QOS MANAGEMENT USING MOBILE AGENTS

PROJECT WORK DONE AT

KUMARAGURU COLLEGE OF TECHNOLOGY

P-1166

PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT
FOR THE AWARD OF
BACHELOR OF ENGINEERING DEGREE IN INFORMATION TECHNOLOGY
OF BHARATHIAR UNIVERSITY, COIMBATORE

SUBMITTED BY

S. FATHIMA	(0027S0073)
K.R. JAIKUMAR	(0027S0079)
R. SRIVIDHYA	(0027S0112)

GUIDED BY

Ms. V. VANITHA M.E., SENIOR LECTURER

Department of Computer Science and Engineering

KUMARAGURU COLLEGE OF TECHNOLOGY
COIMBATORE - 641 006
MARCH 2004

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KUMARAGURU COLLEGE OF TECHNOLOGY
COIMBATORE - 641 006

CERTIFICATE

This is to certify that the project entitled

"QOS MANAGEMENT USING MOBILE AGENTS"

is the bonafide work of

S. FATHIMA	(0027S0073)
K.R. JAIKUMAR	(0027S0079)
R. SRIVIDHYA	(0027S0112)

and submitted in partial fulfillment of the requirement for the award of Bachelor of Engineering Degree in Information Technology of Bharathiar university, Coimbatore

[illegible]

Head of the department
(Dr. S. THANGASAMY)

[illegible] 22/3/04

Guide
(Ms. V.VANITHA)

Submitted for University Examination held on 26th March 2004.

[illegible] 27/3

Internal Examiner

[illegible]

External Examiner

DECLARATION

We,

S. Fathima (0027S0073),

K.R. Jaikumar (0027S0079),

R. Srividhya (0027S0112)

declare that, this project titled "QoS Management using Mobile Agents" was done by us and to the best of our knowledge, so far, a similar work has not been submitted to the Bharathiar university, Coimbatore, or any other institution, as an requirement for the fulfillment of any course of study.

This report is submitted by us in partial fulfillment of the requirement for the award of Bachelor of Engineering Degree in Information Technology of Bharathiar university, Coimbatore.

Place: Coimbatore.

Date: 22nd March 2004.

Fathima [illegible] [illegible]

S. Fathima | K.R. Jaikumar | R. Srividhya

TABLE OF CONTENTS

SYNOPSIS

The Mobile Agent (MA) Technology has raised considerable interest in the research community. It is gaining importance in the distributed management of networks and services for heterogeneous environments. This new emerging technology of agent programming is a challenging opportunity for delivering more flexible services and dealing with network programmability.

Our project is devoted to gain knowledge about Mobile Agents and about QoS management especially in Int-serv and Diff-serv environments; and to use the Mobile Agent Technology in Diff-serv environment.

Int-serv environment deals with per-flow guarantees, while Diff-serv environment deals with relative guarantees for aggregate flows. Based on a distributed programming environment, we adopt the agent programming paradigm to prove its flexibility in aggregated traffic condition through virtual links (diff-serv model). The three main QoS parameters are delay, delay variation (jitter) and packet loss ratio. All these depend on the virtual link between the source and destination. The availability of bandwidth is responsible for the QoS. So negotiating on the amount of resources (here, bandwidth), the link between the source and destination is established. Mobile agents are used to gather information from other systems for the negotiation.

Introduction

1. INTRODUCTION

1.1 EXISTING SYSTEM AND ITS LIMITATIONS

QoS management has been done so far using the negotiation of parameters, resource allocation, monitoring or policing. In some cases, best effort service is used, where the best level of QoS parameters are used, irrespective of the type of data transferred. But in most cases, end-to-end QoS is negotiated, or the best service is provided based on the type of data.

In the case of negotiation, the source and destination systems have a lot of data transfer between them before starting transmission. The source waits until it gets replies from destination, like Client/Server approach.

In the case of resource allocation, the resource information should be obtained from the link to the destination. Here too, the source has to wait until it gets details regarding the link.

In the case of monitoring, the route or link, across which the data is sent should be constantly monitored.

In the case of policing, a lot of information is collected, and the route maps are also defined which are made to match on certain flow criteria and it allows a specific QoS across a link.

All these approaches require a lot of data transmission regarding the status of the link even before the actual transmission starts.

The limitation is that, here, the source should be connected to the destination until the request processing is completed in the destination or across the link and the result is sent back to the source. If the connection fails, all the information is lost.

1.2 PROPOSED APPROACH AND ITS ADVANTAGES

Mobile Agent Technology is an upcoming technology. Considering the benefits of using it for Information Retrieval, we propose the use of Mobile Agents for QoS management, instead of the usual approach, where the source has to wait until it gets information from the destination.

Mobile Agent is an autonomous piece of code that has the unique ability to transport itself from one system in a network to another. When Mobile Agent Technology is used, it acts independently. The ability to travel allows a Mobile Agent to move to a system that contains an object with which the agent wants to interact and then to take advantage of being in the same host or network as the object.

Here, there is no need for the source to be waiting for the results for a long time. Once it dispatches an agent to the destination, the source can continue its work. When the mobile agent finishes collecting details from the destination, it returns back to the source and displays its results. Then the source can start transmission.

Our approach is to send a mobile agent to a destination system. There, it acquires details about the destination system's capacity and priority and comes back to the source.

The reasons for using Mobile Agents are:

- They reduce network load.
- They overcome network latency.
- They encapsulate protocols.
- They execute asynchronously and autonomously.
- They adapt dynamically.
- They are naturally heterogeneous.
- They are robust and fault-tolerant.
- They have the ability to communicate with other agents.
- To work efficiently in low bandwidth connections.

There are several Mobile Agent Platforms available. Here, we use Aglets. Aglets are a special type of Agents developed by IBM Research. Aglets are Java objects that are able to travel from host-to-host (Mobile Agents). Aglets are hosted by an Aglet Server that provides the environment for them to run in. The Java Virtual Machine and a special Aglet security manager ensure safety while receiving and hosting Aglets.

1.3 PROJECT DEFINITION

The project mainly aims at applying Mobile Agent Technology in QoS management in Diff-serv environment.

1.4 PROJECT PLAN

The project can be divided into the following modules:

- Mobile Agent Creation in the source
- Acquiring details from other systems using Mobile Agents
- Calculation the link capacity and notification.

Literature Survey

2. LITERATURE SURVEY

2.1 AVAILABLE IMPLEMENTATIONS

2.1.1 Implementation Of Mobile Agent Technology In Multiparty Event Scheduling

Mobile Agent Technology has been applied in Multiparty Event Scheduling. Many events require the participation of several parties. Prior knowledge of the date when most of the targeted participants are available is often a pre-requisite for scheduling them. Here static agents are used to retrieve their own residing party information while, mobile agents are used to gather information from other parties and both are combined in scheduling.

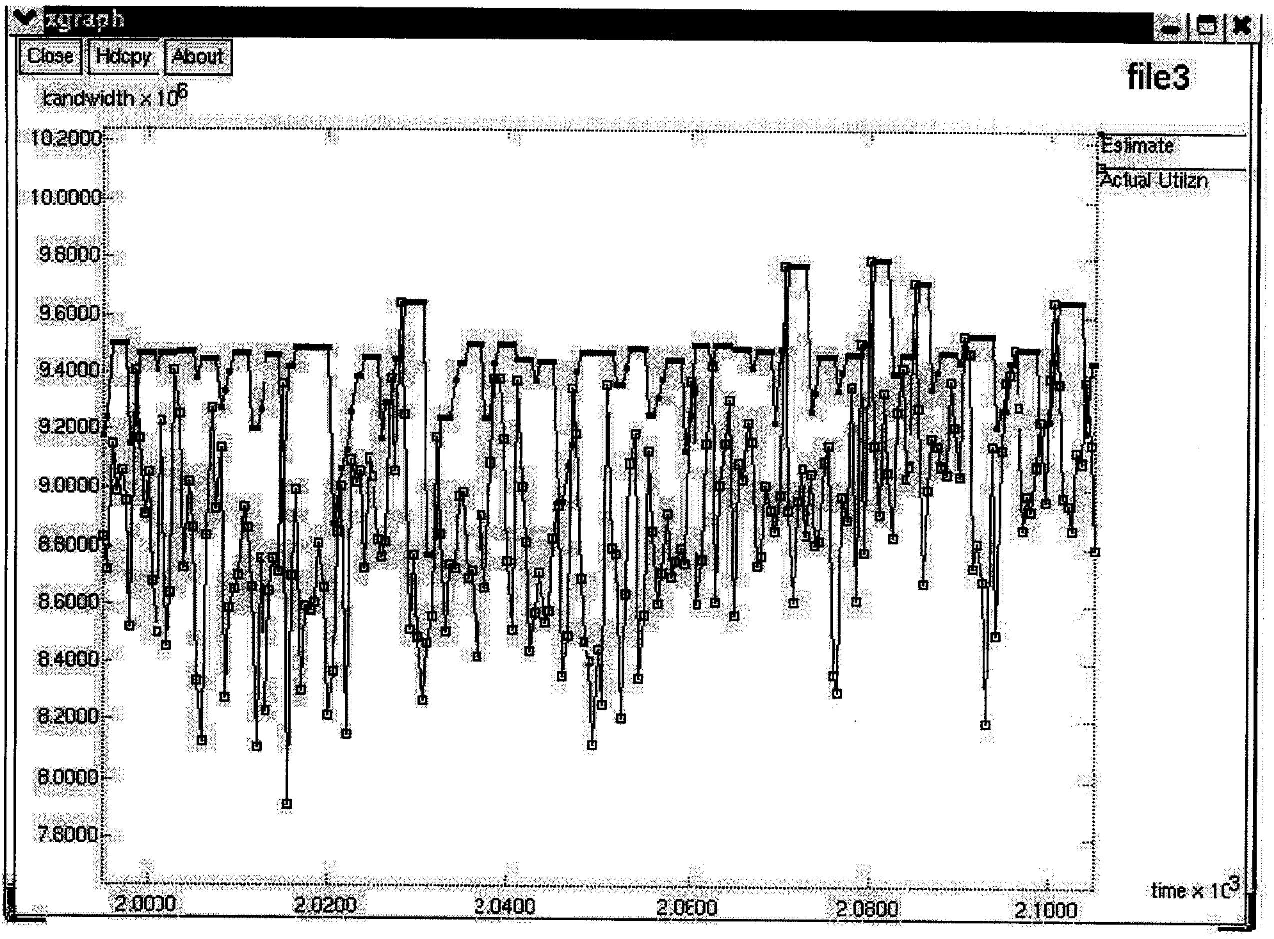
2.1.2 Implementation Of Qos Management Using Map(Mobile Agent Platform) Server

