A Development of Resource/Commander Agents Used in AgentTeamwork Grid Computing Middleware

Inter-mediate Report

Enoch Mak
Professor Munehiro Fukuda
CSS497 Faculty Research Internship
March 17, 2005

I. Introduction

The UWB Distributed Systems Laboratory has been developing the AgentTeamwork grid computing middleware that dispatches mobile agents to coordinate an execution of a user application over remote computers. UWAgent and UWPlace, the mobile agent execution engine, have been implemented, so that mobile agents are able to migrate over Internet and keep running on a different machine. There are four types of mobile agents, commander, resource, sentinel, and bookkeeper agents. All of them are extended from the UWAgent class. The resource agent is responsible to access a central ftp server, downloads new resource XML files from it, maintains these files in it eXist database, returns a list of remote computers to its commander agent, and periodically checks the status of all remote computer nodes which are enumerated in the local database.

In this project, I will complete the first version of resource agent and enhance the commander agent which can communicate with resource agent by exchanging messages. The development will be performed in parallel for the resource and commander agents.

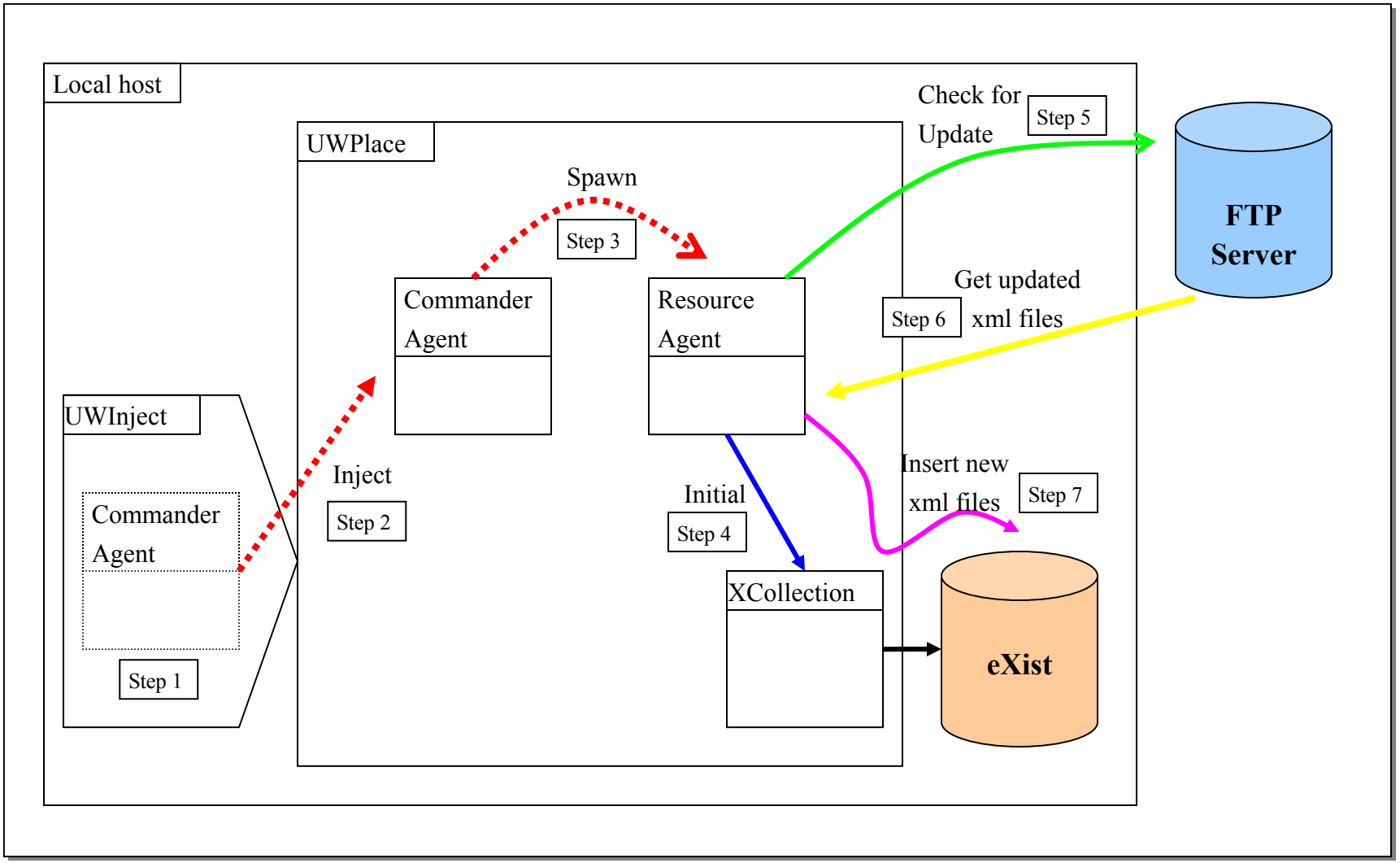
II. Progress

The project consists of seven phases. I have to finish all these phases in two quarters. The projector schedule is shown in the table below. Under the help and supervision of Professor Fukuda, the project has been running smoothly. I have finished the scheduled first four phases of the project. Up to now, the resource agent and commander agent have been ported to the medusa cluster. The resource agent is able to access the local eXist database through the XCollection class which is eXist interfacing class. The round trip communication between commander agent and resource agent has been successfully implemented. In the other words, the resource agent can return a list of available computer ip name upon the request from the commander agent. Furthermore, the resource agent is able to periodically check and update the status of all remote computer nodes which are enumerated in the local database.

