*International journal of scientific and technical research in engineering (IJSTRE)*
*www.ijstre.com Volume 1 Issue 1 ‖ April 2016.*

# Motivation to a Deadlock Detection in Mobile Agents with Pseudo-Code

Rashmi Priya

***Abstract*** *: The solution presented locates locality of reference during the deadlock detectionprocess by migrating detector agents to query multiple blocked agents.To message each blocked agent individually and gather theirresponses at the shadow agent itself is an alternative to this single migration.Thepseudo code provides a context for the solution and insightinto the responsibilities and activities performed by each entity.*

***Keywords****:Deadlock, Agents, pseudo code, detector, entity*

## I. Introduction

As presented in the previous section traditional distributed solutions commonly have fault and location assumptions that make them unsuitable for mobile agent systems. To solve this problem, mobile agent specific solutions are required. The properties of the presented deadlock detection algorithm illustrate how it is a fully adapted mobile agent solution.

The presented technique is fault tolerant and robust. Lost agents or messages during the deadlock detection process do not represent a critical failure. This fault tolerance is due to three properties of the algorithm: the autonomous nature of the agents, the periodic nature of the detection process and the copying of deadlock information. Shadow, deadlock detection and consumer agents execute asynchronously .They do not depend on continual communication during the deadlock detection process. The algorithm is designed around incremental construction of the global wait-for graph. Finally therefore if a portion of the graph is lost, the next update will recover that information. Hence copying of the partial wait-for graph into deadlock detection agents make the loss orfailure of a particular deadlock detection agent trivial and has no impact on the detectionprocess, outside of slowing the process. Additional safeguards can be built into the agent hosts. such as agent crash detection, to improve fault tolerance

## II. Algorithm Motivation and Agent Properties

Bylimiting the number of messages that would be required in other solutionsthe Detector migration reduces network load. It is difficultto compare the network load of this mobile agent solution to that generated in traditionaldistnbuted deadlock detection solutions due to the significantly different paradigm andproperties of the environment.This is due to the parallel /distributed nature of thetechnique, which enforces the lack of a central point of messaging and coordination.This reduces the risk of flash congestion and allows the technique to handle deadlockinvolving many blocked agents.

The load is spreadacross many host environments, if the network load of the presented solution is considered as a whole.

Additionally, networkorganization independence is guaranteed through a clear separation of mobile agents from the mechanics of routing and migration, the agents are not aware of the number of hosts in the mobile agent system and do not have explicit knowledge of resource locations. It should be noted that even though the solution is network independent, the topology is static once the algorithm begins. If the topology is allowed to change, a dynarnic topology update protocol must execute in the background to provide new routes to the hosts.

A common use of mobile agents is to encapsulate complex protocols and interactions [24]. This technique uses the combination of shadow agents and deadlock detection agents to encapsulate a complex series of probes, interactions and acknowledgments.

Additionally. these protocols are isolated from the consumer agent; therefore, can be easily modified and upgraded. The deadlock detection phase could be implemented as remote procedure calls or another fom of distributed programming, but would require network organization assumptions and the continual exchange of messages. Detector and shadow agents cary out their deadlock detection tasks in an asynchronous manner. They coordinate their efforts in defined ways, but are able to keep working without regular contact and do not require constant supervision while carrying out tasks. This asynchronous and autonomous operation contributes to the previously discussed fault tolerance.. For examplethe combination of consumer, shadow and detector agents

adapt to their environment tosolve deadlock situations, shadow agents react independently to changingnetwork conditions and the state of their target consumer agent to initiate the deadlock
detection processing. Similarly the separation of the implernentation from facilities specific to a particular mobile agentsystem or operating system detector agents can react to network failures or therequests of other agents while gathering global wait-for graph information.
allows the solution to execute in a heterogeneousenvironment. Moreover the separation of replica and detector agents from the consuming agents they monitor, allows them to be adapted to many different environments without (or with minor) modifications to the entities performing the work.