The MAP server has been developed by University of Catania, Italy. Here, the MAP server is used. The applications to be run should be written in Java. It requires Borland Visibroker's Object Request Broker(ORB) to be active (i.e., the osagent process should be running), SSL jar files and extra class files. The recent version of MAP server is 1.4. MAP server should be running in all systems and each system is known as a MAP region. The configuration of this server is complex, though it has several functionalities.

2.1.3 Implementation Of Int-Serv Model In Network Simulator

This netowk simulator is provided by Information Sciences Institute(ISI) of the University of Southern California. Here, the simulation of various models are done. Diff-serv model simulation is not fully developed. In the case of int-serv model, when the user chooses the algorithm and the simulation time, it simulates the utilization of resources based on the algorithm over a period of the specified time. The simulator is developed in TCL scripting language. The recent version of the Network Simulator is ns - 2.27.

An example graphical output of the simulation is shown below. Here, measured sum algorithm is chosen to be simulated in an int-serv environment, over a period of five minutes. After the time, the graph is displayed by passing the output file to xgraph function, which produces a graphical display.



This simulator doesn't implement mobile agent technology.

2.1.4 Implementation Of Int-Serv And Diff-Serv Models Using Jsim

J-Sim (formerly known as JavaSim) is a component-based, compositional simulation environment. It has been built upon the notion of the autonomous component programming model. The model defines the generic structure of a node (either an end host or a router) and the generic network components, both of which can then be used as base classes to implement protocols across various layers. This is used to simulate int-serv and diff-serv models but this too doesn't support mobile agent technology.

2.2 AVAILABLE AGENT SOFTWARES

- **Aglets Software Development Toolkit(ASDK)** - Aglets are java objects that can move from one host on the network to another. That is, aglet that executes on one host can suddenly halt execution, dispatch to a remote host, and start executing again. When then aglet improves it takes along its program code as well as the states of all objects it is carrying. A built-in security mechanism makes it safe to host untrusted aglets. Aglets support dynamic and powerful communication that enables agents to communicate with unknown agents as well as well-known agents.
- **Massyve kit version 2.2** - This provides a small, easy, and interactive tool for the fast development of simple applications of multiagent systems. It is essentially oriented towards the education and training activities, although in principle it can also be used to develop non-sophisticated multiagent systems for various application domains.
- **AgentBuilder** - an integrated software toolkit that allows software developers to quickly develop intelligent software agents and agent-based applications. AgentBuilder LITE is ideal for building single-agent, stand-alone applications and small agencies and AgentBuilder Pro has all the features of Lite plus an advanced suite of tools for testing and building multi-agent systems
- **JAFMAS** - Java based Agent Framework for MultiAgent Systems.JAFMAS provides a framework to guide the coherent development of multiagent systems along with a set of classes for agent deployment in Java.

Other Agent Softwares available are: ActorFoundry, ADE, ADK, Agent Factory, JAFMAS, AgentSheets, AgentTool, AGILO, AMETAS, BOND, BotBox Agents, COUGAAR, Gypsy, INGENIAS-IDE , Kaariboga, Madkit, MAGE, RETSINA, SWARM, XRAPTOR, Voyager, Concordia Server, etc.

2.2.1 Detailed Description Of Aglets Software Development Kit 1.0.3

2.2.1.1 Environment For Aglets Server

To run the Aglets Software Development Kit, two environment variables namely AGLET_HOME, and PATH have to be set.

AGLET_HOME variable

The AGLET_HOME variable must be set to the "Aglets1.0.3" directory of the directory in which the Aglets Software Development Kit (ASDK) was installed.

For eg:

```
C:\>set AGLET_HOME=C:\Aglets1.0.3
```

PATH variable

The PATH variable must include the \Aglets1.0.3\bin subdirectory of the ASDK installation (i.e., AGLET_HOME%Aglets1.0.3%\bin) to run.

For eg:

```
C:\>set PATH=&PATH%;%AGLET_HOME %\bin
```

HOME variable

The HOME variable must be specified to an existing directory.
Under the directory, agletsd will make a sub-directory ". aglets"

For eg:

```
C:\>set HOME= c:\
```

2.2.1.2 Environment for developing aglets

To develop an aglet program, further configuration has to be done. To compile any aglet program and run it, three additional environment variables (CLASSPATH, AGLET_PATH, and AGLET_EXPORT_PATH) have to be set.

CLASSPATH variable

The CLASSPATH variable must include the 'lib' directory in which the Aglets library was installed.

For eg:

```
C:\> set CLASSPATH=
C:\Aglets1.0.3\lib\aglets.jar; C:\Aglets1.0.3\public;
```

AGLET_PATH variable

AGLET_PATH is the default look-up path for creating aglets with no codebase. This is used when the null value is passed to createAglet API on the

AgletContext class as the code base argument. For example, if the aglet classes have been placed at C:\ASDK\public, the following is the AGLET_PATH

For eg:

```
C:\> set AGLET_PATH=C:\ASDK\public
```

AGLET_EXPORT_PATH variable

The AGLET_EXPORT_PATH is for files that can be fetched from remote sites. In most cases, it can be the same as AGLET_PATH.

For eg:

```
C:\>set AGLET_EXPORT_PATH=%AGLET_PATH%
```

2.2.1.3 Agent Transfer Protocol(ATP)

ATP is a simple application-level protocol designed to transmit an agent-system-independent manner. Its request consists of a request line, header fields, and content. The request line specifies the method of request, while the header fields contain the parameters of the request. ATP defines the following four standard request methods:

Dispatch:

The dispatch method requests a destination agent system to reconstruct an agent from the content of a request and start executing the agent. If the request is successful, the sender must terminate the agent and release any resources consumed by it.

Retract:

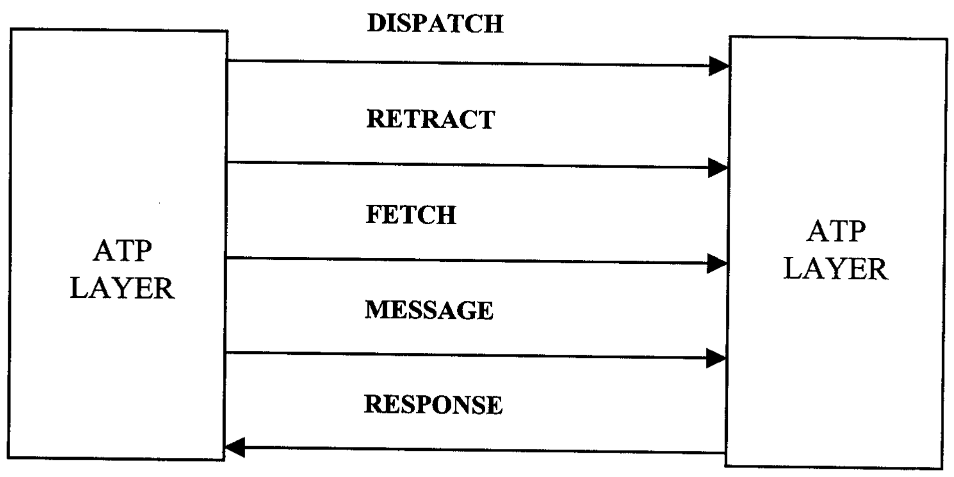
The retract method requests a destination agent system to send the specified agent back to the sender. The receiver is responsible for reconstructing and resuming the agent. If the agent is successfully transferred, the receiver must terminate the agent and release any resources consumed by it.

