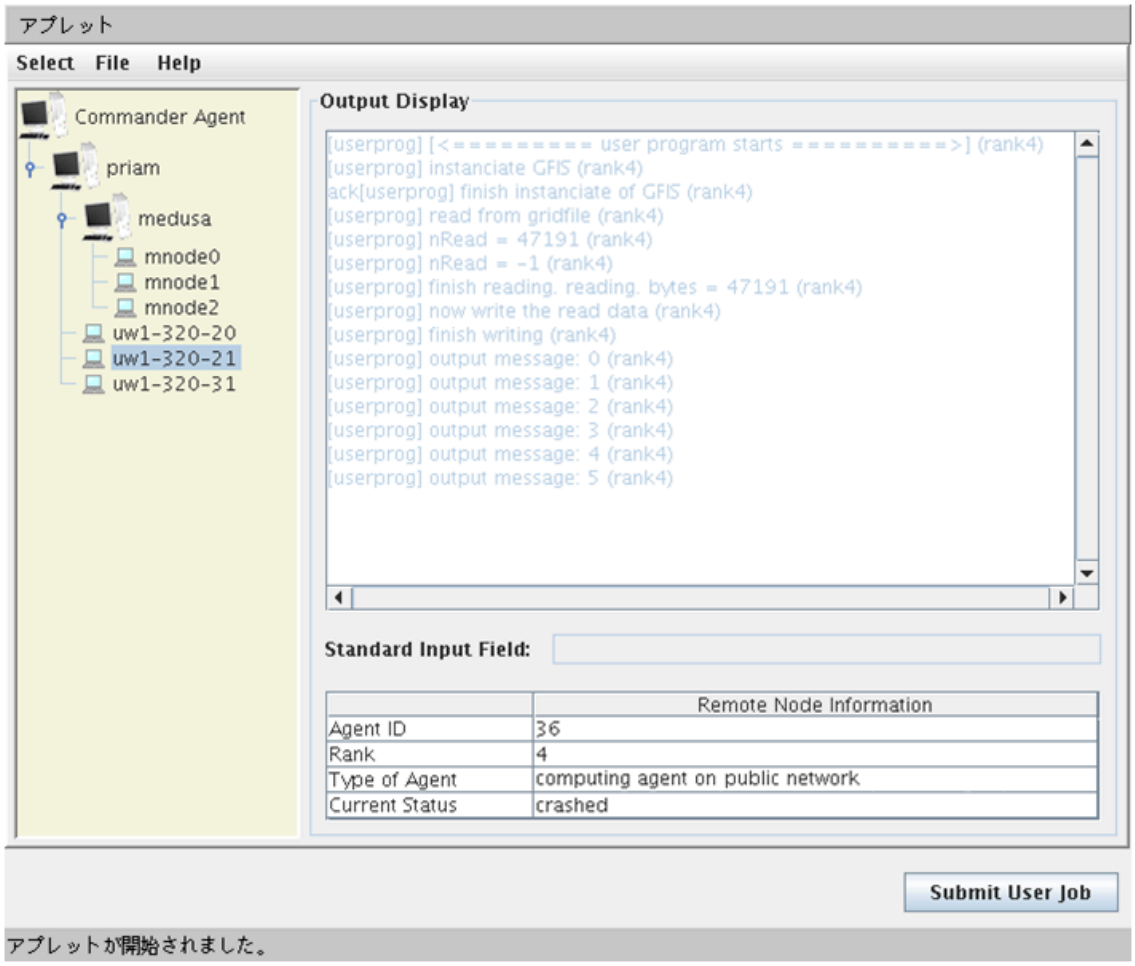

Figure 14. Job monitoring display of crashed agent

	Remote Node Information
Agent ID	36
Rank	4
Type of Agent	computing agent on public network
Current Status	resuming on uw1-320-31/69.91.198.182

Figure 15. Resumed agent information

4.3 Job Termination

You may terminate the current job anytime after submitting it. Click the “Abort User Job” button, located at the bottom-left corner of the submitGUI window, (see Figure 16.)