## III. Deadlock Detection Pseudocode

This pseudo code provides a context for the solution and insightinto the responsibilities and activities performed by each entity. This section presents pseudo-code of each element that plays a significant role in thepresented solution.
First, pseudo-code forthe consumer, shadow and detection agent is presented. Finally, code for the mobileagent environment is presented.

3.1 Agent A

```
public class AgentA extends MobileAgent
{
public AgentA( String int heartbeat )
{
state = IDLE;
}
public void run()

{
while( true )
{
messages = getMessagesFromBlackboard( agentId );
processMessages( messages );
switch ( state )
{
case IDLE:
case WAITING:
// do nothing
break;
case MOVING:
if( currentHost is not targetEnvironment )
{
postRouteRequest( targetEnvironment ) ;
{
else
{
// Made it!
state = IDLE;
{
break;
{
sleep ( heartbeatDelay ) ;
\
{
private Vector processMessages( messages )
{
while ( more Messages )
if message was accepted remove from list;
return unprocessed messages;
{
```

```
private boolean processMessaget BlackboardEntry msg )
if ( message equals amove" AND state is IDLE )
{
targetEnvironment = get terget from message;
state = MOVING;
/ / lets ask our current environment to route us
postBlackboardMsg( "route”,
targetEnvironment ) ;
{
else if( message eqyals "lockw AND
state is IDLE OR WAITING )
{
extract lock type and resource from message ;
lockResource( lockType, resource );
if( message equals
{
ext ract resource f rom message;
unlockResource( resource );
{
{// locked. the AgentEnvironment should have created
// a shadow agent and placed us under it's watchful
// eye.
// Special case. if the resource we just locked was
// the same as a resource we were blocked on, it
// means the Environment notified us and we should
// move into the idle state

AND state
private void lockResource( String locktype,
String resourceName )
{
if( lockType equals  “exclusive” )
{
get resource manager;
if resourceManager.lockResource( resourceName,
lockType ) succeeds
{
if( resourceName equals blockedResourceName )
state = IDLE;
{
{
else
/ / locked failed .. time to block
state = WAITING;
blockedResourceName = resourceName;
// Notify our gracious host ...
postBlackboardMessage(“agentBlock”, resourceName );
private void unlockResource( String resourceName )
{
get Resource Manager
resourceManager.unlockResource(resourceName);
}}
```