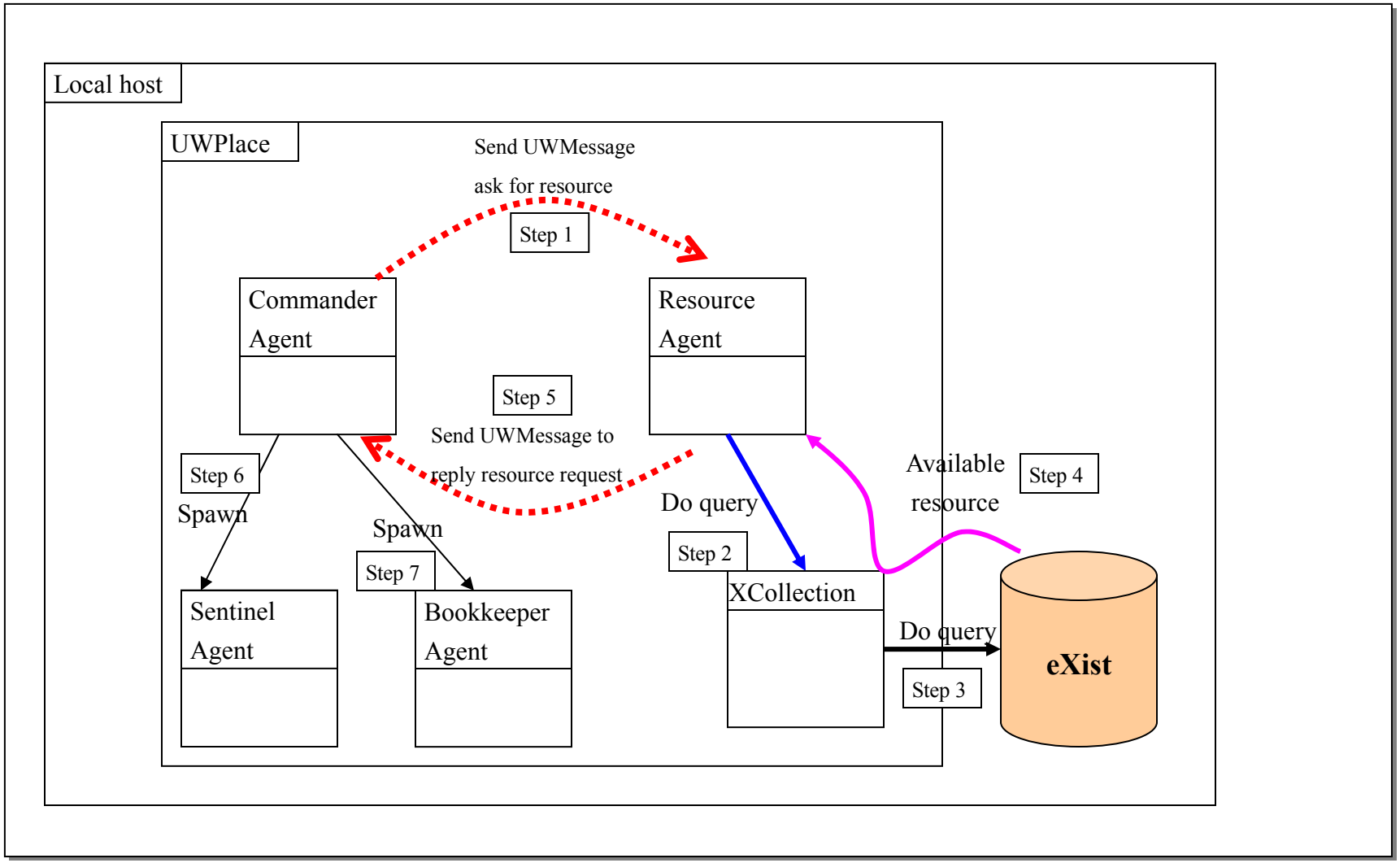
Project Schedule Table		
Qtr	**Week**	**Work Items**
Wi	1	Phase 0: Understand Shane's work and port it to medusa.
Wi	2	Phase 0: Complete Phase 0 and verify the correctness.
Wi	3	Phase 1: Implement an Xcollection.java that is a general eXist-interfacing class.
Wi	4	Phase 1: Complete Phase 1 and verify the correctness.
Wi	5	Phase 2: Implement a round-trip communication from commander to resource.
Wi	6	Phase 2:
Wi	7	Phase 2: Complete Phase 2 and verify the correctness.
Wi	8	Phase 3: Add periodic-probing code to the resource agent.
Wi	9	Phase 3:
Wi	10	Phase 3: Complete Phase 3 and veriy the correctness.
Wi	11	Inter-mediate report and demonstration
Sp	1	Phase 4: Merge your code into the latest version of AgentTeamwrok.
Sp	2	Phase 4:
Sp	3	Phase 4: Complete Phase 4 and verify the correctness.
Sp	4	Phase 5: Allow multiple resource agents to share an eXist DB
Sp	5	Phase 5:
Sp	6	Phase 5:
Sp	7	Phase 5: Complete Phase 5 and verify the correctness.
Sp	8	Phase 6:
Sp	9	Phase 6:
Sp	10	Phase 6:
Sp	11	Final report and demonstration