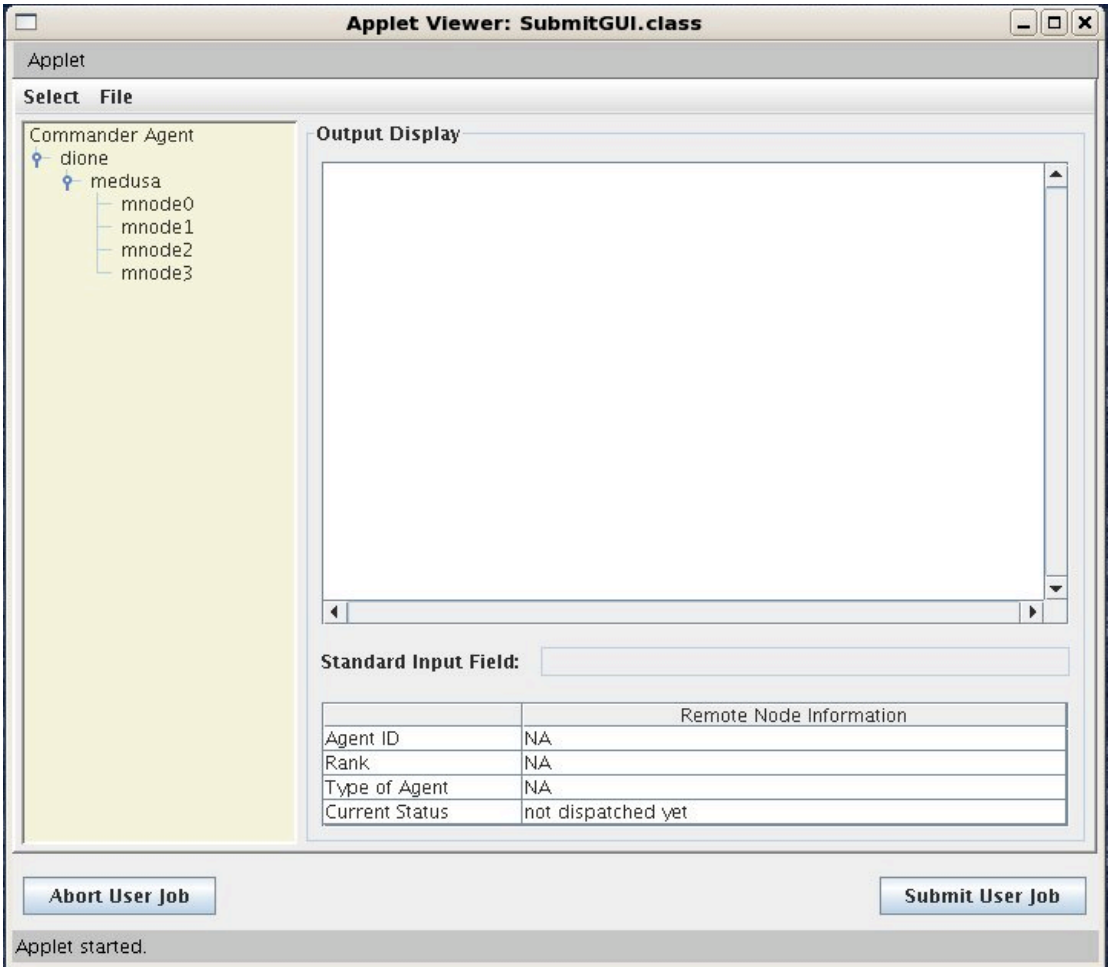

Figure 16. Aborting the current user job

4.4 Monitoring and Terminating UWPlace

Although you should use SubmitGUI to control your agents engaged in your current job, you may also want to manage remote UWPlace daemons, (i.e., the underlying agent-execution platforms) so as to monitor their status, to terminate if they got stuck, and to restart them. The following shell scripts and batch files are available for these purposes.