3.2 Agent B

```
public class ReplicaAgent extends MobileAgent
(
public ReplicaAgent ( String id, String targetAgent, int heartbeat )
```

```
{
state = IDLE;
targetAgentName = targetAgent;
reset locked resource list
reset detection info table
reset detector agent
{
public void run()
while( true )
{
// Check to see if Our detector is dead
checkForDetectorDeath () ;
if ( state is not MOVING )
{
messages = getMessagesFrom BlackBoard();

messages =processMessages(messages);
{

switch ( state
{
case IDLE:
break;
get~messagesFromBlackboard();
processMessages( messages );
case MOVING:
if( currentHost is not targetEnviroment ) )
{
routeRequest( targetEnvironment );
{
else
{
state = IDLE;
}
break;
}
sleep ( heartbeatDelay ;
}
}
private Vector processMessages( messages )
{
while( more Messages )
{
processMessage( currentMessage );
if message processed remove from list;
{
return unprocessed messages;
{
private boolean processMessage( BlackboardEntry msg )
{
Vector attachments = msg.getAttachments();
if( message equals "move" AND state is IDLE )
{
String target = extract target from message;
targetEnvironment = target;
state = MOVfNG;
{
else if( message equals “addLock” )
{
```

```
addlock( attachment #l,
attachent #2,
attachment #3,
attachent #4 );
retVal = true;
else if( message equals  “removeLock” )
{
removelock ( attachment #I, attachment #2 ) ;
state = IDLE;
retVal = true;
{
else if( message equals ablockedm ) )
{
blockedTarget( attachment #l 1;
retVal = true;
{
else if( messge equals wunblockedw ) )
{
unblockedTarget ( ) ;
retVal = true;
{
else if( message equals “deadlockReport” ) )
{
processReturnOfDetector( attachment#1, attachment #2, attachment #3 ) ;
retVal = true;
{
else if( message equals “deadlockInfoRequest” ) 1
retVal = true;
else
{
// We dontt understand this message, but our superclass
// might have some good ideas ...
retVal = super.processMessage( msg );

{
{
public void exit ()
{
if( detector )
{
Remove ( detector from host environment);
{
super. exit () ;
private void addlock( String environment,String resource,String owner,int priority1 )
(// Check  to see if we contain  this  lock already)
if( resource not already locked )
{
new   r e s o u r c e I n f o ( env, res, owner, p r i o r i t y1 ) ;
s t o r e resourceInfo  i n  locked  resource list
}
}
Private  void  removeLock( S t r i n g env1, S t r i n g resourceName1 )
{
i f ( resourceName1 is i n locked resource list )
{
remove resource from locked resource list
{
{
privatevoid unblockedTarget1 ( )
```

```
{
//  unblocked target
s t a t e = IDLE;
{
private void blockedTarget( Agent blockedAgent,
S t r i n g (resourceName )
{
State1= TARGETBLOCKED;
owner = query host environment for owner of resource;
blockedResourceName = resourceName;
localAgents = query host environment about l o c a l agents;
i f ( owner inlocalAgents )

Table .put ( targetAgentName, new DetectionInfo( .. ) );
// 1) Create  agent
Detector1 = new DetectorAgentO;
// 2) Put the agent
postBlackboardMsg(  d e t e c t o r );
// 3) agent start
postBlacKboardMsg( buildDetectorLocks () ) ;
numOfDetectionStarts++;
1astDetectionStartTirne = current  time ;
{
{
p r i v a t e void processReturnOfDetector( DetectorAgent agent )
(// Our d e t e c t o r is back, let's see what's new)
switch( state )
{
case TARGET-BLOCKED :
i f ( checkForDeadock( agent.getDetectionTables0 ) )
(
// We have a deadlock, b e t t e r resolve it.
resolveDeadlock( agent ) ;
reset detectionInfoTable;
{
case IDLE:
// Our t a r g e t unblocked i f t h i s is t h e case ...
// let's k i l l t h e d e t e c t o r ...
removeDetector () ;
(
// The state w i l l have changed t o waiting f o r unlock
// i f t h e deadlock check succeeded
s w i t c h ( state )
(
case TARGET-BLOCKED:
// reset and restart t h e d e t e c t o r
postBlackboardMsg( " s t a r t " , buildDetectorLocks() ); numOfDetectionStarts++; )
1astDetectionStartTime = c u r r e n t t i m e ;
break;
case WAITING,FOR~UNLOCK:
// W e are breaking the deadlock don*t s t a r t
// any new p r o c e s s i n g .
break;
{
{
p r i v a t e boolean checkForDeadlock( Vector detectionTableList )
{
/ / - If we f i n d t h e resource t h a t our master is blocked
// on i n t h e r e t u r n e d lock list ... we have a
```

```
// deadlock
/ / - If we don't f i n d t h e resource, j u s t add the locks
// to our g l o b a l list.
// L e t ' s go through t h e returned t a b l e .. e n t r y by e n t r y
// and add it t o our d e t e c t i o n t a b l e .
while ( d e t e c t i o n T a b l e L i s t has more e n t r i e s )
{
d e t e c t ionTable = c u r r e n t d e t e c t i o n t a b l e ;
a g e n t L i s t = get agent list from d e t e c t i o n Ta b l e ;
while( a g e n t L i s t has more e n t r i e s )
{
d e t e c t i o n I n f o = d e t e c t i o n i n f o from c u r r e n t t a b l e
r e l a t e d t o c u r r e n t agent;
add d e t e c t i o n I n f o t o g l o b a l table;
i f ( c u r r e n t agent narne e q u a l s targetAgentName )
{
deadlockEound = True;
{
{
{
r e t u r n deadlockFound;
{
p r i v a t e void resolveDeadlock( DetectorAgent agent )
{
// Build a list of t h e resources involved i n t h e cycle ...
if( s t a t e is TARGETBLOCKED )
{
// So we found a deadlock .. t h e question is are
// we t h e one to break it ?
// Find t h e Cycle ...
cycleList = findElementsInCycle( detectionInfoTable ) ;
while( cycleList has more elements )
{
/ / f i n d r e s o u r c e w i t h lowest p r i o r i t y
1ockToBreak = lowest p r i o r i t y resource;
{
i f ( 1ockToBreak equals resource we are blocked on )
{
// L e t ' s send our d e t e c t o r off on h i s mission
// t o unlock t h e resource ..
// But f i r s t we b e t t e r set him up with the
// correct information t o survive t h e
// d e s t i n a t i o n ResourceManager*~ interogation.
// 3) s t a r t t h e d e t e c t o r agent
postBlackboardMsg( "unlock",
LockToBreak ) ;
s t a t e = WAITING-FOR-UNLOCK;
{
{
private Vector findcycle ( Hashtable detectionInfoTable )
{
Vector cycleVector = new V e c t o r ( ) ;
cyclevector add( resource we are blocked on ) ;
info = find entry i n detectionInfoTable whose primary
lock is t h e c u r r e n t resource;
// Loop u n t i l we f i n d t h e e n t r y f o r our agent
while( c u r r e n t e n t r y ' s agent name i s n r t equal t o
our t a r g e t agent )
{
```

```
// Walk up the tree t o t h e parent node.
i n f o = f i n d entry i n detectionInfoTable whose primary
lock is the c u r r e n t resource;

cyclevector add( info );
{
return cyclevector;
{
private void checkForDetectorDeath()
{
Date currentTime = current time;
BlackboardEntry msg;
if( num0fDetectionStarts >O AND ( state is TARGETBLOCKED OR  state is WAITING-FOR,UNLOCK ) {
{
// We have a dead detector
// 1) Create a detector agent
detector = new DetectorAgent 0;
// 2) Inject the agent
postBlackboardMsg ( "inject", detector ) ;

if( state is TARGET-BLOCKED )
{
// 3) start the agent
postBlackboardMsg( "start", buildDetectorLocks() );
{
else if( state is WAITING-FOR-UNLOCK )
resolveDeadlock( detector.getIdentifier());
detector.getToken() );
{
1astDetectionStartTime = current time;
}
}
}
}
```