III. Implementation Diagram

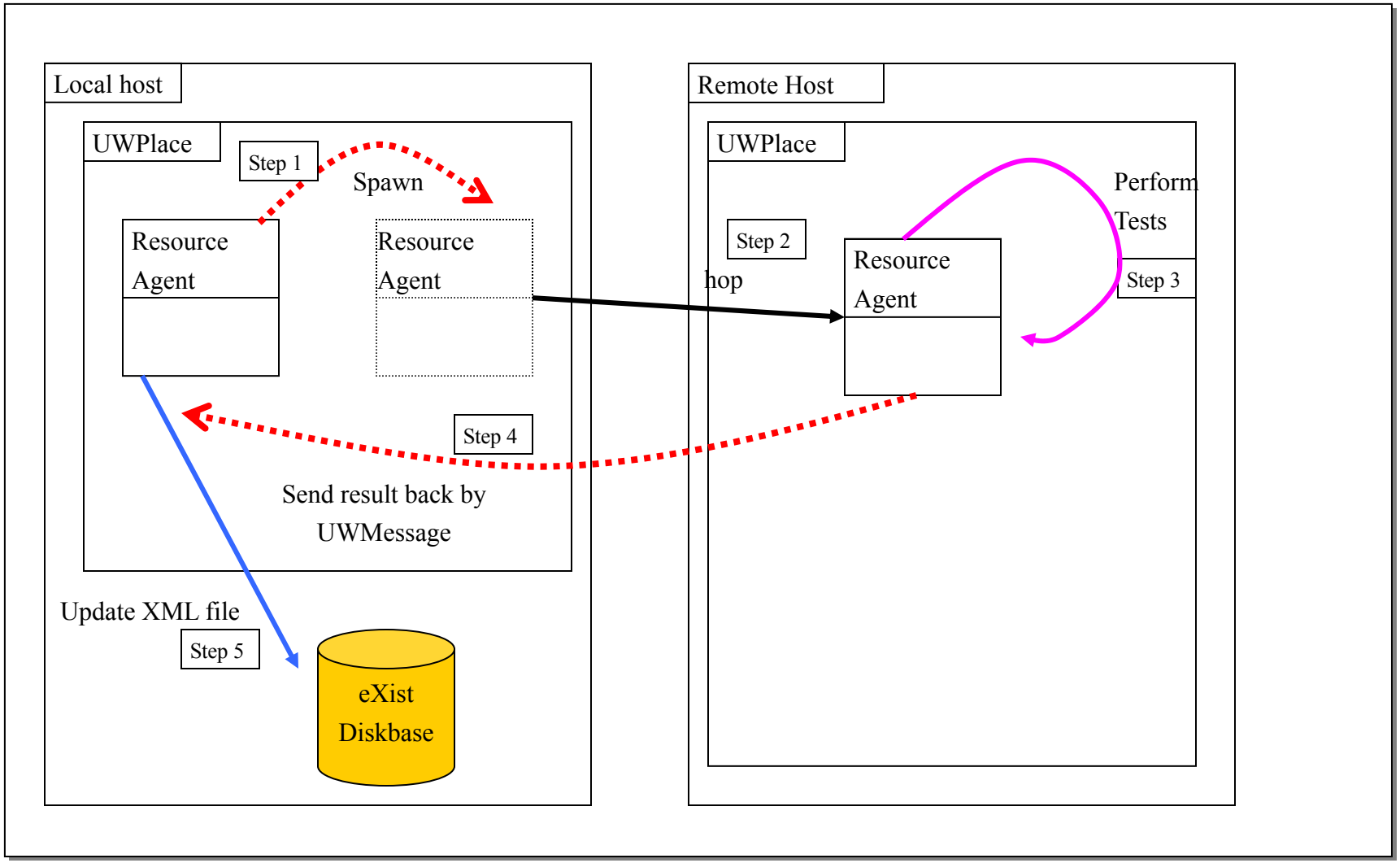
A. AgentTeamwork Startup



B. Round-trip communication of CommanderAgent and ResourceAgent



C. Periodic-probing of ResourceAgent



IV. ResourceAgent API

The following section is the Application Programming Interface of the ResourceAgent class.

