AgentTeamWork Mobile Agent Manual

I Introduction

In AgentTeamwork Mobile Agent, each agents are communicating with each other to get the job done. In this manual, how to inject these agents and each components are explained.

II How to Use

Each agent works within the domain of UWPlace, so before injecting any agents, you have to start UWPlace just like other UWAgents.

1. How to inject Commander Agent, Sentinel Agent, and BackupAgent
You can inject agents by using UWInject.
$ java UWInject (hostname) CommaderAgent S_(the number of Sentinel Agent)_(my rank)_(host name)_(gateway name)_(my rank)_(hostname)_(gateway name)… B_(hostname) U_(the name of user program) -s (the list of classes used by Commander Agent) –u (the location of the class files)

Example:
$ java UWInject localhost CommanderAgent S_2_1_mnode2_-_2_mnode3_-S_2_2_mnode3_-_1_mnode2_- U_UserProg2 B_mnode5_mnode6 –s SentinelAgent, BackupAgent, UserProg2 –u ~/MA/agents

2.How to compile

1. Log in to mnode and set the class path

```
        $ ssh mnode1
        $ export
CLASSPATH=$CLASSPATH:.:~/MA/UWAgent/UWAgent.jar:~/MA/GridTcp/GridTcp.ja
r
```

2. recreate the UWAgent.jar by compiling the updated files under ~/MA/UWAgent/ if any changes have been made to the UWAgent files which include AtentUtil.java.

```
        $ cd ~/MA/UWAgent
        $ ./uwacompile.sh
```

3. simply compile CommanderAgent.java, SentinelAgent.java, or Backup Agent.java using javac

```
        $ cs ~/MA/agents/
        $ javac BackupAgent.java
```

III Each Components(Agents)

1 Commander Agent

1.1 Overview
A new Commander Agent is allocated to a new job, and spawns Sentinel agents and Backup Agents.

1.2 Design sheet

a. **Summary**
The **init** method is calls at first. In the **init** method, receiveMessageThread is started and the Commander Agent spawns(by using **spawn** method) Sentinel Agents according to the **sList** and **sArgs**. Likewise, the Commander Agent spawns Backup Agents according to the **bList** and **bArgs**. **mainMethod** is also called.

The **run** method is called to start the Commander Agent. In the **run** method, **awaitInstruction** is called to wait for the instruction(message) from other agents.

Message Type List
send_snapshot
-the snapshot information received from the backup agent

agents_locations2
-the adding new sentinel agent id request received

end_of_userprogram
-the message received from sentinel agent as a notification of the end of the user program

b. **Methods and Constructor List**
public void printErr(String errMes)
Description: print error message
Precondtion:errMes is a string which indicates the error message to be displayed.
Postcondtion:the program will be terminated
Calls:none

public CommanderAgent(String[] args)
Description: constructor
Precondition:args is the array of string which indicate the arguments of Agents.
Postcondition:each agents has correct arguments.

public CommanderAgent()
Description: default constructor
Precondition:none
Postcondition:none
calls: none

public void init()
Description: the first function to be called
Precondition:none
Postcondition:receiveMessageThread is started
Calls: receiveMessageThread()

private void spawn()

Description : spawns Sentinel Agents and Backup Agents according to the list
Precondition:none
Postcondition:returns none. Sentinel Agents and Backup Agents are spawned in the locations
Calls:spawnChildAtPlace()

public void spawnChildAtPlace(String destHost, String type, String[] args)
Description: spawns an Agent in the destination
Precondition:none
Postcondition:returns none.Sentinel Agents or Backup Agents are to be spawned in the destination.
Calls:spawnChild()

public Thread receiveMessageThread()
Description : starts receiveMessageThread()
Precondition:none
Postcondition:returns a thread which is a receiveMessageThread.
Calls:spawnChild()

public Thread receiveMessageThreadStop()
Description: stops receiveMessageThread()
precondition:none
postcondition:none. receiveMessageThread will be terminated.
calls:restartThread()