Fetch:

The fetch method is similar to the GET method in HTTP; it requests a receiver to retrieve and send any identified information. (Normally class files).

Message:

The message method is used to pass a message to agent identified by agent-id and returns a reply value in the response. Although the protocol adopts a request/reply form, it does not lay down any rules for communication scheme between agents.



2.2.1.4 <u>Tahiti Server</u>

Tahiti is an application program that runs as an agent server. Multiple servers (Tahiti) can be run on a single computer by assigning them different port numbers. Tahiti provides a user interface for monitoring, creating, dispatching, and disposing of agents and for setting the agent access privileges for the agent server.

2.2.1.4.1 Running Tahiti

Create a command prompt window and start the daemon by executing a script in the aglets package:

```
agletsd
  or
agletsd –port 9000(on UNIX platform)
```

This daemon will listen for agents on the default port 434-a port reserved for mobile agents or port 9000(on UNIX).It also makes **Tahiti** appear on your screen. **Tahiti** is a desktop interface to manage running agents.

Create another command prompt window and start daemon here:

```
agletsd –port 500
    or
agletsd –port 10000(on UNIX platform)
```

This daemon will listen for agents on another port-in this case port 500.Any port that is currently not occupied by some other application and its operating system can be chosen.

2.2.1.4.2 Option Settings in Tahiti

There are four setting options in **Tahiti**.

1. General preference

Font:

Defines the size of the font in Tahiti. Whether the change of the setting becomes effective immediately or on the next Tahiti invocation depends on the version of Java you are running.

Startup:

It is used to specify an aglet that automatically starts up when Tahiti is started.

Cache Control:

It is used for clearing up the class cache.

User Information:

It contains the registered user information. It enables us to check and modify the name and organization, which are registered in Tahiti and referred to by aglets as the properties of the AgletContext.

2. Network preference

Http Proxy:

In case the system is protected by a firewall, a http proxy server to access the information at the outside of the firewall can be specified here.

3. Security preference

As a security option, the access control details can be specified for each aglet security category (trusted and untrusted are defined as security categories) on the resource categories (File system, Network, Property, and others).

- A trusted aglet is an agent that is created on a server by using the "CreateAglet" panel as a local aglet. A trusted Aglet has the privilege of creating another trusted aglet on the same aglet server.
- An untrusted aglet is one that comes from an external server or that is generated as an instance of an aglet class located at a remote site. In general, the

System Requirements

3. SYSTEM REQUIREMENTS

3.1 HARDWARE REQUIREMENTS

Processor	:	Pentium III
CPU Speed	:	600 MHz
RAM	:	64 MB
Hard Disk	:	4 GB
Keyboard	:	Standard Keyboard

3.2 SOFTWARE REQUIREMENTS

Operating System	:	Windows XP/Windows NT
SDK	:	JDK 1.1.7B
		Aglets 1.0.3 SDK

3.3 OTHER REQUIREMENTS

- Internetworking
- Tahiti – an application program that runs as an agent server

Implementation

4. IMPLEMENTATION

4.1 DEVELOPMENT ENVIRONMENT

Mobile agents can be created using Aglets Software Development Kit. This requires a special environment called the Agent Execution Environment. This environment provides basic facilities such as:

- Mobility

This facility caters to the transportation of the agent between physical nodes.

- Communications

This facility caters to the communication of the agent with the external world.

- Naming and Location

Mobile Agents, like all entities, need to be named. Furthermore, it is important to know at which nodes they are at a given time.

- Security

Mobile Agents need to be protected against hosts and hosts need to be protected against Mobile Agents.

Most Mobile Agent platforms are implemented as Java applications that run on top of Operating systems. Aglets SDK version 1.0.3 has been used for creating and deploying Mobile Agents. Java applications developed with JDK 1.2.x do not support Aglets 1.0.3. They work well only with JDK1.1.7.

Java Development Kit (JDK) is the development environment for building applications, applets and components that can be deployed on the Java platform. The Java 2 SDK software includes tools useful for developing and testing programs written in the Java programming language.

4.2 SYSTEM IMPLEMENTATION

4.2.1 <u>Introduction To Aglet Programming</u>

AGLET:

Aglets are java objects that can move from one host on the Internet to another. An aglet that executes on one host can suddenly halt execution; dispatch itself to a remote host and resume execution there. When the aglet moves, it takes along its program code as well as its data. Aglets are created using Aglet API,which is an agent development kit containing a set of Java classes and interfaces.

AGLET PROXY:

Aglet Proxy interface is a placeholder for aglets. The purpose of this interface is to provide a mechanism to control and limit direct access to aglets.

AGLET CONTEXT:

The AgletContext interface is used by an aglet to get information about its environment and to send messages to the environment and other aglets currently active in that environment. It provides means for maintaining and managing running aglets in an environment where the host system is secured against malicious aglets.

4.2.1.1 <u>Life Cycle Of An Aglet</u>

The main events in the life cycle of an aglet are categorized as:

•	Creation	-	Create or Clone
•	Disposal	-	Dispose
•	Mobility	-	Dispatch and Retract
•	Persistence	-	Deactivate and Activate

4.2.1.2 <u>Creation Of An Aglet</u>

Aglets can be created using the createAglet() function of the AgletContext class. It returns the proxy of the created aglet.

Syntax:

```
Public abstract AgletProxy createAglet(URL codebase,
String code,Object init) throws IOException,
AgletException, ClassNotFoundException,
InstantiationException
```

Creates an instance of the specified aglet class.The aglet's class code file can be located on the local file system as well as on a remote server. If the codebase is null, the context will search for the code in the local system's aglet search path (AGLET_PATH). When an aglet is created its onCreation() function followed by run() will be invoked.

4.2.1.3 <u>Cloning An Aglet</u>

An aglet can be cloned using the clone() method of the Aglet class.

Syntax:

```
public final Object clone() throws
CloneNotSupportedException
```

Clones the aglet and the proxy that holds the aglet. Notice that it is the cloned aglet proxy, which is returned by this method.

4.2.1.4 Dispatching An Aglet

The dispatch()function of the Aglet class is used to dispatch an aglet to any specified destination.

Syntax:

```
public final void dispatch(URL destination)throws
IOException,   RequestRefusedException
```

Dispatches the aglet to the location(host) specified by the argument.This method invokes the onDispatching() followed by onArrival() and run()functions of this aglet.

4.2.1.5 Disposing An Aglet

The function dispose() of the Aglet class is used to dispose an aglet.

Syntax:

```
public final void dispose()
```

Destroys and removes the aglet from its current aglet context. A successful invocation of this method will kill all threads created by the given aglet. This function will invoke the onDispatching() method of the Aglet.

4.2.1.6 Retracting An Aglet

The function retractAglet() is used to call an Aglet back from a remote server.

Syntax:

```
public abstract AgletProxy retractAglet(URL url) throws
IOException, AgletException
```

Or

```
public abstract AgletProxy retractAglet(URL url,AgletID
aid) throws IOException,AgletException
```

4.2.1.7 Deactivation Of An Aglet

An aglet can be deactivated using deactivate() function of the Aglet class.

Syntax:

```
public final void deactivate(long duration)throws
IOException
```

Deactivates the aglet. The aglet will temporarily be stopped and removed from its current context. It will return to the context and resume execution after the specified period has elapsed.