Data Members	
Member:	private Hashtable rscArgsTable;
Description:	contain the types of ResourceAgent args for contstuctor
Member:	private int numRqdArgs = 5
Description:	number of required args for ResourceAgent constructor
Member:	private transient XCollection localDataBase = null
Description:	An instance of the XCollection database
Member:	String rsc_Collection_Name = "resources"
Description:	Name of Resource Collection
Member:	private boolean isReadyQuery = false
Description:	Status of "resources" database Collection, does it ready to perform query
Member:	private FTPClient ftp
Description:	An instance of the Ftp client
Member:	private boolean isConnectedToFTPServer = false
Description:	True if the ftp client is connected to Ftp Server
Member:	private String rscXMLDir = "ResourceXML"
Description:	Name of the directory in FTP server where all resource xml files are located
Member:	private long FTPTimeOffset
Description:	Time zone difference between FTP server and local machine

Member:	private Timer probeTaskTimer
Description:	A timer instance which can schedule the ResourceAgent periodic probing task
Member:	private boolean isPerformTest = false
Description:	Determine this resourceAgent thread is for remote probing(or so-called “perform test” mode) or not
Member:	private String PerformTestHostName
Description:	The ip Name of the targeting computer node. (only if this thread is for remote probing or so-called “perform test” mode)
Member:	private Vector allRemoteNode = null
Description:	All remote host name to perform periodic probing
Member:	private Process ttcpServerProcess = null
Description:	Sub-process for ttcp Server(for bandwidth testing)
Member:	private List childAgentIds = new ArrayList()
Description:	All ResourceAgent's child Ids
Member:	private transient Thread receiveMessageThread = null
Description:	Receive message sub-thread
Member:	private transient volatile boolean receiveMessageThreadContinue
Description:	Continue the receive message sub-thread or not
Member:	private transient volatile boolean main_thread_cont
Description:	determine main thread continue or not

Methods	
Function:	ResourceAgent(String[] args)
Description:	ResourceAgent is launched with this constructor in general. It receives an array of String arguments which serves in two main purposes. The first purpose is to determine this ResourceAgent thread is a "regular" ResourceAgent or a "Perform test" ResourceAgent which will be migrated to remote computer node to perform tests. The second purpose is to provide some information from the user for starting the resourceAgent.
Arguments:	args: this String array incluldes the following options and corresponding arguments. If the args[0] is equal to "PerformTest" and args length is 2, which mean the resourceAgent is in "Perform test"(remote probing) mode, args[1] should contain the host name of the targeting remote computer node. If resourceAgent is in "regular" mode, then the args should contain all corrresponding informations which is stated in getAllConstrArgs() method.
Calls:	getAllConstrArgs()
Called by:	none
Return:	none
Function:	getAllConstrArgs(String[] args)
Description:	This function will put the resource args into a hashtable.
Arguments:	args: array containing all the resource requirements for this task
Calls:	getAllConstrArgs()
Called by:	ResourceAgent(String[] args)
Return:	none
Function:	usage(String errMes)
Description:	Print how to launch a resource agent.
Arguments:	args: String message need to print out
Calls:	none
Called by:	ResourceAgent(String[] args)
Return:	none

Function:	init ()
Description:	This is the first fuction executed after a resource agent is constructed. In "perform testing" mode: print out my identity, migrate to the targeting computer node to perform "performTest" method. In "regular" mode: print out my identity, start the receive message thread, initialize the local database, call the "mainMethod" method
Arguments:	none
Calls:	initDB(), mainMethod();
Called by:	N/A
Return:	none
Function:	initDB()
Description:	This fuction init the local data through the XCollection class.
Arguments:	none
Calls:	XCollection.initCollection()
Called by:	init()
Return:	none
Function:	fatalError()
Description:	Print an error message with the "Resource: " header and exit the program.
Arguments:	args: String error message need to print out
Calls:	none
Called by:	N/A
Return:	none
Function:	run()
Description:	All new threads initially call this function, however each of them is eventually dispatched and dedicated to a different function.
Arguments:	none
Calls:	awaitInstruction ()
Called by:	init()
Return:	none

Function:	awaitInstruction()
Description:	This fuction is called by the receiveMessage sub-thread to keep checking for new message/instruction from other uwagent.
Arguments:	none
Calls:	respondToMessage ()
Called by:	run()
Return:	none
Function:	respondToMessage()
Description:	This fuction is called by the awaitInstruction method to respond to different message.
Arguments:	message: A UWMessage to response to
Calls:	isReadyForQuery(), replyResourcesNeededRequest(), updateRemoteNodeCurrentStatus(), killAgentMessage()
Called by:	awaitInstruction ()
Return:	none
Function:	updateRemoteNodeCurrentStatus ()
Description:	This fuction is call by the respondToMessage method after remote probing result is returnd from child by UWMessage. The resultare used to update the local database
Arguments:	message a UWMessage contain the result from remote computer node
Calls:	XCollection.query (), XCollection.update()
Called by:	respondToMessage ()
Return:	none
Function:	xUpdateStatementGenerator ()
Description:	This fuction is called by the updateRemoteNodeCurrentStatus method to generate a xUpdate statement according to the args .
Arguments:	remoteNodeName: The remote computer node ipName xmlElement: The element in the xml file should be updated to. Value: The xmlElement should be updated by this value.
Calls:	none
Called by:	updateRemoteNodeCurrentStatus ()
Return:	String: a xUpdate statement