UWMonitor.sh / UWMonitor.bat:

```
sshUWPlace.sh <port#> <as | kill agentId | suspend agentId | resume
agentId | help>
```

This script/batch file shows the status of agents or controls a given agent on the local UWPlace daemon:

Options	Remarks
as	Shows the status of all local agents
kill agentId	Terminates a given agent. An agent Id is obtained with “as”

suspend agentId	Suspends the thread that is executing a given agent.
resume agentId	Resumes the thread that is executing a given agent.
help	Lists all the available options of the UWMonitor.sh script.

sshUWMonitorAs.sh / sshUWMonitorAs.bat:

```
sshUWPlace.sh <port#> <-f filename | ipName {ipName}>
```

This script/batch file shows the status of all agents running on each of remote computers that are listed in filename or in forms of IP names. More specifically, it runs "UWMonitor port# as" at each of these remote computers.

sshTail.sh / sshTail.bat:

```
sshUWPlace.sh <port#> <-f filename | ipName {ipName}>
```

If a UWPlace daemon is launched remotely from sshUWPlace.sh, its log messages are all saved in the /tmp/yourAccount_uwplace.log file. The script/batch file displays the last 10 lines of this log file. If a UWPlace daemon stays idle, the log shows:

#agentList.size = 0

If the list size is larger than 0, the daemon has some agents regardless of their active or inactive status.

sshTrunc.sh / sshTrunc.bat:

```
sshUWPlace.sh <port#> <-f filename | ipName {ipName}>
```

As explained in sshTail.sh/sshTail.bat, a remote UWPlace daemon keeps saving its log messages in the /tmp/yourAccount_uwplace.log file. To prevent the file from growing unacceptably, it is recommended that you should sometimes run the sshTrunc.sh/sshTrunc.bat script to truncate the log file.

kill.sh:

```
sshUWPlace.sh <port#> <-f filename | ipName {ipName}>
```

This shell script shuts down the local UWPlace daemon running at a given IP port#. No corresponding DOS batch file.

sshKill.sh / sshKill.bat:

```
sshUWPlace.sh <port#> <-f filename | ipName {ipName}>
```

This script/batch file shuts down all the remote UWPlace daemons that are running at a given IP port# and listed in filename or in the form of IP names.

As described in Section 2, you can use the following two shell scripts or DOS batch files to restart a UWPlace daemon.

runUWPlace.sh / runUWPlace.bat:

```
runUWPlace.sh <port#>
```

This script/batch file starts a UWPlace daemon locally at a given IP port#.

sshUWPlace.sh / sshUWPlace.bat:

```
sshUWPlace.sh <port#> <-f filename | ipName {ipName}>
```

This script/batch file starts a UWPlace daemon at a given IP port# on all computing nodes that have been listed in filename or in form of IP names.

5. XML Resource Database

5.1 XML resource definition

To use AgentTeamwork's resource database, an XML Resource definition should be written for every computing node and uploaded to an FTP server accessible by all AgentTeamwork daemons.

There are two types of XML resource definitions, one for public nodes and one for clusters. An example of each appears below. The name of the resource definition files must be <cur_ip_name>.xml for public nodes and the name of definition files for clusters must be <cl-cluster_name>.xml

Public node XML resource definition: uw1-320-10.xml

```
<?xml version="1.0" ?>
<resource>
      <design_time>
            <domain>UWB</domain>
            <ip_name>uw1-320-10</ip_name>
            <ip_addr>69.91.198.161</ip_addr>
            <human_owner>uwb</human_owner>
            <cpu_speed>2100</cpu_speed>
            <cpu_arch>intel</cpu_arch>
            <cpu_count>2</cpu_count>
            <memory>1024</memory>
            <os_type>linux</os_type>
            <disk_space>40</disk_space>
            <cpu_load>100</cpu_load>
            <availability multiple="true">
                  <time>0000-1159</time>
                  <time>1200-2359</time>
            </availability>
            <time_zone>pacific</time_zone>
            <inter_net_device>ethernet</inter_net_device>
            <intra_net_device>ehternet</intra_net_device>
            <libraries multiple="true">
                  <name>cexec</name>
                  <name>mpirun</name>
            </libraries>
            <inter_net_band>100</inter_net_band>
            <intra_net_band>100</intra_net_band>
      </design_time>
</resource>
```