4.3 PROGRAMMING

4.3.1 Initializations required

For Java,

- set path=%path%;C:\jdk1.1.7B\bin
- set CLASSPATH=C:\jdk1.1.7B\lib\classes.zip; C:\Aglets1.0.3\lib\aglets.jar; C:\Aglets1.0.3\public;

For Aglets,

- set AGLET_HOME=C:\Aglets1.0.3
- set JDK_HOME=C:\jdk1.1.7B
- set HOME=C:\

4.3.2 Pseudocode

```
BEGIN
        CREATION OF MOBILE AGENT
        GET THE GENERAL NEED, DESTINATION1 CAPACITYAND& PRIORITY, DESTINATION2
                CAPACITY AND PRIORITY, AND SOURCE CAPACITY.
        CHECK THE NEED WITH THE DESTINATION1 , DESTINATION2 & SOURCE'S CAPACITIES
        IF THE NEED IS HIGHER THEN
                SAY "TRANSMISION IS NOT POSSIBLE"
        ELSE
                CHECK FOR THE PRIORITIES OF DESTINATION1 & DESTINATION2
        END IF
        IF DESTINATION1 PRIORITY IS HIGHER THEN
                CHECK FOR ITS CAPACITY
                IF NEED IS LESS THAN OR EQUAL TO DESTINATION1 AND SOURCE CAPACITY
                        THEN
                        ALLOCATE THE NEEDED CAPACITY OF DESTINATION1
                        INTIMATE the CAPACITY ACQUISITION
                        RELEASE THE CAPACITY OF DESTINATION1 IF ANY
                ELSE ALSO USE THE CAPACITY OF DESTINATION2
```

```
                    INTIMATE THE CAPACITY ACQUISITION
                    RELEASE THE CAPACITY OF DESTINATION2 WHICH IS FREE
            END IF
        ELSE USE THE CAPACITY OF DESTINATION2 AND SOURCE
            ALLOCATE THE NEEDED CAPACITY OF DESTINATION2
            INTIMATE the CAPACITY ACQUISITION
            RELEASE THE CAPACITY OF DESTINATION2 IF ANY
            IF DESTINATION2 AND SOURCE CAPACITY NOT SUFFICIENT THEN
                    USE THE CAPACITY OF DESTINATION1
                    INTIMATE THE CAPACITY ACQUISITION
                    RELEASE THE CAPACITY OF DESTINATION1 WHICH IS FREE
            END IF
        END IF
END
```

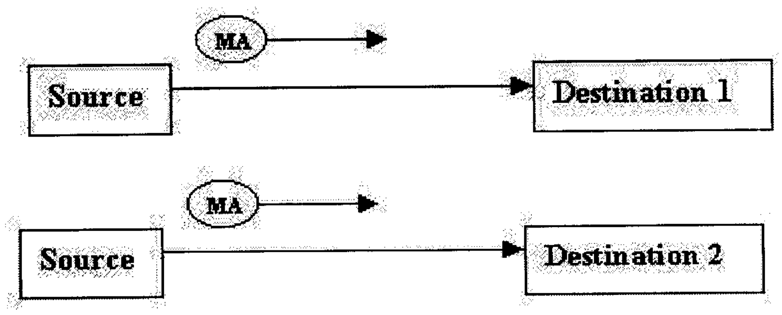
4.3.3 <u>Description of Modules</u>

The steps involved are:

- Mobile Agent Creation in the source
- Acquiring details from other systems using Mobile Agents
- Calculation the link capacity and notification.

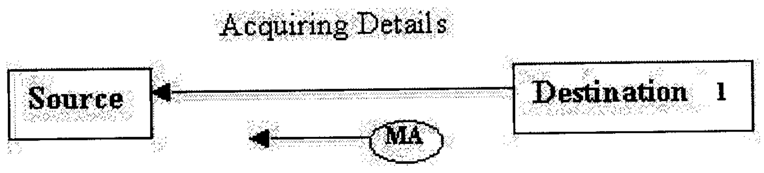
- **Mobile Agent Creation in the source**

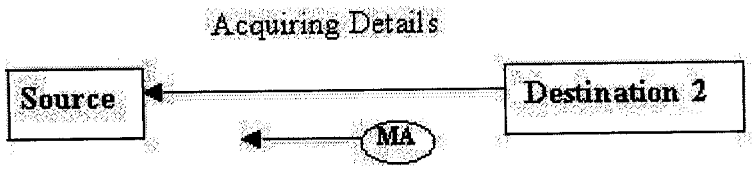
Here, a mobile agent is created. The address of the destination is obtained using a dialog box.



- **Acquiring details from other systems using Mobile Agents**

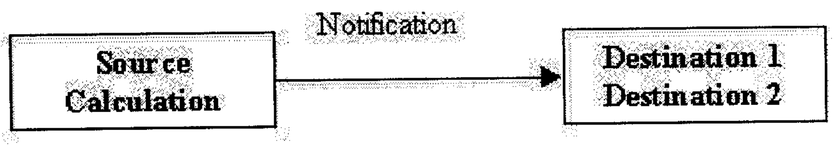
The mobile agent is sent to the destination system. There, it invokes a dialog box to get details like priority and capacity that can be supported in a virtual link.





- **Calculation of the link capacity and notification**

The mobile agent is returns back to the source after getting priority and capacity details from the destinations. The source also has its own capacity. The capacity of the source and the need for the virtual link is obtained from the server. Based on these details, the sufficiency of capacity to the need is calculated, and the usage of resources is notified to the appropriate systems.



Testing

5. TESTING

Testing is a mandatory and continuous process for the successful completion of any project. Testing is done at various levels. A final testing is done to access the correctness of the whole software as such. Here several test cases are taken and real life situations are created while testing.

Initial settings

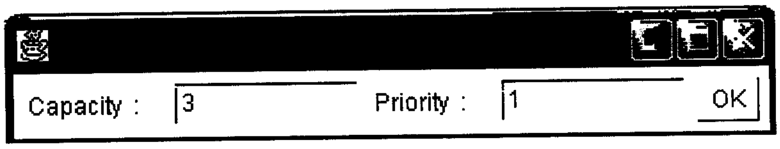
Source: atp://ccs87:10000

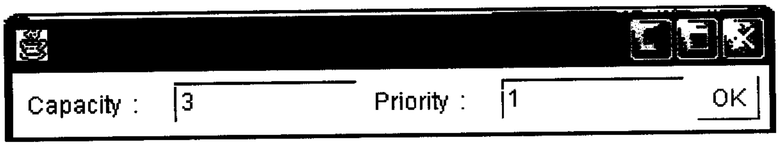
Destination 1 : atp://ccs86:10001

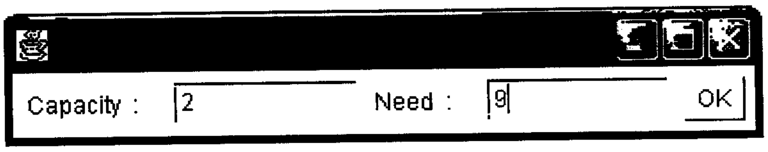
Destination 2 : atp://ccs86:10002

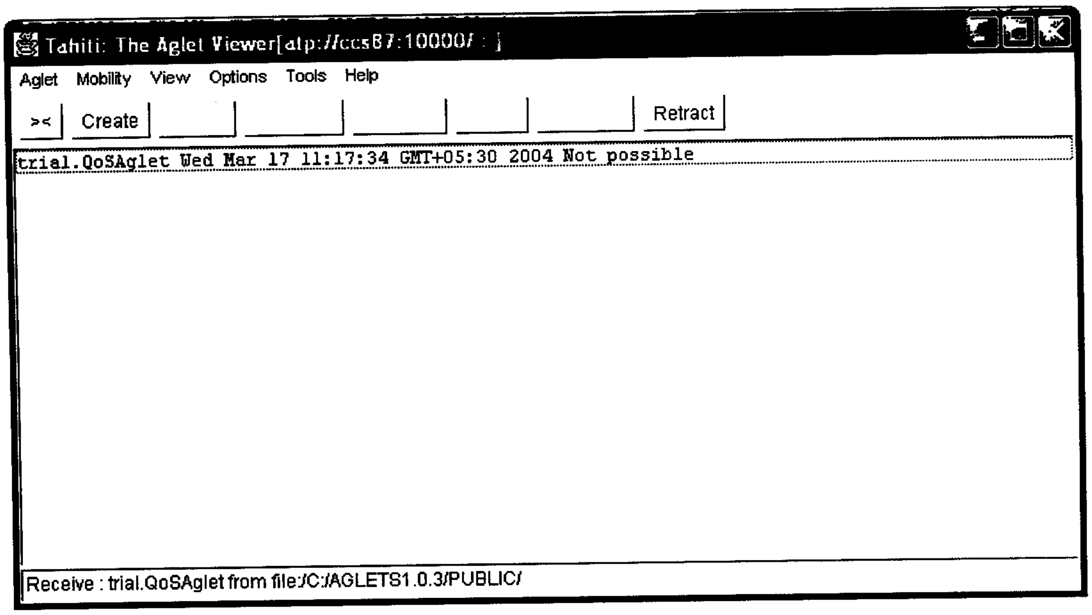
Test case 1:

Need	:	9
Source capacity	:	2
Destination 1 Capacity	:	3
Destination 1 Priority	:	1
Destination 2 Capacity	:	3
Destination 2 Priority	:	1
Result	**:**	**Not Possible**