Function:	replyResourcesNeededRequest()
Description:	This function is called by the respondToMessage method after received a "ResourcesNeedRequest" message from another agent (Mostly CommandAgent). This function will call other function to do a query in the local database and return the result by uwmessage.
Arguments:	message: a message contain the resource requirement
Calls:	queryForIpName ()
Called by:	respondToMessage ()
Return:	none
Function:	queryForIpName()
Description:	This function is called by the replyResourcesNeededRequest method after received a "ResourcesNeedRequest" message from another agent(Mostly CommandAgent). This function will query the local database and return the results (ipName) in a vector.
Arguments:	message: a message contain the resource requirement
Calls:	XCollection.query(), checkIpNameTimeSlot()
Called by:	replyResourcesNeededRequest ()
Return:	Vector: a list of ipName which is available
Function:	checkIpNameTimeSlot ()
Description:	This function is called by the queryForIpName method after received a "ResourcesNeedRequest" message from another agent (Mostly CommandAgent). This function will query the local database to check if the computer node(ipName) is available in the specified time(executeTime)
Arguments:	ipName: the remote computer node this function is going to check executeTime: time to execute the program
Calls:	XCollection.query ()
Called by:	queryForIpName ()
Return:	Boolean: if the specified executeTime is fall into any time slot or not
Function:	MainMethod ()
Description:	Is executed by the main thread. An return from mainMethod means a

	successful termination of this commander agent. This main thread takes care here of: update the local database from the ftp server, start the ttcp server for bandwidth test, start the periodic probing, Waiting for the commanderAgent to send end instruction, Stopping all sub threads, Stopping the ttcp server
Arguments:	none
Calls:	updateDB (), startTtcpServer(), scheduleProbingTasks(), stopSubThreads(), stopTtcpServer()
Called by:	init ()
Return:	none
Function:	stopSubThreads()
Description:	Is called by mainMethod( ) at its end. This method terminates all sub threads.
Arguments:	none
Calls:	receiveMessageThreadStop()
Called by:	mainMethod()
Return:	none
Function:	receiveMessageThreadStop ()
Description:	Is called by stopSubThreads( ). This method terminates the receive message thread
Arguments:	none
Calls:	restartThread ()
Called by:	stopSubThreads ()
Return:	none
Function:	updateDB ()
Description:	This function is call by the mainMethod method. This function will doing the following: Call connectToFTPServer method to connect to ftp server, Call getResourcesListFromFTPServer method to get the file, listing in the FTP server directory, Call getResourcesListFromDatabase method to get the file listing of all XML resources in the local database, Call downLoadXMLResourcesFromFTPServer method to download new

	xml file from the ftp server and update the local database, return whether all operations are successfully performed or not
Arguments:	none
Calls:	connectToFTPServer (), getResourcesListFromFTPServer (), getResourcesListFromDatabase (), downLoadXMLResourcesFromFTPServer ()
Called by:	mainMethod ()
Return:	Boolean: return whether all operations are successfully performed or not
Function:	connectToFTPServer ()
Description:	This function is call by the updateDB method. This function will doing the following: Start the ftp client, connect to the ftp server, login to the ftp server, if succeed, enable passive mode, call getFTPServerTimeOffset method to compute the time(timezone) difference between the ftp server and the local database, if not succeed, discount to the ftp server
Arguments:	none
Calls:	FTPclient.connect(), FTPclient.login(), FTPclient.getreplycode(), FTPclient.disconnect(), FTPclient.enterLocalPassiveMode(), getFTPServerTimeOffset()
Called by:	updateDB ()
Return:	Boolean: successfully connected to ftp server or not
Function:	getFTPServerTimeOffset()
Description:	This function is called by the connectToFTPServer method. This function compute the time offset between the local database and the ftp server
Arguments:	none
Calls:	FTPclient.storeFile(), FTPclient.listFiles(), FTPclient.deleteFile()
Called by:	connectToFTPServer ()
Return:	long: the time offset between the local database and the ftp server
Function:	getResourcesListFromFTPServer ()
Description:	This function is called by the updateDB method. This function will get

	xml file list from the ftp server
Arguments:	none
Calls:	FTPclient.listFile()
Called by:	updateDB ()
Return:	FTPFile : xml file list in the ftp server
Function:	getResourcesListFromDatabase ()
Description:	This fuction is called by the updateDB method. This function will get xml file list from the local database
Arguments:	none
Calls:	XCollection.getResourceList()
Called by:	updateDB ()
Return:	Vector : xml file list from the local database
Function:	downLoadXMLResourcesFromFTPServer ()
Description:	This fuction is call by the updateDB method. This function will doing the following: check the timestamp of each file in the ftp listing, Update the local database if the file is not exist in the database or the file in the ftp server is more update
Arguments:	ftpRscXMLFilesList: xml file list in the ftp server dbRscXMLFilesList: xml file list from the local database
Calls:	XCollection.GetFileLastModDate(), addResourceFromFtpServerToDB()
Called by:	updateDB ()
Return:	Boolean: successfully perform the operation or not
Function:	addResourceFromFtpServerToDB ()
Description:	This function is called by the downLoadXMLResourcesFromFTPServer method. This function will download the specified file from the ftp server and insert to the local database.
Arguments:	ftpResourceFile: file name to specify which file to download
Calls:	FTPclient.retrieveFile(), XCollection.insert()
Called by:	downLoadXMLResourcesFromFTPServer ()