public void run()
Description: runs the main program of Commander Agent
Precondition:none
Postcondition:none. The agent will be started.
Calls:awaitInstruction()

public void mainMethod(Thread runner)
Description: main thread for the CommanderAgent
precondition:none
postcondition:
calls:

private boolean requireSnapshot(String sentinelId, String backupId)
Description: this is called when the snapshot is required to restart the UserProgram
Precondition:sentinelId is the Id of the sentinel agent backupId is the id of the backup Agent which has the snapshot
Postcondition: returns true if the "search_snapshot" message is sent to the backup agent with the given id. returns false if not.
Calls:talk()

private boolean restartUserProg(String sentinelId)

Description: restarts the user program
Precondition:sentinelId is the Id of the sentinel which restarts the corresponding user program
Postcondition:returns true if the restart request. message is sent to the sentinel id. returns false if not.
Calls:talk()

public void awaitInstruction()
Description: waits instruction(message)
Precondition:none
Postcondition:returns none.
Calls:retrieveNextMessage(), respondToMessage()

public void respondToMessage(UWMessage message)
Description: takes an action according to the message content
Precondition:message is a UWMessage sent by other agents
Postcondition:returns none.
Calls:receiveSendSnapshotMessage(), receiveLocationMessage()
endOfUserProgMessage()

public void receiveSendSnapshotMessage(UWMessage message)
Description:
Precondition:message is a UWMessage sent by other agents
Postcondition:returns none.
Calls:searchCandidateHost(),retrieveAgent()

public void receiveLocationMessage(UWMessage message)
Description:
Precondition:message is a UWMessage sent by other agents
Postcondition:returns none.
Calls:searchCandidateHost(),retrieveAgent()

private String getParentBackupAgentId(List idList)
Description: determines the parents of the backup agents
Precondition:idList is the list of back up agents
Postcondition:returns String which is the id of the parents of backup agents.
Calls:none

private void endOfUserProgMessage(UWMessage message)
Description: sends the end of user program message to the agents
Precondition:message is a user program end message
Postcondition:returns none.
Calls:getBackupAgentId()

2 Sentinel Agent

2.1 Overview

Sentinel agents are spawned by Commander Agent. Each sentinel agent migrates to a different computing node where it starts a corresponding user program wrapper. Each sentinel agent monitors a user process and collects the results sending snapshot to its corresponding Backup Agent.

2.2 Design sheet

a. **Summary**

Sentinel Agent is initialized in **Costructor** and **init**. In **init**, **work** which is a initialization helper function is called(*receiveMessageThread* is started inside).
Work also calls **mainMethod**.
In **mainMethod**, all threads(*receiveMessageThread*, *receiveSnapshotThread*, *checkResourceThread*, *funcMethodThread*) are stopped and **sendEndMessage** is called to send an end message to the Commander Agent.
The **run** method is used to start the threads of the Sentinel Agent. The Sentinel Agent waits for the instruction(message) by calling **awaitInstruction**. The *receiveSnapshotThread* receive snapshot by using **upw.receiveSnapshot** function, and sends the snapshot to the backup Agents which are in the *backupAgentId* list. The checkResourceThread is used to check resourses; however, it is not implemented yet.

Message Type List

restart_userProgram
-The restarting request received from the Commander Agent

b. **Methods and Constructor List**

public SentinelAgent(String[] args)
Description: Sentinel Agent Constructor
Precondition:args is a string array which contains NumberOfTriplets myRank ip - [[rank ip {-|gatewayIp] ...] ProgName [args ...]
Postcondition:userProgWrapperArgs stores arguments

public SentinelAgent()
Description: Default Constructor -nothing happens
Precondition:none
Postcondition:none
Calls:none

public void init()
Description: init function
Precondition:none
Postcondition:ancestorID is saved in sentinelAgentIds
Calls:AgentUtil.initialLocation(this), work()