Cluster XML resource definition: cl-medusa-8-15.xml

```
<?xml version="1.0" ?>
<cluster>
      <design_time>
            <domain>UWB</domain>
            <name>cl-medusa-8-15</name>
            <gateway>medusa.uwb.edu</gateway>
            <alias>medusa</alias>
            <group>
                  <ip_list>
                        <ip_name>mnode8</ip_name>
```

```
                        <ip_name>mnode9</ip_name>
                        <ip_name>mnode10</ip_name>
                        <ip_name>mnode11</ip_name>
                        <ip_name>mnode12</ip_name>
                        <ip_name>mnode13</ip_name>
                        <ip_name>mnode14</ip_name>
                        <ip_name>mnode15</ip_name>
                  </ip_list>
                  <human_owner>uwb</human_owner>
                  <cpu_speed>3200</cpu_speed>
                  <cpu_arch>intel</cpu_arch>
                  <cpu_count>1</cpu_count>
                  <memory>512</memory>
                  <os_type>linux</os_type>
                  <disk_space>30</disk_space>
                  <cpu_load>100</cpu_load>
                  <availability multiple="true">
                        <time>0000-1159</time>
                        <time>1200-2359</time>
                  </availability>
                  <time_zone>pacific</time_zone>
                  <inter_net_device>ethernet</inter_net_device>
                  <intra_net_device>gigaether</intra_net_device>
                  <libraries multiple="true">
                        <lib_name>java</lib_name>
                        <lib_name>prunjava</lib_name>
                  </libraries>
                  <inter_net_band>100</inter_net_band>
                  <intra_net_band>1000</intra_net_band>
            </group>
      </design_time>
</cluster>
```

For running AgentTeamwork in the UWB domain, connect to ftp.tripod.com, username: agentTeamwork, and email dslab@u.washington.edu for the password. At present, the FTP server registers the following xmls for use within UW Bothell, and thus you have no need to upload or modify xml files:

Public/ directory:

```
mnode0.xml, mnode1.xml, mnode2.xml, mnode3.xml, mnode4.xml,
mnode5.xml, mnode6.xml, mnode7.xml

uw1-320-00.xml, uw1-320-01.xml, uw1-320-02.xml, uw1-320-03.xml,
uw1-320-04.xml, uw1-320-05.xml, uw1-320-06.xml, uw1-320-07.xml

perseus.xml, tarvos.xml
```

Cluster/ directory:

```
cl-medusa-8-15.xml, cl-uw1-320-08-15.xml
```

For outside domains, establish a public FTP which AgentTeamwork can access so as to maintain its XML database in that FTP. The FTP account must have two directories named “Cluster” and “Public”.

Upload all public node XML resource definitions to the “Public directory” and all cluster definitions to the “Cluster” directory.

5.2 Database Set-up

Before starting up the Database, ensure that port 8000 is free. AgentTeamwork is hard-coded to this port for this purpose, so port 8000 must be free for the database before you can use AgentTeamwork properly.

To start up the database properly run startDB.sh or startDB.bat, reproduced below.

startDB.sh or startDB.bat

```
#!/bin/sh

cd ../xmls/
java -cp ../jars/Agents.jar AgentTeamwork.Agents.XDBase 8000
```

Shut down the database by running shutdownDB.sh or shutdownDB.bat reproduced below. Before a shutdown, the database saves any resource changes in the "$Ateam/xmls/" directory, which will be reused upon a next invocation. You can also kill the database process with CTRL+C, however no changes will be recorded into the xmls directory.

shutdownDB.sh or startDB.bat

```
#!/bin/sh

echo "shutdownDB: shutdown the database"

java -cp ../jars/Agents.jar AgentTeamwork.Agents.ShutdownDB
```

Ensure that these scripts run where the resource agent is dispatched, which is specified with the commander agent's R option.