Return:	none
Function:	killAgentMessage()
Description:	This fuction is call by the respondToMessage method after receive the “kill_agent" message from other agent(mostly commanderAgent). This function will Send end message to all the child agents, Stop the mainMethod while loop
Arguments:	Message: "kill_agent" message received from other agent
Calls:	AgentUtil.sendEndMessage()
Called by:	respondToMessage()
Return:	none
Function:	scheduleProbingTasks()
Description:	This fuction is call by the mainMethod method. This function will start the schedule Probing tasks timer, and schedule the probing test by the user specified value
Arguments:	none
Calls:	Timer.schedule()
Called by:	mainMethod()
Return:	none
Function:	performTest()
Description:	This fuction is called by the Init method when the resourceAgent is in "perform test" mode and located in remote computer node. This function will call findOsType method and put the result in the hashtable, call findMemSize method and put the result in the hashtable, call findCPULoad method and put the result in the hashtable, call findDiskSize method and put the result in the hashtable, call findBandwidth method and put the result in the hashtable, Send the hashtable in a "remote_node_run_time_stats" message to the Parent resourceAgent
Arguments:	none
Calls:	findOsType(), findMemSize(), findCPULoad(), findDiskSize(), findBandwidth()
Called by:	init()

Return:	none
Function:	findMemSize()
Description:	This function is called by the performTest method when the resourceAgent is in "perform test" mode and located in remote computer node. This function will find out the memory size of the computer node
Arguments:	none
Calls:	Runtime.getRuntime()
Called by:	performTest()
Return:	String: memory size
Function:	findDiskSize()
Description:	This function is called by the performTest method when the resourceAgent is in "perform test" mode and located in remote computer node. This function will find out the disk size of the computer node
Arguments:	none
Calls:	Runtime.getRuntime()
Called by:	performTest()
Return:	String: disk size
Function:	findOsType()
Description:	This function is called by the performTest method when the resourceAgent is in "perform test" mode and located in remote computer node. This function will find out the OS type of the computer node
Arguments:	none
Calls:	System.getProperty()
Called by:	performTest()
Return:	String: os type
Function:	findCPULoad()
Description:	This function is called by the performTest method when the resourceAgent is in "perform test" mode and located in remote

	computer node. This function will find out the CPU load of the computer node
Arguments:	none
Calls:	Runtime.getRuntime()
Called by:	performTest()
Return:	String: CPU load
Function:	findBandwidth()
Description:	This function is called by the performTest method when the resourceAgent is in "perform test" mode and located in remote computer node. This function will find out the bandwidth of the computer node
Arguments:	none
Calls:	Runtime.getRuntime()
Called by:	performTest()
Return:	String: bandwidth
Function:	startTtcpServer()
Description:	This function is called by the mainMethod method. This function will execute the ttcp server for bandwidth test
Arguments:	none
Calls:	Runtime.getRuntime()
Called by:	mainMethod()
Return:	none
Function:	stopTtcpServer()
Description:	This function is called by the mainMethod method. This function will destroy the ttcp server
Arguments:	none
Calls:	None
Called by:	mainMethod ()
Return:	none

V. XCollection API

The following section is the Application Programming Interface of the XCollection class.

Data Members	
Member:	private String db_Driver = "org.exist.xmldb.DatabaseImpl";
Description:	The driver of the local eXist database
Member:	private String root_URI = "xmldb:exist:///db/";
Description:	The location of the root URI of the local database
Member:	private String userName = "admin";
Description:	The user name of the local database
Member:	private String pswd = "";
Description:	The password for the local database
Member:	private Collection root_Col = null;
Description:	A collection instance which is the root collection
Member:	private Hashtable col_Table = null;
Description:	A Hashtable instance to store all the Collections in the database. This implementation allow the local database to have more than one collections at the same time.
Member:	private Collection current_Col = null;
Description:	Store which collection is currently working on
Member:	private String current_Col_Name = null;
Description:	Store the collection name which is currently working on

Methods	
Function:	XCollection(String p_DataBase_Home_Dir)
Description:	The constructor will start the exist XML:DB database if both appropriate and necessary (If there is a exist database on the current host machine and if the exist database has not already been started.)
Arguments:	p_DataBase_Home_Dir: this String is used to set the system property
Calls:	none
Called by:	none
Return:	none
Function:	initCollection
Description:	This function will create a collection specified by the String p_Collection_Name, if a collection does not exist. It will also open a connection to the collection specified by the String p_Collection_Name parameter, if a connection does not already exist. The new collection will store in the hash table, the p_COllection_Name will become the key of this collection. The current_Col will point to this new collection. The current_Col_Name will store the current collection name
Arguments:	p_Collection_Name: a string which store the collection name
Calls:	shutdown()
Called by:	N/A
Return:	none
Function:	switchCurrentCollection
Description:	This function will check if the collection specified by the String p_Collection_Name is initialized and existed in the hashtable. If so, it will make the current_Col point to that collection, the current_Col_Name will be updated to that collection name and return true; If the collection specified in p_Collection_Name is not initialized and not existed in the hashtable, it will return false.
Arguments:	p_Collection_Name: a string which store the collection name
Calls:	none
Called by:	N/A