public void work()
Description: work function

Precondition:none
Postcndition:receiveMessageThread is started
Calls:mainMethod()

public void mainMethod()
Description: main Method function:
Precondition:none
Postcndition:funcMethodThread is started
Calls:displayAgentsInfo(), stopSubThreads(), AgentUtil.sendEndMessage(this)

public void stopSubThreads()
Description: stops sub threads
Precondition: none
Postcondition:receiveMessageThread and checkResourceThread are stopped.
Calls:receiveMessageThreadStop() and checkResourceThreadStop()

public void move(String destHost)
Description: moves to a new host
Precondition: destHost is a string which indicates the new location(host name)
Postcondition:Sentinel Agent is hopped to a new host
Calls:receiveMessageThreadStop() and checkResourceThreadStop()

private void sendSnapshot(Hashtable snapHash, String destId)
Description: sends the file shown by fileName to the BackupAgent indicated by destId
Precondition:snapHash is a Hashtable which indicates the snapshot to be sent. destId is a string which indicates the id of the destination backup agent.
Postcondition:the bakup agent at the destination receives the snapHash.
Calls: sendSnapshotS2B();

public void run()
Description: runs the current thread
Precondition:none
Postcondition:if this is called from receiveMessageThread, then awaitInstruction() is called to wait for the instruction
if this is called from receiveSnapshotThread, then receiveSnapshot() is called to get the snapshot
if this is called from checkResourceThread() then checkResource() is called to check the resource.
Calls awaitInstruction(), recieveSnapshot(), checkResource()

public void awaitInstruction()
Description: waits for instructions as long as receiveMessageThread is alive
Precondition:none
Postcondition:Message is received if there are any.

Calls:retrieveNextMessage(), AgentUtil.respondToMessage(),respondToMesage()

public void respondToMessage(UWMessage message)
Description: takes action according to the message
Precondition: message is a UWMessage
Postcondition:if message has backupAgent_ID then, BackupAgentId is set to the ID contained in messageText[0].
if message has backupAgnet_ID_IP then backupAgentIdIp is set to the IP contained in messagetText[0]
if message has userProgram name, then,funcMethodStop() is called.
Calls: funcMethodStop()

public void receiveSnapshot()
Description: recieves snapshot
Precondition: none
Postcondition:
Calls:upw.receiveSnapShot(), sendSnapShot(), setSnapshotHash()

public void checkResource()
Description: checks if the resource requirements are met in the current node.
<u>This function is to be implemented in the future.</u>
Precondition:none
Postcondition:none
Calls:none

public void funcMethod()
Description: creates a user program object and a userProgWrapper object. starts receiveSnapshotThread and function scheduler.
Precondition:none
Postcondition:a user program object is created. a userProgWrapper object is created. receiveSnapshotThread and the function scheduler are started.
Calls:upw.funcSch(userProgWrapperArgs)

public void funcMethodStop()
Description: terminates funcMethodThread
Precondition:none
Postcondition:funcMethodThread is stopped.
Calls:upw.funcOut()

public void receiveMessageThreadStop()
Description: terminates receiveMessageThread
Precondition:none
Postcondition:receiveMessageThread is stopped.
Calls:restartThread()

public void receiveSnapshotThreadStop()

Description: terminates receiveSnapshotThread
Precondition:none
Postcondition:receiveSnapshotThread is stopped.
Calls:upw.restartThread()

public void checkResourceThreadStop()
Description: terminates checkResourceThread
Precondition:none
Postcondition:checkResourceThread is stopped.
Calls:checkResourceThread.interrupt(),checkResourceThread.join()

public void setBackupAgentId(UWMessage message)
Description: sets the backupAgentId as the one in the message
Precondition:message is a UWMessage
Postcondition:backupAgentID is set as messHead[1]
calls: none

public void setBackupAgentIdIp(UWMessage message)
Description: saves the backupID and its corresponding Ip address
Precondition:message is a UWMessage
Postcondition:backupAgentIdIp is set as messHead[1]
Calls:registerAgentLocation(backupAgentId, backupIp);

3 Backup Agent or Bookkeeper Agents

3.1 Overview

Bookkeeper Agent maintain execution snapshots received from a corresponding sentinel agent. Bookkeeper Agents should migrate to a node where the corresponding sentinel agent exists to prevent the both agents getting killed simultaneously.