3.3 Agent C

```
public class AgentC extends MobileAgent
{
public  AgentC( String id, int heartbeat,
ShadowAgent parent )
{
reset detection Table List;
reset resources To Vist;
reset targetEnvironment;
reset targetResource;
state = IDLE;
set parent = parent;
{
public void run ()
{
while ( true )
{
if ( state is not MOVING )
{
messages = getMessagesfromBlackboard () ;
messages = processMessages( messages );
```

```
}
switch ( state)
getMessagesfromBlackboard () ;
processMessages( messages );
(
case IDLE:
break;
case MOVING:
if( currentHost is not targetEnvironment ) )
{
else
{
state = CHECKING-LOCKS;
{
break;
if( current Host is not targetEnvironment )
{
if( host.unlockResource( targetResource, agentToNotify )
{
( state = RETURN-FROM-UNLOCK);
{
else
state = IDLE;
{
case RETURNRNFROM, the LOCK:
if( currentHost is not startingEnvironment )
(
Shadow,removeLock( targetEnvironment, targetResource );
state = IDLE;
{
case DONE:
if( currenthost is not startingEnvironment )

(
state = REPORT-RESULTS;
)
case CHECKING-LOCKS:
checklocks ( ) ;
break;
case REPORT-RESULTS:
postMessageToBlackboard( shadowÀgent, deadlockInfo );
break;
}
sleep( heartbeatDelay ) ;
private Vector processMessages( messages )
{
while ( more Messages )
{
processMessage( currentMessage );
if message processed remove from list;
{
return unprocessed messages;
{
private boolean processMessage( BlackboardEntry rnsg )
{
attachments = msg,getAttachments();
if( message equals  “startW ”)
{
startDetection ( (Vector) attachment  ;
```

```
retVal = true;
else if( message equals "unlock" ) {
(
startunlock ( attachment #1,
attachment #2,
attachent #3 ) ;
retVal = true;
message ;
deadlockRequestResponse ( attachment #l ) ;
retVal = true;
{
else
{
super.processMessage( msg 1;
{
return retVal;
{
private void startDetection( resources )
( setVisitlist( resources ));
targetEnvironment( entry.getEnvlame() };
targetResource ( entry. getResName () ) ;
// Reset the table ...
detectionTablelist( new VectorO 1;
start ingEnvironment ( getHost () . getName () ) ;
state ( MOVING ) ;
{
private void checklocks ()
{
while( shadowlist has more elements )
count expected responses;
{
if( expected responses >0)
(
state = WAITING-FOR-RESPONSE;
{
else
{
findNewTarget () ;
}
private void deadlockRequestResponse( newTable )
{
shadowList = query current host for agents blocked on
the resource we are visiting;
expectedResponses--;
detectionTablelist.add( newTable );
if( al1 expectertesponses received )
{
findNewTarget 0 ;
}
}
private void findNewTarget0
{
if( more resource to visit )
get next resource;
// Let's get started ...
targetEnvironment = entry.getEnvName();
targetResource = entry.getResName();
{
else
```

```
{
// Tirne to head home ..
targetEnvironment = startingEnvironment;
state = DONE;

}
}
}
```