Return:	Successfully switch the collection or not
Function:	insert
Description:	function to insert a new xml document to specified collection in the database
Arguments:	element: the path to the new document p_Collection_Name: a string which store the collection name
Calls:	none
Called by:	N/A
Return:	none
Function:	query
Description:	searching for specific data specified by "xpath" in each XML file in the specified collection.
Arguments:	xpath: this String is the xPath statement p_Collection_Name: a string which store the collection name
Calls:	none
Called by:	N/A
Return:	Vector: query result
Function:	update
Description:	Update the selected collection with the xUpdate statement provided
Arguments:	xUpdate: this String is the xUpdate statement p_Collection_Name: a string which store the collection name
Calls:	none
Called by:	N/A
Return:	Long: the number of nodes being updated
Function:	delete
Description:	delete a xml file specified by the input 'element' from the specified collection
Arguments:	element: the path to the document p_Collection_Name: a string which store the collection name
Calls:	none

Called by:	N/A
Return:	true if remove the file from the db successfully, false otherwise
Function:	retrieve
Description:	function to retrieve a desired xml file from the database
Arguments:	element: the path to the document p_Collection_Name: a string which store the collection name
Calls:	none
Called by:	N/A
Return:	String: the content of the file if found, error msg otherwise
Function:	getResourceList
Description:	function to retrieve a Resource list from a collection
Arguments:	p_Collection_Name: a string which store the collection name
Calls:	none
Called by:	N/A
Return:	Vector: resource file list of the specified collection
Function:	GetFileLastModDate
Description:	function to retrieve the last modify date of a specify file(resource) in a collection
Arguments:	element: the path to the document p_Collection_Name: a string which store the collection name
Calls:	none
Called by:	N/A
Return:	Date: the timestamp of the file
Function:	shutDown
Description:	Shutdown current collection
Arguments:	none
Calls:	none
Called by:	initCollection()
Return:	none

VI. XML Database (eXist)

This section describes how eXist, open source native XML, database is used in this project. For further information about eXist, please refer to the Appendix section.

A. eXist as an embedded instance

   Currently, the eXist database is implemented as an embedded instance in the XCollection Class. It equips with the following features:

   1. Create and remove collections
   2. Retrieve, store, remove and query XML files
   3. Allows multiple collections exist at the same time.

B. The eXist database can be query and update by using xPath and xUpdate language respectively. The tutorial for xPath and xUpdate language can be found in the Appendix section.

VII. Files Description

This section briefly describes all files which are related to this project. Since I spent a lot of time to start this project, I think it is a good idea to document every single file that I have been used or updated.

Files Description	
File Name:	ResourceAgent.java
Location:	/home/uwagent/enoch/UWAgent
Description:	This is the source file of the ResourceAgent
File Name:	XCollection.java
Location:	/home/uwagent/enoch/UWAgent
Description:	This is the source file of the XCollection class
File Name:	CommanderAgent.java
Location:	/home/uwagent/enoch/agents
Description:	This is the source file of the CommanderAgent
File Name:	cr.sh
Location:	/home/uwagent/enoch/UWAgent
Description:	This is the shell script to compile the ResourceAgent, all the necessary classpaths for compilation are included in this script. After the compilation, the file ResourceAgent$RemoteRscProbeTask.class will be copy to /home/uwagent/enoch/agents for CommanderAgent.
File Name:	cx.sh
Location:	/home/uwagent/enoch/UWAgent
Description:	This is the shell script to compile the XCollection, all the necessary classpaths for compilation are included in this script
File Name:	cc.sh
Location:	/home/uwagent/enoch/agents
Description:	This is the shell script to compile the CommanderAgent, all the necessary classpaths for compilation are included in this script

File Name:	run.sh
Location:	/home/uwagent/enoch/UWAgent
Description:	This is the shell script to start the RMI server and UWPlace. A port number need to be provided when running this script file. For example, “run.sh 35353”
File Name:	runittry.sh
Location:	/home/uwagent/enoch/UWAgent
Description:	This is the shell script to inject the CommanderAgent and start the ResourceAgent
File Name:	ResourceAgent$RemoteRscProbeTask.class
Location:	/home/uwagent/enoch/agents
Description:	This is the sub class of ResourceAgent. In order to start the AgentTeamwork correctly, this file should be resided in the same directory with CommanderAgent.
File Name:	ttcp
Location:	/home/uwagent/enoch/UWAgent and /home/uwagent/enoch/agents
Description:	This is the ttcp program for the ResourceAgent to run the bandwidth testing.
File Name:	eXist database directory
Location:	/home/uwagent/eXist/webapp/WEB-INF/data
Description:	This is the directory which eXist store all the xml file.
File Name:	mnode0.xml – mnode7.xml
Location:	ftp://agentTeamWork:test@ftp.tripod.com/ResourceXML
Description:	All the xml files are located in the ftp server

VIII. Starting AgentTeamWork

This section describes the steps to start the AgentTeamwork.