6. Command Line Operations

6.1 Job Injection from Command line

Log in a computer that has installed AgentTeamwork. From the command line, you must run FileThread.sh that takes care of file transfer between the front end and a user application:

runFileThread.sh or runFileThread.bat

```
runFileThread.sh port# directory
```

There *port#* must be the same as the GP option given to a commander agent, (see table 3), and *directory* is where your input/output files are stored. If you intend to inject a job from the same terminal window where runFileThread.sh was invoked, you must run runFileThread.sh with the "&" delimiter.

Thereafter, you can inject a commander agent to dispatch your application, using UWInject. The command format is:

Inject a job

```
java -Xmx512M -cp linkToUWAgent.jar:linkToAgents.jar:linkToGUIUtil.jar
UWAgent.UWInject optionsforUWInject destination
AgentTeamwork.Agents.CommanderAgent optionsforCommander
```

Table 1: Parameters for Command Line Input

Parameters of the command line input	Remarks	**Example** values
linkToUWAgent.jar	A symbolic link or a pass to UWAgent.jar	jars/UWAgent.jar (assuming that the current working directory is agentteamwork.)
linkToAgents.jar	A symbolic link or a pass to Agent.jar	jars/Agents.jar (assuming that the current working directory is agentteamwork.)
linkToGUIUtil.jar	A symbolic link or a pass to GUIUtil.jar	jars/GUIUtil.jar (assuming that the current working directory is agentteamwork.)
optionsforUWInject	Options required/recommended for UWInject	See table 2
destination	The IP where a commander agent is injected	localhost (Injected locally)
optionsforCommander	Options required/recommended for a commander agent	See table 3

Table 2: Options for UWInject

Options	Remarks	Values **required**
-p	The IP port through which you would like to inject a commander agent	-p 12345 (should be between 5001 and 65535.)
-m	The maximum number of children each agent can spawn. Mandatory.	-m 4 (The value must be **always 4**.)
-u	The directory where agents' code is located. **Mandatory.**	-u /home/uwagent/agentteamwork/AgentTeamwork/Agents (This option is always **fixed**.)
-j	Jar files an agent should carry with it. Must be delimited with a comma. **Mandatory.**	-j jars/GUIUtil.jar,jars/Agents.jar,jars/Ateam.jar,jars/MPJ.jar,jars/commons-net-1.4.1.jar,jars/jakarta-oro-2.0.8.jar,applications/applications.jar (This option is always **fixed**.)

Table 3: Options for Commander Agent

Options	Remarks	**Example** values
AP	An IP port number used for the underling GridTcp communication. **Mandatory**.	AP_11112 (Any number between 5001 and 65535)
GP	An IP port number with which the commander agent contacts SubmitGUI or FileThread. **Mandatory**.	GP_11111 (Any number between 5001 and 65535, but different from AP)
GI	An IP address with which the commander agent contacts SubmitGUI or FileThread. If omitted, GI is set to "localhost".	GI_medusa (Any IP name or address)
S	A list of ip names to dispatch a sentinel agent. If it is not given, a resource agent is responsible to provide the commander with such a list. The 1st IP name is dedicated to the root sentinel with id 2 that does not participate in actual computation.	S_priam_uw1-320-00_uw1-320-01_uw1-320-02_uw1-320-03 (The root sentinel reins all others at priam. The other nodes including uw1-320-00, uw1-320-01, uw1-320-02, and uw1-320-03 will participate in computation.)
CL	A remote cluster to use as well. The cluster gateway name comes first, followed by the cluster gateway node alias, followed by a list of machine nodes within that cluster. If it is not given with the S option, a resource agent is responsible to provide the commander with such a list. NOTE: To specify extra-compute-nodes within this cluster, supply an ECL parameter (see below) using the same cluster name as this CL parameter.	CL_medusa.uwb.edu_medusa_mnode0_mnode1_mndoe2_mnode3 (medusa.uwb.edu is a cluster gateway whose cluster-internal alias is medusa. It reins four cluster-internal nodes including mnode0, mnode1, mnode2, and mnode3.)
E	A list of extra IP names to resume an agent when one is crashed. If a resource agent is invoked due to missing of the S and/or B option(s), it will provide the commander with the E option.	E_uw1-320-04_uw1-320-05 (If a sentinel agent is crashed, it is resumed at uw1-320-04. If one more crash occurs, the next choice will be uw1-320-05.)
ECL	An extra remote cluster to use. The cluster gateway name comes first, followed by the cluster gateway node alias, followed by a list of machine nodes within that cluster. If the same cluster gateway is specified in the CL option, this ECL means a list of additional computing nodes in the same cluster for recovery purposes.	ECL_medusa.uwb.edu_medusa_mnode4_mnode5 (When the medusa.wub.edu cluster detects any node crash, it will resume computation on mnode4 and thereafter mnode5.)