Capacity : 3 Priority : 1


Capacity : 3 Priority : 1


Capacity : 2 Need : 9

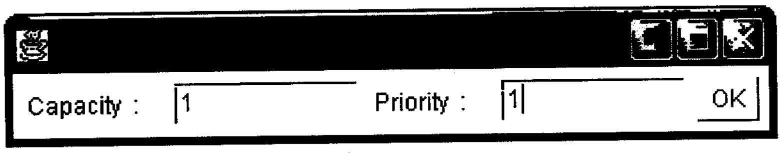

Tahiti: The Aglet Viewer[atp://ccs87:10000/ :]
Aglet Mobility View Options Tools Help
>< Create Retract

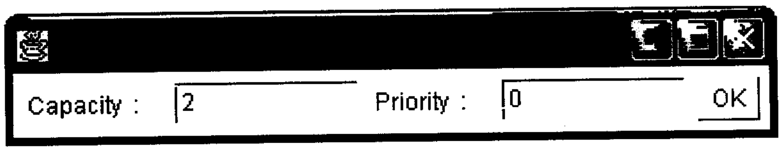
```
trial.QoSAglet Wed Mar 17 11:17:34 GMT+05:30 2004 Not possible
```

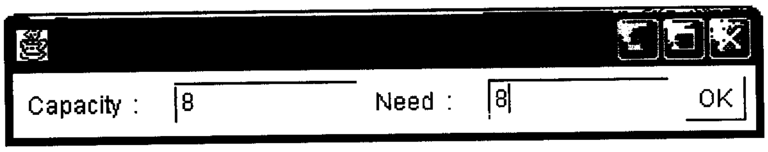
Receive : trial.QoSAglet from file:/C:/AGLETS1.0.3/PUBLIC/

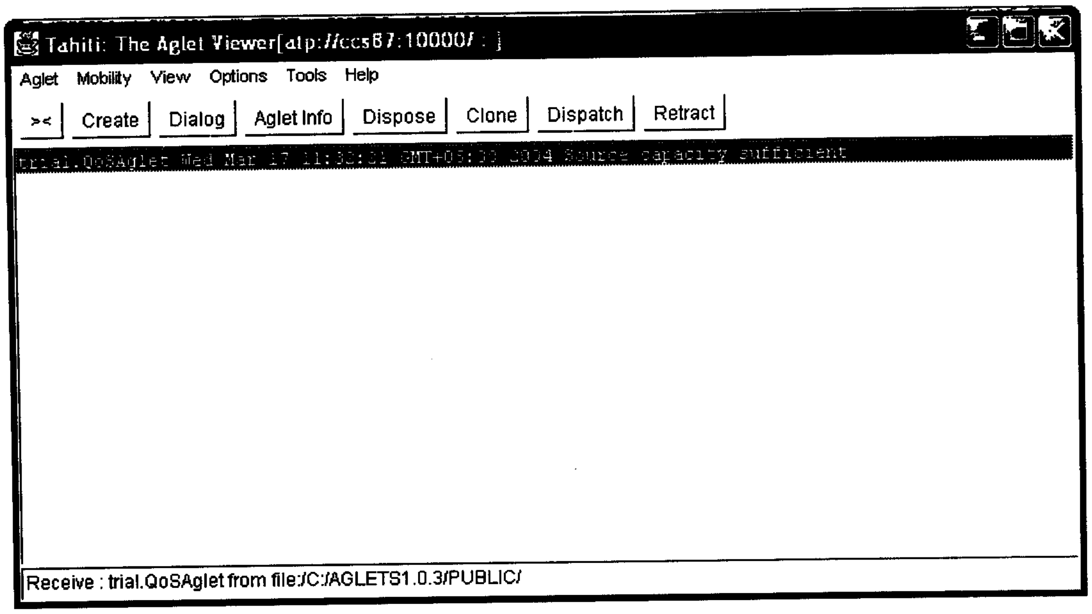
Test case 2:

Need	:	8
Source capacity	:	8
Destination 1 Capacity	:	1
Destination 1 Priority	:	1
Destination 2 Capacity	:	2
Destination 2 Priority	:	0
Result	:	**Source Capacity Sufficient**







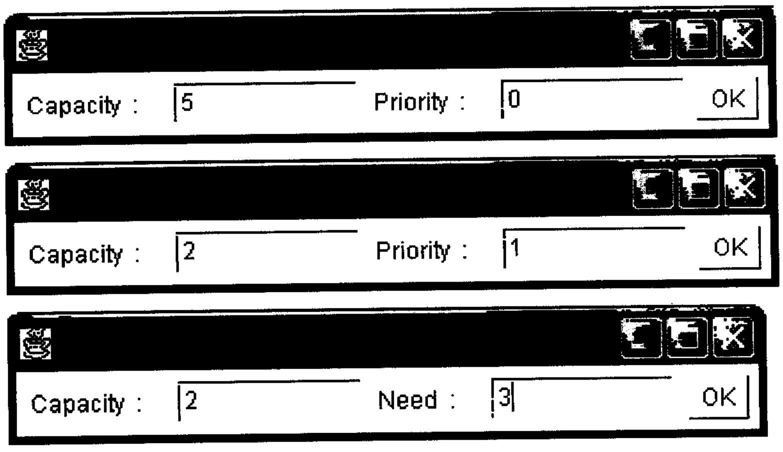


```
trial.QoSAglet Wed Mar 17 11:33:31 GMT+05:30 2004 Source capacity sufficient
```

```
Receive : trial.QoSAglet from file:/C:/AGLETS1.0.3/PUBLIC/
```

Test case 3:

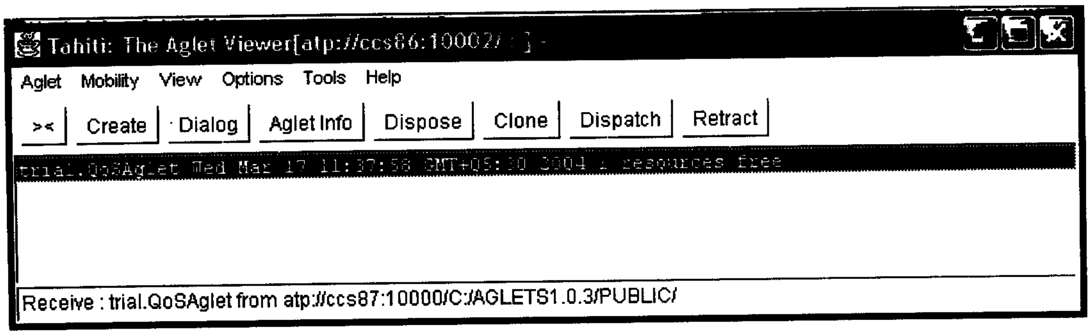
Need	:	3
Source capacity	:	2
Destination 1 Capacity	:	5
Destination 1 Priority	:	0
Destination 2 Capacity	:	2
Destination 2 Priority	:	1
Result	**:**	**1 resource free in Destination 2 (Destination 1 capacity not used)**



Capacity : 5 Priority : 0

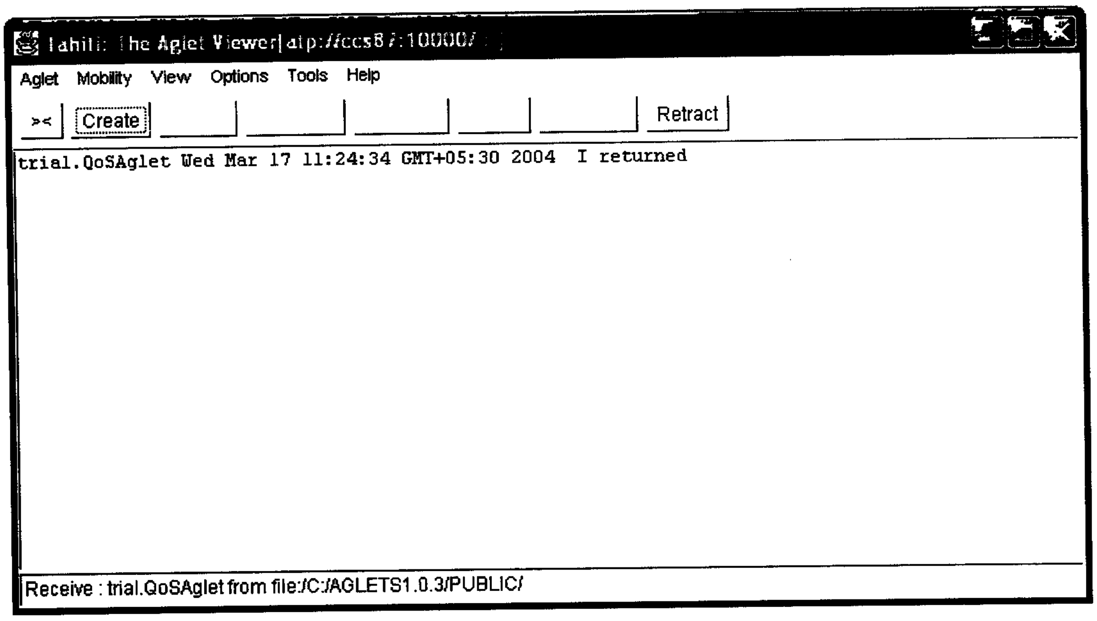
Capacity : 2 Priority : 1

Capacity : 2 Need : 3


Tahiti: The Aglet Viewer[atp://ccs86:10002/ :] -
Aglet Mobility View Options Tools Help
>< Create Dialog Aglet Info Dispose Clone Dispatch Retract

```
trial.QoSAglet Wed Mar 17 11:37:58 GMT+05:30 2004 1 resources free
```

```
Receive : trial.QoSAglet from atp://ccs87:10000/C:/AGLETS1.0.3/PUBLIC/
```