A. Two easy steps to start AgentTeamwork by using the shell scripts:

1. Starting the RMI Server and UWPlace

   Use the shell script run.sh to start the RMI Server and UWPlace. Simply just type in the shell script and the port number. For example, I want to start the RMI Server with port 35353:

   run.sh 35353

2. Injecting the CommanderAgent

   Use the shell script runittry.sh to injecting the CommanderAgent. Just type in the script name like below:

   runittry.sh

B. Start AgentTeamwork manually without using the shell scripts:

1. Setting CLASSPATH

   The environment variable need to be set correctly otherwise many compile time and run time errors might be occurred. You can either update the environment variable setting or use the –classpath option when you start the java virtual machine. Below is a list of all necessary CLASSPATH to launch AgentTeamwork.

   –classpath
   /home/uwagent/mpiJava/lib/classes:/home/uwagent/MA/benchmark/MPJv1.0:/home/uwagent/eXist/exist.jar:/home/uwagent/eXist/lib/core/xmldb.jar:/home/uwagent/eXist/lib/core/resolver-20030708.jar:/home/uwagent/eXist/lib/core/jakarta-oro-2.0.6.jar:/home/uwagent/eXist/lib/core/antlr.jar:/home/uwagent/eXist/lib/core/xmlrpc-1.2.jar:/home/uwagent/eXist/lib/core/commons-pool-1.1.jar:/home/uwagent/eXist/lib/endorsed/xerces-2.6.1.jar:/home/uwagent/eXist/lib/endorsed/xalan-2.5.2.jar:/home/uwagent/eXist/lib/endorsed/xml-apis.jar:/home/uwagent/eXist/lib/core/log4j.jar:/home/uwagent/eXist/commons-httpclient-2.0.2/commons-httpclient-2.0.2.jar:/home/uwagent/eXist/commons-net-1.3.0/commons-net-1.3.0.jar:/home/uwagent/enoch/UWAgent/UWAgent.jar:/home/uwagent/enoch/GridTcp/GridTcp.jar:/home/uwagent/enoch/agents:/home/uw

agent/enoch/UWAgent:/home/uwagent/eXist:/home/uwagent

2. Starting the RMI Server
   Start the RMI server by typing the follow command, 35353 is the port number to run the server:

   rmiregistry 35353 &

3. Starting the UWPlace
   Starting the UWPlace by simply type in the command below (assume that the environment variables have been set correctly):

   java UWPlace&

4. Injecting the CommanderAgent by type in the command below (assume that the environment variables have been set correctly):

   java UWInject localhost CommanderAgent -u "/home/uwagent/enoch/agents" -p UWPlace -n 35353 -c myClient -s ResourceAgent\$RemoteRscProbeTask -m 100 RQ_ip_10.1.0.0_ip_10.1.0.1_ip_10.1.0.2_ip_10.1.0.3_cpuspeed_456_cpucount_5_memory_45_os_linux_disk_45_total_2_time_180100_cpuarch_i386_cpuload_80_bandwidth_800 RA_ftp.tripod.com_agentTeamWork_test_1_/home/uwagent/eXist RC_ResourceAgent\$RemoteRscProbeTask U_shane_arg1_arg2_arg3 C_class1_class2_class3

   Please refer to Professor Fukuda's UWAgents User's Manual for more detail about the original arguments of CommanderAgent. The updated arguments are shown in the following table.

Updated arguments of CommanderAgent	
{S_ipname{[_ipname]}}	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.

{B_ipname{[_ipname]}}	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.
{R_ipname}	An ip name to dispatch a resource agent. If it is not given, a resource agent is launched at the same computing node as the commander is working.
{E_ipname{[_ipname]}}	A list of extra ip names to resume an agent when one is crashed. If it is not given, a resource agent is responsible to provide the commander with such a list.
U_programName{[_argument]}	Expresses a user program name and its arguments. This option is mandatory. .
{C_classname{[_classname]}}	A list of classes accessed from and thus carried with a user application.
{RA_argument{[_argument]}	A list of arguments passed to ResourceAgent
{RQ_option_parameter{[_option_parameter]}	A list of query option/paramter pairs.
{RC_argument{[_argument]	A list of classes passed to ResourceAgent

IX. Appendix

The following is a list of resources that are useful for the development of resource agent.

1. eXist Home Page
   http://exist.sourceforge.net/

2. eXist API Specification
   http://exist.sourceforge.net/api/

3. xPath tutorial
   http://www.w3schools.com/xpath/default.asp

4. xUpdate tutorial
   http://www.xmldatabases.org/projects/XUpdate-UseCases/
   http://xmldb-org.sourceforge.net/xupdate/xupdate-wd.html