B	A list of IP names to dispatch a bookkeeper agent. If it is not given, a resource agent is responsible to provide the commander with such a list.	B_tarvos_phoebe (Two bookkeeper agents are dispatched to tarvos and phoebe respectively.)
U	A user program name and its arguments. This is a **mandatory** option.	U_Mandelbrot_-2.0_1.0_0.0_200000_GRAD-BLUE (Mandelbrot is a Java user program and the rest are its arguments.)
R	An IP name to dispatch a resource agent to. If it is not given, a resource agent is launched at the same computing node as the commander is working. No more than one resource agent should be invoked.	R_dione (A resource agent is dispatched to dione.)
RA	A list of arguments passed to a resource agent. It is mandatory if a resource agent is invoked, (due to missing of the S and B options.) 1st argument: a shared ftp name 2nd argument: the ftp account 3rd argument: the ftp password 4th argument: the resource probing frequency in minutes. (default: 5) 5th argument: the resource domain Note: To keep a resource agent from spawning sensor agents, the 4th argument must be -1.	RA_ftp.tripod.com_agentteamwork_********_-1_UWB (A resource agent will contact ftp.tripod.com with the account: agentteamwork and the password ******** and choose resources in the UWB domain. It won't spawn sensor agents due to -1 as the 4th argument.)
RB	If the B option is not specified, the number of bookkeeper agents to spawn must be given with RB.	RB_2 (2 agents dispatched to a different node)
RQ	A list of query option/parameter pairs. **ip**: directly specify where to run an application. This corresponds to the S option. **cpuspeed**: the CPU speed in MHz **cpuarch**: the CPU architecture such as intel, 68K, PowerPC, and SPARC **cpucount**: # cpus in each computing node **memory**: per-node memory size in Mbytes **disk**: per-node disk size in Gbytes	RQ_cpuarch_linux_total_2 Two computing nodes under the Linux control are required for an application.

	total: # computing nodes required for a given user application (**mandatory if RQ is specified.**) **time**: when to run an applicatioin 855pm34se = 205534 or now = 0 **os**: operating system type such as linux, windows, and solaris **cpuload**: percentage of cpu idle state (in general 100%) **bandwidth**: the network bandwidth required in Mbps **choice**: choice of public, cluster, or a specific lcuster	
RN	A multiplier to determine how many backup computers should be requested from a resource agent. Value should be 1.0, 1.5, 2.0 or 3.0. Otherwise, 1.0 will be used as multiplier.	RN_2.0 The twice large number of computing nodes will be allocated to a user program.
RC	A list of classes that a resource agent will use.	XCollection_SensorAgent_DatabaseManagementService_Service_DeletionService_RetrievalService_QueryService_XPathUtil_StoreService_SensorAgent\$1_SensorAgent\$RemoteRscProbeTask_Ttcp_Ttcp\$Connect_Option_StopWatch (This option is always **fixed**.)

For running a job with AgentTeamwork repeatedly, it is highly recommended to create and run a shell script that includes the above parameters.