3.2 Design sheet

a. **Summary**
The first Backup Agent (agent #2) is spawned by and receives multiple lists of candidate sites from the Commander Agent in **Constructor**. **Init** method is called at first. In **init**, *receiveMessageThread* is started to receive message by calling the **receiveMessageThread** method.

In the **mainThreadForParent** method called in the **ParentWork** method, the first Backup Agent (agent #2) spawns(the **spawn** method is used) child Backup Agent according to the *initialCandidateHosts* and sends a new list of *backupAgentIds* to Commander Agent.

The **run** method is used to start the Backup Agent. The Backup Agent waits for the instructions given by the message from other agents by calling **awaitInstruction** method. If the message is received, the message is examined by calling the **respondToMessage** method.

Message Type List
snapshot_backup_attached

forward_snapshot

search_snapshot
-the snapshot request received from the Commander Agent. This is needed to restart the user program.

retrieve_snapshot

agents_location2

agents_location3

kill_agent

kill_agent2
Terminates itself

Comments:
Resource Checking is not implemented yet.

b. **Method and Constructor List**

public BackupAgent(String[] args)
Description: Constructor for a parent agent
Precondition:args are the list for the hosts of backup agents
Postcondition:initialCandidateHosts is set to args

public BackupAgent()
Description: Constructor for a child agent. initialCandiateHosts is set to null
Precondition:none
Postcondtion:initialCandiateHOsts is set to null

public void init()
Description: init method The init method is executed when a BackupAgent is spawned / injected at first.
Precondition:none
Postcondition: returns none. if it is a parent Backup Agent, then parentWork() is called. if it is a child Backup agent, then childWork() is called.
Calls: parentWork(), childWork()

public void parentWork()

Description: work method for parent. At first receiveMessageThread() is executed. Next child agents is spawned by spawn() method. Finally, mainThreadForParent(runner) method is executed, where runner is return value of receiveMessageThread() method.
Precondition:none
Postcondition:runner(receiveMessageThread) is started
Calls:receiveMessageThread(), mainThreadForParent()

public void childWork()
Description: work method for parent. At first receiveMessageThread() is executed. Next child agents is spawned by spawn() method. Finally, mainThreadForChild(runner) method is executed, where runner is return value of receiveMessageThread() method.
Precondition:none
Postcondition:runner(receiveMessageThread) is started
Calls:receiveMessageThread(), mainThreadForChild()

public Thread receiveMessageThread()
Description: starts receiveMessageThread()
Precondition:none
Postcondition:returns a thread for receiveMessageThread
Calls:receiveMessageThread(), mainThreadForChild()

private void spawn()
Description: spawns a child thread at each candidate hosts
Precondition:none
Postcondition:returns a thread for receiveMessageThread
Calls:spawnChildAtPlace()

public void spawnChildAtPlace(String destHost)
Description: spawns a child at a particular host
Precondition: String destHost is a candidate host name where a child thread should be spawned.
Postcondition: child thread is spawned at destHost. if it is not there, childId will be saved in the backupAgentIds list.
Calls:spawnChild(), uwa.getAgentId();

public void mainThreadForParent (Thread runner)
Description: main thread for parent
Precondition:runner is a thread
Postcondition:returns none. Runner(receiveMessageThread) is stopped.
Calls: AgentUtil.childrenNumInDir(), AgentUtil.sendAgentsLocations(). receiveMessageThreadStop()

public void mainThreadForChild (Thread runner)
Description: main thread for child
Precondition:runner is a receiveMessageThread.
Postcondition:receiveMessageThread will be terminated.