Host Environment

```
public class AgentEnvironment extends Thread
public AgentEnvironment( String name, int id, int 1oggingLevel )
{
resourceManager = new ResourceManagerO;
topologyManager = new TopologyManager();
reset agentTable;
reset messageBoard;
reset blockedAgentTable;
globalIdentifier = id;
state = PROCESSING;
}
// Global Identifier can be used as the priority

public void run ()
(
while( true )
{
checkEorMessages () ;
updateRoutes () ;
sleep( 1000 );
}
)
public synchronized void agentEnter( Agent newAgent )
{
if( state is PROCESSING )
{
agentTable.put( newAgent ) ;
newAgent.enter0;
}
}
private synchronized void agentExit( Agent 1eavlngAgent
{
if ( state is PROCESSING )
(

1eavingAgent. exit ( ) ;
)
}
private void agentBlock( Agent blockedAgent, String resourceName)
(
// Look for a replica  agent ...
replica= find replica agent for blockedAgent;
if ( shadow found )
{
...
postBlackboardMsg( "blockedAgent", resourceName ) ;
private void checkForMessages()
```

```
{
get messages from blackboard;
while ( more messages)
{
processMessage( current message );
}
}
private void processMessage( BlackboardEntry msg
attachments = rnsg.getAttachments0;
if( message equals “pause” ) )
state( PAUSED ) ;

else if( message equals wresumen ) )
(
state( PROCESSING ) ;
message equals
(
agentBlock ( msg . getAgent Id () , attachment t1 ,attachment t2);
if( message equals 0)
{
routeRequest(msg.getAgentId(), attachment # l ,attachment #2);
{
else if (( message equals *inject)
this.injectAgent( attachment)

else if( message equals "remove" ) )
{
removeAgent ( attachrnent #l ) ;
}
}
private boolean routeRequest( String movingAgent, EnvironmentToken token,
String targetEnv )
{
if( state() is PROCESSING )
{
movingAgent = get moving agent from agent tables;
(
return true;
)
if( check for shadow information in the token )
shadowAgentId = get shadow name from token;
If ( check for shadow agent in agent tables )
{
shadow = get shadow agent from agent tables;
}
else
{
retVal = f alse;
}
}
if ( retVal is true )
(
AgentEnvironment env = request route from
topologyManager;
if ( env is not nuil )
(
suspendAgent( movingAgent );
agentExit( movingAgent 1;
agentlable () . remove ( movingAgent Id ) ;
```

```
env.agentEnter( movingAgent );
if ( shadow is not nul1 )
{
suspendAgent ( shadow ) ;
agentExit( shadow );
agentlable () . remove ( shadow ) ;
env.agentEnter ( shadow ) ;
{
Else
{
retVal = false;
}
}
}
else
(
retVal = false;
)
r e t u r n retVal;
1
public Vector getBlockedAgents( String resourceName )
I
return list of agents blocked on resourceName;
{
public synchronized void postMessage( String agentId,
String message )
{
add message for agentId to the message lists;
{
public synchronized Enurneration getMessages( String agentId )
{
return messages for agentId;
{
private injectAgent( Agent newAgent )
{
newAgent . start () ;
agentTable.put( newAgent );
private void removeAgent( String agentName 1
}
agentTable.remove( agentName ) ;
}
private void updateRoutes()

}}}
```