An example shell script

```
#!/bin/sh

cd $HOME/agentteamwork-dev/

java -Xmx512M -cp jars/UWAgent.jar:jars/Agents.jar:jars/GUIUtil.jar \
UWAgent.UWInject localhost AgentTeamwork.Agents.CommanderAgent \
-p 12345 \
-m 4 \
-u AgentTeamwork/Agents \
-j \
jars/GUIUtil.jar,jars/Agents.jar,jars/Ateam.jar,jars/MPJ.jar,jars/commons-
net-1.4.1.jar,jars/jakarta-oro-
2.0.8.jar,jars/xalan.jar,xercesImple.jar,applications/applications.jar \
U_Wave2DAteam_448_3000_200_show \
AP_11112 \
GP_11111 \
R_dione \
```

```
RQ_total_2 \
B_tarvos \
RA_ftp.tripod.com_agentteamwork_********_-1_UWB
RC_XCollection_SensorAgent_DatabaseManagementService_Service_DeletionServic
e_RetrievalService_QueryService_XPathUtil_StoreService_SensorAgent\$1_Senso
rAgent\$RemoteRscProbeTask_Ttcp_Ttcp\$Connect_Option_StopWatch
```

This example shell script injects a commander agent through IP port 12345, local to where a user currently logs in. The application is Wave2Ateam that takes 448, 3000, 200, and show as its arguments. The commander agent spawns a resource agent at dione, requesting two computing nodes for the application. It also spawns a bookkeeper agent at tarvos. The resource agent accesses ftp.tripod.com through the agentteamwork account and its password ********. No sensors will be generated.

6.2 Job Termination from Command line

After dispatching a job with a commander agent, runFileThread.sh/runFIleThread.bat shows the status of the job execution. If you want to terminate a job, simply type:

```
abort
```

Upon receiving a completion signal from the commander agent, runFileThread.sh/runFileThread.bat will return its control back to a new command line.

6.3 File Transfer with Command Line

If the user submits the job from a command line utility, input files are not transferred automatically. In this case, the user can transfer them by using FileThread which has similar functions to SubmitGUI. FileThread allows the user to transfer input files, receive standard outputs, and receive output files. FileThread options are specified when it is run, such as –p [port#], -d [directory name], -i, and –w.

- -p: is port number to communicate with Commander agent.
- -d: is directory input/output directory name to transfer and receive.
- -i: is specified if input files are in user side. Thus, files are transfer if this option is specified. If this option is not specified, directory information is sent to Commander, and then Commander agent itself reads the files.
- -w: is specified if input files are stored in remote computing node's /tmp directory

These options can *only* be used in the following combinations.

Accepted FileThread option combinations	
Specified option	Behavior
-p [prot#] –d [directory name] (-w) –i	Transfer input files to commander agent
-p [port#] –d [directory name] (-w)	Input files are not transferred. The files are read by commander agent, and then they are passed by commander agent.
-p [port#]	Just receives standard output

7. Trouble Shooting

Errors with Serializable

When using the snapshot feature of AgentTeamwork to save the state of execution of a user program ensure that all objects in the program to be saved are serializable. If they are not, the snapshot cannot be transmitted to a Bookkeeper and the checkpoint will fail.

Error java.lang.outofmemory

This error can occur if the user is running both the Commander and the Bookkeeper Agents on the same computer. In order to avoid or fix this error give the Commander and Bookkeeper Agents different nodes to execute on.

Constructor Error

User applications **must** have a constructor that accepts an ATeam object. This constructor must exist in addition to any other user-defined constructors.

Sample constructor accepting ATeam object argument

```
public UserProgram( Ateam o ) { }
```

MPJ Error

If ATeam complains about MPI initialization, the user may not have passed the correct arguments to ATeams MPI implementation of the Init function. Be sure to invoke `MPJ.Init(args, ateam)`. Any other invocation (i.e. `MPJ.init(args)`) will result in execution errors.

Bookkeeper Bottleneck/Crashes

In order to avoid performance issues stemming from bottleneck at the Bookkeeper accepting and retrieving snapshots, ensure that there are multiple Bookkeeper Agents running. If one crashes the others can handle the load and if there are a lot of snapshots coming in from other agents having multiple Bookkeepers allow these messages to be processed more quickly.

Hanging while "... mainThread waiting for all_locations"

This message means that all the agents specified by the user have not been started. Review the application code ensuring that all agents specified by the program are instantiated.