private void move(Thread runner)
Description: moves(migrates) to a different location
Precondition:runner is a message receive thread
Postcondition:threadContinue is set to false. Runner is stopped. hop to a new location(mnode4)
Calls: hop()

private void move(Thread runner, String destHost)
Description: moves(migrates) to the destination
Precondition:runner is a message receive thread, desthost is the name of the destination host.
Postcondition:threadContinue is set to false. Runner is stopped. hop to a new location
Calls: hop()

public void receiveMessageThreadStop(Thread runner)
Description: terminates the thread runner(receiveMessageThread)
Precondition:runner is a receiveMessageThread
Postcondition:threadContinue=false, runner.join()
Calls: none

public void run()
Description: run method wait for the instruction by the message
Precondition:none
Postcondition:none
Calls: awaitInstruction();

public void awaitInstruction()
Description: wait and receive messages. get hostnames of the child thread location
Precondition:none
Postcondition:none
Calls:respondToMessage()

public void respondToMessage(UWMessage message)//////////////???????????????///
Description: responds to the message received from other agents
Precondntion:none
Postcondition:output to the screen
Calls:

private void resSnapshotBackupAttached(UWMessage message)
Description: receive and stores the snapshot
Precondition:message is a UWMessage which contains back up data to be stored
Postcondition:none
Calls: receiveSnapshot(message)

private void receiveSnapshot(UWMessage message)
Description: receives the snapshot and stores the snapshot. forwards the snapshot message if needed
Precondition: message is a UWMessage contains the snapshot information
Postcondtion: if the sendingID in the snapshot message is in the list of backupAgentIds
Calls: message.getSendingAgentId(), saveSnapshot(UWMessage message),

private void saveSnapshot(UWMessage message)
Description: save snapshot
Precondition:message is a UWMessage
Postcondition:
calls:saveSnapshot

private void forwardSnapshot(UWMessage message, String forwardedId)
Description: forward snapshot to the
Precondition:message to be forwarded, forwardedId for the receiver's host id
Postcondition:message is forwarded to the forwadedID
Calls: talk()

private boolean searchSnapshot(UWMessage message)
Description: finds the snapshot according the inquiry by Sentinel Agent or Backup Agent
Precondition:message is a UWMessage which is a inquery about snapshot by Sentinel Agent or Backup Agent
Postcondition:if found, returns true, if not, returns false

private boolean retrieveSnapshot(UWMessage message)
Description: receives and sends the message to the destination
Precondition: message is the retrieve snapshot request received from the sentinel agents
Postcondition:if message header is not null or length is bigger than 2 then snapshot is sent to messHead[2]. if not, returns false
Calls: sendSnapshot()

private boolean sendSnapshot(String sentinelId, String destId)
Description: sends a snapshot to the destination
Precondition:sneitnelId is the sentinel Id the backup agent is corresponding to. destId is the id of the receiver of snapshot.
Postcondition:send snapshot of sentinelID to the destId

private void sendAgentsLocationMes(UWMessage message)
Description: send backupAgent locations to SentinelAgents and other BackupAgents
Precondition:message is a UWMessage which contains the information of the BackupAgent locations
Postcondition:message is sent to Sentinel Agent and hashtable is sent to BackupAgent
Calls:AgentUtil.mesAgentsLocations(message, this, sentinelAgentIds)
AgentUtil.sendAgentsLocations(this, locationsHash, backupAgentIds)

private void killAgentMessage(UWMessage message)
Description: terminates the main thread
Precondition: message is a UWMessage which indicates the message to terminate the main thread
Postcondition: cont is set to false and end message is sent
Calls:AgentUtil.sendEndMessage(this, backupAgentIds)

private void killAgent2Message(UWMessage message)
Description: terminates the thread?
precondition: message is a UWMessage which indicates the message to terminate the main thread.
Postcondition:cont is set to false

4. Other Agent

I/O Agent is not available at this point, but it will be. I/O Agent manages Input/Output of the user program.