## IV. Conclusion

The presented algorithm is designedwith the unique properties and challenges of mobile agent systems as a motivating factor. As a result, the solution has some of the properties and features that are comrnonly found in mobile agent implementations. This section lists the properties of the proposed algorithm which make it a mobile agent solution. The solution is network organization independent. The algorithm makes no assumptions concerning network topology (i.e., ring). the numberof hosts or node locations to support the solution. Resource-based routing and tracking of the nodes visited by a particular agent elirninate the need for explicit topology knowledge

## References:

[1] Walsh, T., Paciorek, N. and Wong, D. "Security and Reliability inConcordia.", Appeared in *Mobility: Processes, Computer and Agents*, Addison-Wesley, Reading, 1999.

[2] Mitsubishi Electric Information Technology Center. "Concordia - Java Mobile Agent Technology." *World Wide Web,* January 2000, http://www.meitcacom/HSUProjects/Concordia/

[3] University ofStuttgart. "TheHomeof the Mole."*WorldWideWeb*,September 2000, httJ)://mole.infonnatik.uni-stuttgart.de/

[4] University of Tromse and Cornell University. "TACOMA - Operating System Support For Mobile Agents." *World Wide Web.* August 1999. http://www.tacoma.cs.uit.no/

[5] ObjectSpace Inc. "ObjectSpace Voyager Core Package Technical Overview.", Appeared in *Mobility: Processes, Computer and Agents*, Addison-Wesley, Reading, 1999.

[6] ObjectSpace Inc. "ObjectSpace Product Information: Voyager." *WorldWideWeb.* September 2000.htm://www.objectspace.com/products/voyager/

[7] Fachbereich lnfonnatik and Johann-Wolfgang-Goethe-Universitaet Frankfurt. "ffMain". *World Wide Web.* February 2000. htn>://www.tm. informatik.uni• frankfurt.de/Projekte/MN

[8] University ofGeneva. "TheMessenger Proejct." *WorldWideWeb*.December 1997. http://cui.unige.ch/tios/msgr/

[9] General Magic, Inc. "Odyssey". *World Wide Web.* November 2000. http://www.genmagic.com/

[10] University ofModena. "MARS(Mobile Agent Reactive Space)." *WorldWideWeb.* October 2000.http://sirio.dsi.unimo.it/MOON/MARS/ index.html

[11] CardelliyL.ltMobileComputational Ambients." *WorldWide Web.* September 2000. http://www.luca.demon.co.uk/ Ambit/Ambit.html

[12] TU Berlin andUniversity ofBologna. "ThePageSpace Effort." *WorldWideWeb.* January1997.http://flp.cs.tu-berlin.de/pagespc/

[13] Flanagan, D.*JavainaNutshell*.O'Reilly &Associates, Inc.,Cambridge, 1996.

[14] Horstmann, C. and Cornell, G. *Core Java Volume I - Fundamentals.* Sun MicrosystemsPress,California. 1999.

[15] Sun Microsystems Inc. "Java 2Platform vl.2.2 API Specification." *WorldWide Web*.September 1999.htnz:l/java.sun.comlproducts/jdk/1.2/docs/ api/index.htrnl