Error java.net.ConnectException: Connection refused

Check that all locations specified to run the application are also running UWPlace. This error occurs most often when UWPlace is not running on a target machine.

8. Final Comments

AgentTeawork, ATeam and associated classes are copy write University of Washington Bothell. No advanced notice of changes and revisions to AgentTeamwork and ATeam or relate classes are required. Users may use classes and associated methods at their own risk.

Appendix. System Directory

The following table summarizes AgentTeamwork's directory structure. All scripts necessary for job set-up and execution are located at the "**agentteamwrok/scripts**" directory.

Directory Structure

```
agentteamwork/   ..............................the root of the AgentTeamwork system
             AgentTeamwork/
                          Agents/   ..........................includes all agents
                          Ateam/   ...................includes AgentTeamwork APIs
                          GridFile/   ...................fault-tolerant file I/O
                          GridJNI/    ..............interface to C++ applications
                          GridRuby/   .............interface to Ruby applications
                          GridTcp/    ..........................fault-tolerant TCP
             applications/
                         applications.jar   ..............all applications so far
                         compile.sh   ...........a script to compile applications
                         runAteam.sh   .........a script to inject an application
                         DistributedGrep/ ........a distributed-grep Java program
                         Mandelbrot/   .................a Mandelbrot Java program
                         MatrixMult/   ......a matrix-multiplication Java program
                         Wave2D/   ..........a Schroedinger's wave Java simulator
                         Samepl/   ..........................a sample Java program
             benchmark/
                       runAteam.sh   .......a script to inject a benchmark program
                       runAteamClusters.sh   .......a script to use multi-clusters
                       runRsc.sh   ........a script to use a resource XML database
             doc/   ...................................................JavaDoc files
             GUI/   .......................AgentTeamwork's GUI including SubmitGUI
             jars/   ................................all AgentTeamwork's jar files
             MPJ/   ....................................MPI Java implementation
             scripts/
                     runUWPlace.sh   ..a script to launch a UWAgent daemon locally
                     bgUWPlace.sh   ......to launch a UWAgent daemon in background
                     sshUWPlace.sh   .....to start bgUWPlace.sh at remote machines
                     kill.sh   .....to kill a local UWAgent daemon, (i.e. UWPlace)
                     fileTrunc.sh    ......a script internally used by sshTrunc.sh
                     sshKill.sh   .........a script to kill remote UWAgent daemons
                     sshTail.sh   .........to view the tail of remote daemons' log
                     sshDelete.sh    .......a script to delete remote daemons' log
                     sshTrunc.sh    ......a script to truncate remote daemon's log
                     sshUWMonitorAs.sh   ........a script to monitor remote agents
                     compileAndPackUWAgent.sh   .......a script to compile UWAgent
                     compileAndPackAgents.sh   .....a script to compile all agents
                     compileAndPackAteam.sh   ............a script to compile APIs
                     compileAndPackMPJ.sh   ..........a script to compile MPI Java
                     compileAndPackGUI.sh   ..............a script to compile GUI
                     compileAndPackBenchmark.sh   ...to compile benchmark programs
                     compileAndPackApplications.sh   ..to compile all applications
                     compileAndPackAll.sh   ..................to compile all files
                     genJavaDoc.sh   ..........a script to generate Java documents
                     cleanClassFiles.sh   ......a script to delete all class files
                     runFileThread.sh   .........to launch a file-transfer process
                     launchGUI.sh   ........a script to launch a SubmitGUI process
                     runDBNotify.sh   ...........to inform XDBase of node recovery
                     shutdownDB.sh   .........a script to shutdown an XML database
                     startDB.sh   ............a script to start up an XML database
                     UWMonitor.sh   ........a script to monitor/kill a local agent
                     XDBaseGUI.sh   ......a script to monitor a local XML database
                     bat/ ....DOS batch files that basically correspond to scripts
             UWAgent/   ......................UWAgent mobile-agent execution engine
             SampleAgents/   .........................sample mobile-agent programs
```

```
                xmls/   ......................XML descriptions of computing resources
```