Tahiti: The Aglet Viewer[atp://ccs87:10000/ :]
Aglet Mobility View Options Tools Help
>< Create Retract

```
trial.QoSAglet Wed Mar 17 11:24:34 GMT+05:30 2004  I returned
```

```
Receive : trial.QoSAglet from file:/C:/AGLETS1.0.3/PUBLIC/
```

Future Enhancements

6. FUTURE ENHANCEMENTS

Our project has been designed to implement traffic flow through virtual links in Diff-serv model in a distributed environment using Mobile Agent Technology. This can be further enhanced to

- implement other functionalities of Diff-serv model
- implement Int-serv model

  in a distributed computing environment

Conclusion

7. CONCLUSION

Our project provides an overview of Int-serv and Diff-serv models, Mobile Agents, different Agent softwares available,etc. This also shows the use of Mobile Agents for information retrieval by applying it in the implementation of Diff-serv environment.

We conclude that Mobile Agents are suitable for performance critical applications, where the functionalities offered by Client/Server technology do not suffice. The flexibility of the approach will push network equipment developers, network and service providers to open their systems to allow for the diffusion of more advanced services that can be deployed from the adoption of this new programming paradigm.

This project has also enabled us to gain in depth knowledge of Java and Aglets programming.

Reference

8. REFERENCE

Mobile Agent and QoS management related information

- Aurelio La Corte, Antonio Puliafito and Orazio Tomarchio, "QoS management in programmable networks through mobile agents"
- Hermann De Meer, Aurelio La Corte, Antonio Puliafito, and Orazio Tomarchio, "Programmable Agents for Flexible QoS Management in IP Networks"
- Paolo Bellavista and Antonio Corradi, "An Open Secure Mobile Agent Framework for Systems Management"
- Ichiro Satoh, "A Mobile Agent-Based Framework for Active Networks"
- www.cisco.com
- www.isi.edu

Aglets related information

- www.trl.ibm.com
- Christian Weigel, "IBM Agents – Mobile Java Agents with Aglets"
- Hamsapriya, Jaishree,Swapnapriya, "Multiparty Event Scheduling"

JAVA related information

- Herbert Schildt, Tata McGraw Hill Edition, 2002, "Java 2 : The Complete Reference, Fourth Edition"
- www.java.sun.com

Other related information

- www.agentlink.org
- www.j-sim.org
- www.sun195.iit.unict.it/MAP
- www.borland.com

9. APPENDIX

9.1 IMPORTANT TERMS

- **Quality of Service (QoS) :** QoS refers to the capability of a network to provide better service to selected network traffic over various technologies. The primary goal of QoS is to provide priority including dedicated bandwidth, controlled jitter and latency (required by some real-time and interactive traffic), and improved loss characteristics.
- **Int-Serv model :** Integrated Services model (int serv) is a traffic prioritization paradigm that enables deterministic traffic qualities for select traffic within the highly undeterministic world of IP communications.
- **Diff-Serv model :** Differentiated Services model (diff serv) defines a framework on which differentiated service level offerings can be provided for different traffic classes. This standard defines per-hop-behaviors (PHB) that can be implemented on a node-to-node basis, rather than end-to-end behaviors such as that defined by IntServ and RSVP(Resource Reservation Protocol).
- **Mobile Agent :** A Mobile Agent(MA) is a program that can migrate from system to system within a network environment. It performs some processing at each host.
- **Aglets :** Aglets are a special type of Agents developed by IBM Research. Aglets are hosted by an Aglet Server that provides that the environment for them to run in. The Java Virtual Machine and a special Aglet security manager ensure safety while receiving and hosting Aglets. The word *Aglet* means "lightweight agent", it is composed of the word *agent* and the word *applet* (lightweight application).

9.2 SAMPLE CODING

```
package trial;

import com.ibm.aglet.*;
import com.ibm.aglet.event.*;
import com.ibm.aglet.util.*;
import com.ibm.agletx.util.SimpleItinerary;
import java.lang.InterruptedException;
import java.io.Externalizable;
import java.io.ObjectInput;
import java.io.ObjectOutput;
import java.io.IOException;
import java.net.*;
import java.awt.*;
import java.util.Enumeration;
import java.lang.*;

public class QoSAglet extends Aglet
{
        transient Frame my_dialog = new MyDialog(this);
        ClientDlg c_dialog = new ClientDlg(this);
        ServerDlg s_dialog = new ServerDlg(this);
        String message = "Hello World!";
        String Vlink1 ="";
        String Vlink2 ="";
        public String ms1=new String();
        String home = null;
        String destination="";
        String info="";
        int c1=0,c2=0;
        int c1_capacity,c2_capacity;
        int c1_priority,c2_priority;
        int s_need=0,s_capacity=0;
        boolean link_flag=false;
    SimpleItinerary itinerary = null;
                //Initializes the aglet
        public void onCreation(Object init)
        {
        itinerary = new SimpleItinerary(this);
                my_dialog.pack();
        my_dialog.resize(my_dialog.preferredSize());
                my_dialog.show();
        home = getAgletContext().getHostingURL().toString();
        }
                // handles message
```

```
public boolean handleMessage(Message msg)
     {
  if (msg.sameKind("atHome")) {
    atHome(msg);
  } else if (msg.sameKind("startTrip1")) {
    startTrip(msg);
      } else if (msg.sameKind("iReturned")) {
    iReturned(msg);
  } else if (msg.sameKind("sayHello")) {
    sayHello(msg);
  }  else if (msg.sameKind("AfterCal")) {
    AfterCall(msg);
  } else if (msg.sameKind("dialog")) {
    dialog(msg);
  } else {
    return false;
  }
  return true;
}

public void atHome(Message msg)
      {
          my_dialog = new MyDialog(this);
              my_dialog.pack();
              my_dialog.resize(my_dialog.preferredSize());
     my_dialog.show();

              if(c1==1)
                      setText("client1 "+c1_capacity+"   : "+c1_priority);
      else if(c2==1)
              setText("client2 "+c2_capacity+"   : "+c2_priority);
          }

      public void AfterCall(Message msg)
      {
      setText(info);
      waitMessage(5 * 1000);
              try
              {
                      itinerary.go(home,"iReturned");
              }
              catch (Exception ex) {
      ex.printStackTrace();
    }
 }

      public void iReturned(Message msg)
      {
    setText(" I returned");
```

```
        }

                // Starts the trip of this aglet to the destination.

        public synchronized void startTrip(Message msg)
        {
      destination = (String)msg.getArg();
            try
                {
                itinerary.go(destination, "sayHello");
        }
        catch (Exception ex) {
        ex.printStackTrace();
      }
   }


   public void sayHello(Message msg)
        {
     setText(" gets data");
        waitMessage(5 * 1000);
     try
        {
        c_dialog.pack();
                c_dialog.resize(c_dialog.preferredSize());
                c_dialog.show();
                c_dialog.msg1.setText("");
                c_dialog.msg2.setText("");

        }
        catch (Exception ex)
        {
        ex.printStackTrace();
      }
   }

   // Creates and shows the dialog window.

   public void dialog(Message msg)
        {
        if (my_dialog == null)
                {
                        my_dialog = new MyDialog(this);
                        my_dialog.pack();
                        my_dialog.resize(my_dialog.preferredSize());
                }
                my_dialog.show();
        }
}
```

```
class ServerDlg extends Frame
{
        public int sc,need;
        public boolean ok=false;
        private QoSAglet aglet  = null;
        Label l1=new Label("Capacity  :");
        TextField ServerCapacity      = new TextField(10);
        Label l2=new Label("Need  :");
        TextField ServerNeed  =new TextField(10);
        private Button done         = new Button("OK");
        ServerDlg(QoSAglet aglet)
        {
        this.aglet = aglet;
                GridLayout grid = new GridLayout();
        setLayout(grid);
                add(ServerCapacity);
                add(ServerNeed);
                Panel p = new Panel();
              add(p);
                p.setLayout(new FlowLayout());
                p.add(l1);
                p.add(ServerCapacity);
                p.add(l2);
                p.add(ServerNeed);
                p.add(done);
    }

  //handles events
    public boolean handleEvent(Event ev)
         {
      if (ev.id == Event.WINDOW_DESTROY)
                 {
        hide();
        return true;
      }
      return super.handleEvent(ev);
    }

    // Handles the action

    public boolean action(Event ev, Object obj)
          {
          String m="";
                  int n;
                  if (ev.target == done)
                  {
                  m=ServerNeed.getText();
                  aglet.s_need=Integer.parseInt(m);
```

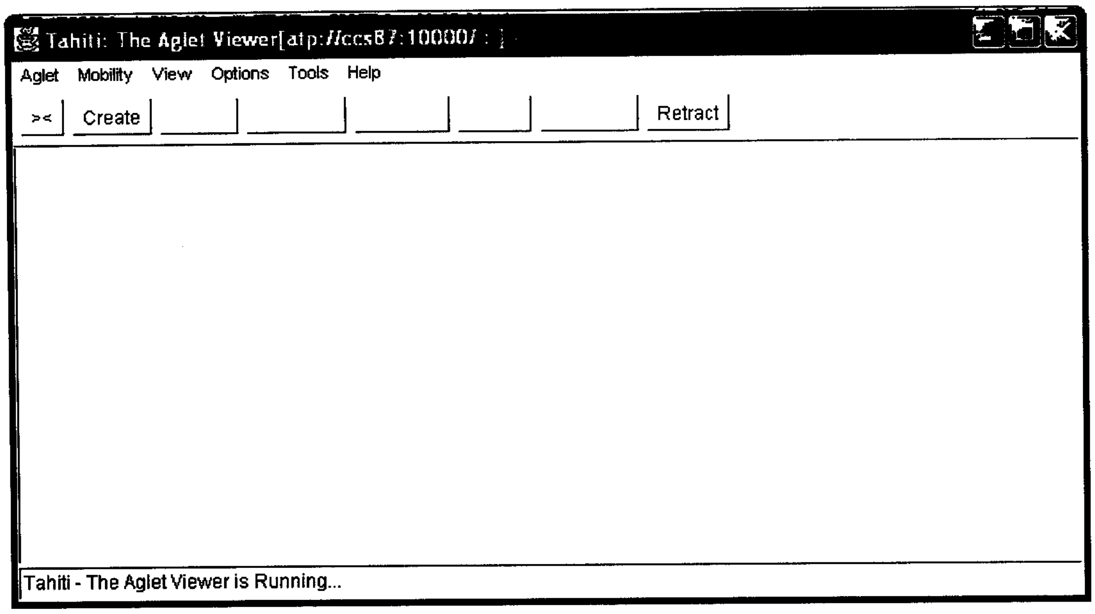
```
                        m=ServerCapacity.getText();
                        aglet.s_capacity=Integer.parseInt(m);
                        hide();


        if(aglet.s_need>(aglet.c1_capacity+aglet.c2_capacity+aglet.s_capacity))
                        {
                                aglet.setText("Not possible");
                                aglet.waitMessage(2000);
                                aglet.dispose();
                        }
                        else
                        {
                        if(aglet.s_capacity>=aglet.s_need)
                                {
                                        //sufficient
                                        aglet.setText("Source capacity sufficient");
                                }
                                else
                                {
                                        //should get resources
                                        if (aglet.c1_priority>=aglet.c2_priority)
                                        {
                                                //c1 higher
                                                aglet.setText("Destination 1 priority
higher");

        if((aglet.s_capacity+aglet.c1_capacity)>=aglet.s_need)
                                        {
                                                        //sufficient --> c1 + s

        n=aglet.s_capacity+aglet.c1_capacity-aglet.s_need;
                                                        aglet.info=n+" resources free";
                                                        try {
                                                        aglet.itinerary.go(aglet.Vlink1,
"AfterCal");
                                                } catch (Exception ex) {
                                                 ex.printStackTrace();
                                                        }
                                                }
                                                else
                                                {
                                                        //use c1 + c2 + s

        n=aglet.s_capacity+aglet.c1_capacity+aglet.c2_capacity-aglet.s_need;
                                                        aglet.info=n+" resources free";
                                                        try {
                                                        aglet.itinerary.go(aglet.Vlink2,
"AfterCal");
```

```
                                                } catch (Exception ex) {
                                                 ex.printStackTrace();
                                                        }
                                                }
                                        }
                                        else
                                        {
                                                //c2 higher
                                                aglet.setText("Destination 2 priority
higher");

        if((aglet.s_capacity+aglet.c2_capacity)>=aglet.s_need)
                                        {
                                                        //sufficient - c2 + s

        n=aglet.s_capacity+aglet.c2_capacity-aglet.s_need;
                                                        aglet.info=n+" resources free";
                                                        try {
                                                aglet.itinerary.go(aglet.Vlink2,
"AfterCal");
                                                        } catch (Exception ex) {
                                                         ex.printStackTrace();
                                                         }
                                                }
                                                else
                                                {
                                                        //use c1 + c2 + s

        n=aglet.s_capacity+aglet.c1_capacity+aglet.c2_capacity-aglet.s_need;
                                                        aglet.info=n+" resources free";
                                                        try {
                                                        aglet.itinerary.go(aglet.Vlink1,
"AfterCal");
                                                } catch (Exception ex) {
                                                 ex.printStackTrace();
                                                        }
                                                }
                                        }

                                }
                        }
                }
              return true;
           }
        }

        class ClientDlg extends Frame
        {
                public int c0,p0;
        String m="";
        private QoSAglet aglet  = null;
        Label l1=new Label("Capacity  :");
        TextField msg1      = new TextField(10);
        Label l2=new Label("Priority  :");
        TextField msg2  =new TextField(10);
        private Button done         = new Button("OK");
        ClientDlg(QoSAglet aglet)
        {
        this.aglet = aglet;
                GridLayout grid = new GridLayout();
                setLayout(grid);
                add(msg1);
                add(msg2);
        Panel p = new Panel();
                add(p);
                p.setLayout(new FlowLayout());
                p.add(l1);
                p.add(msg1);
                p.add(l2);
                p.add(msg2);
                p.add(done);
    }

public boolean handleEvent(Event ev)
        {
         String m="";
      if (ev.id == Event.WINDOW_DESTROY)
                {hide();
              }
      return super.handleEvent(ev);
    }

public boolean action(Event ev, Object obj)
        {

                if (ev.target == done)
                {
                        m=msg1.getText();
                        if(aglet.c1==0)
                        {
                                        //aglet.Vlink1=aglet.destination;
                                aglet.c1_capacity=Integer.parseInt(m);
                                m=msg2.getText();
                                aglet.c1_priority=Integer.parseInt(m);
                                aglet.c1=1;
                        aglet.setText(aglet.Vlink1+"Capacity1
:"+aglet.c1_capacity+"Priority :"+aglet.c1_priority);
                        }
```

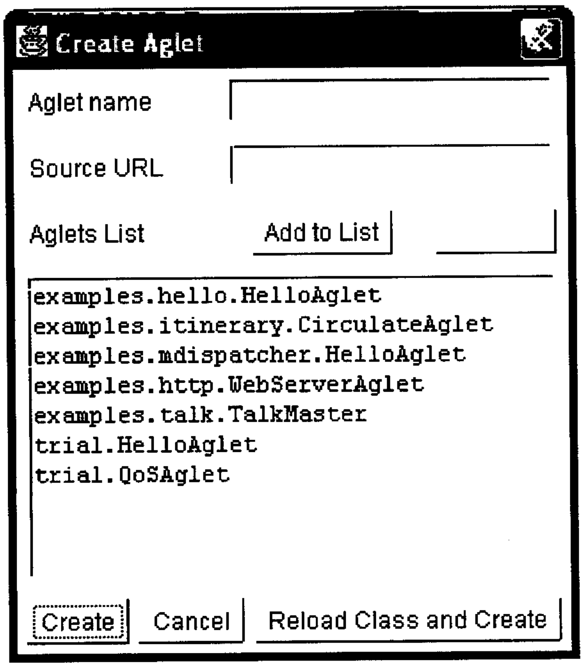
```
                    // set  Capacity & Priority for VLink 1
                    else if(aglet.c2==0)
                    {
                            //aglet.Vlink2=aglet.destination;
                            aglet.c2_capacity=Integer.parseInt(m);
                            m=msg2.getText();
                            aglet.c2_priority=Integer.parseInt(m);
                    aglet.c2=1;
                    aglet.setText("Capacity2 :"+aglet.c2_capacity+"Priority
:"+aglet.c2_priority);
                    }
                hide();
                    try {
                    aglet.itinerary.go(aglet.home, "atHome");
                    }
                    catch (Exception ex) {
                    ex.printStackTrace();
        }

        return true;
      }
      return true;
   }
}

class MyDialog extends Frame
{
   String str=" ";
   private QoSAglet aglet  = null;
   private AddressChooser dest = new AddressChooser();
   private Button go         = new Button("GO!");
   private Button calculate        = new Button("Calculate");
   private Button close       = new Button("CLOSE");

   MyDialog(QoSAglet aglet)
        {
     this.aglet = aglet;
     layoutComponents();
        }

    private void layoutComponents()
        {
        GridBagLayout grid = new GridBagLayout();
      GridBagConstraints cns = new GridBagConstraints();
      setLayout(grid);

      cns.weightx = 0.5;
      cns.ipadx = cns.ipady = 5;
      cns.fill = GridBagConstraints.HORIZONTAL;
    cns.insets = new Insets(5,5,5,5);

    cns.weightx = 1.0;
    cns.gridwidth = GridBagConstraints.REMAINDER;
    grid.setConstraints(dest, cns);
    add(dest);


    cns.weighty = 0.0;
    cns.fill = GridBagConstraints.NONE;
    cns.gridheight = 1;

    Panel p = new Panel();

    grid.setConstraints(p, cns);
    add(p);
    p.setLayout(new FlowLayout());
    p.add(go);
    p.add(calculate);
    p.add(close);
  }

 //handles events
  public boolean handleEvent(Event ev)
       {
    if (ev.id == Event.WINDOW_DESTROY)
              {
          //hide();
               return true;
    }
    return super.handleEvent(ev);
  }

  // Handles the action

  public boolean action(Event ev, Object obj)
       {
              if (ev.target == go)
              {
                     if(aglet.link_flag==false)
                     {
                            aglet.Vlink1=dest.getAddress();
                     }
                     else
                     {
                            aglet.Vlink2=dest.getAddress();
                     }
                     if(aglet.link_flag==false)
                            aglet.link_flag=true;
```

9.3 SAMPLE SCREENS

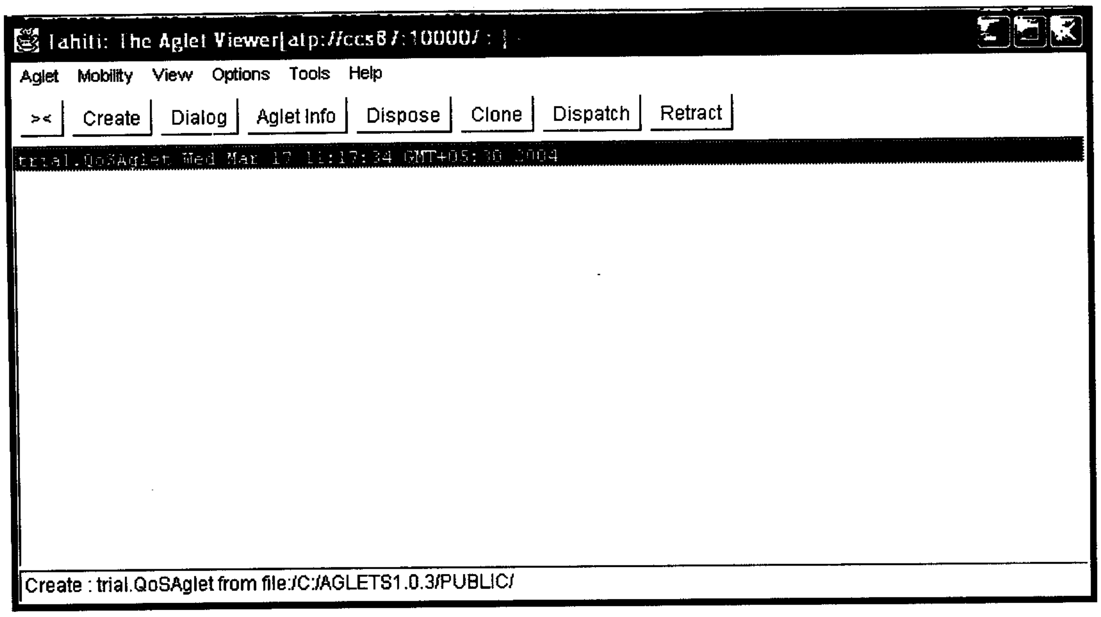
9.3.1 Source


Tahiti: The Aglet Viewer[atp://ccs87:10000/ :]
Aglet Mobility View Options Tools Help
>< Create Retract

Tahiti - The Aglet Viewer is Running...



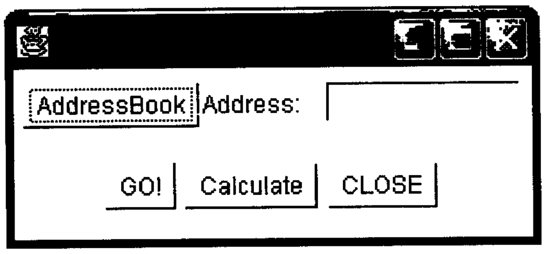
```
examples.hello.HelloAglet
examples.itinerary.CirculateAglet
examples.mdispatcher.HelloAglet
examples.http.WebServerAglet
examples.talk.TalkMaster
trial.HelloAglet
trial.QoSAglet
```

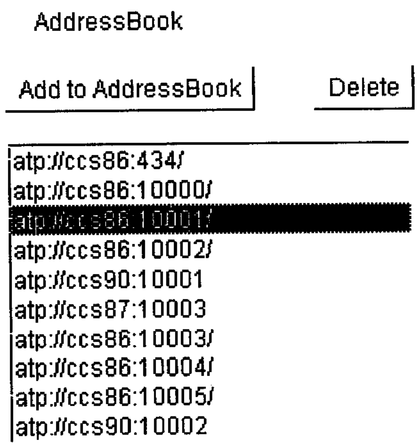


Tahiti: The Aglet Viewer[atp://ccs87:10000/ :]

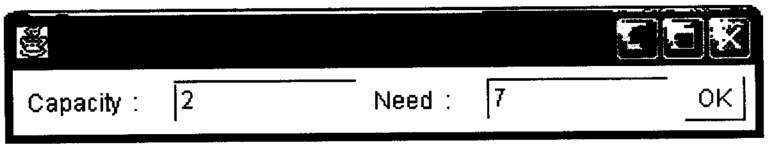
trial.QoSAglet Wed Mar 17 11:17:34 GMT+05:30 2004

Create : trial.QoSAglet from file:/C:/AGLETS1.0.3/PUBLIC/

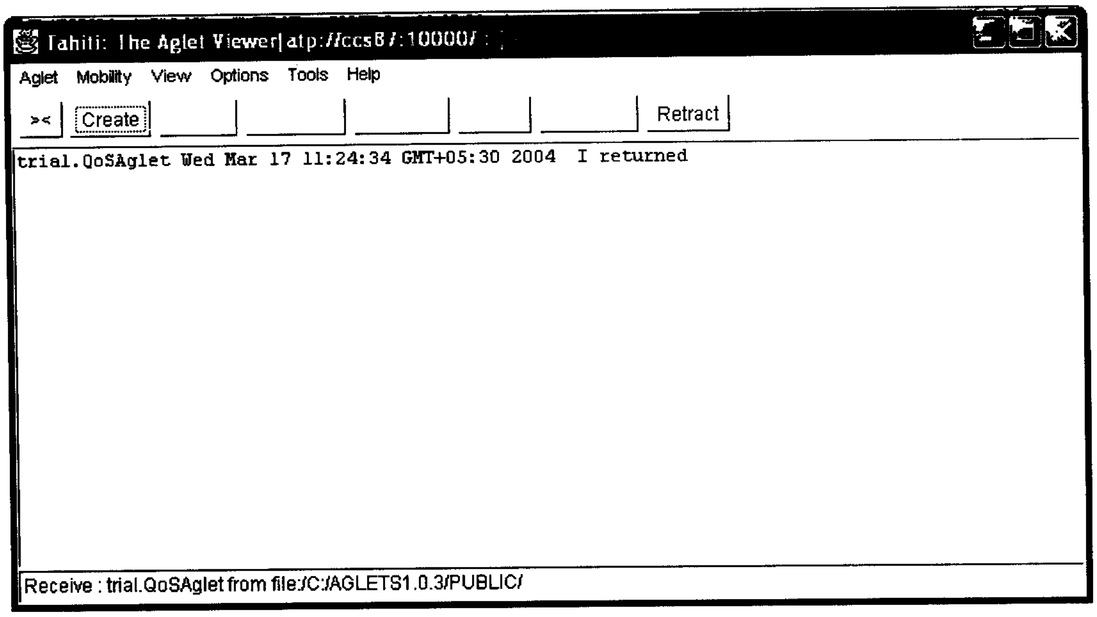




atp://ccs86:434/
atp://ccs86:10000/
atp://ccs86:10001/
atp://ccs86:10002/
atp://ccs90:10001
atp://ccs87:10003
atp://ccs86:10003/
atp://ccs86:10004/
atp://ccs86:10005/
atp://ccs90:10002



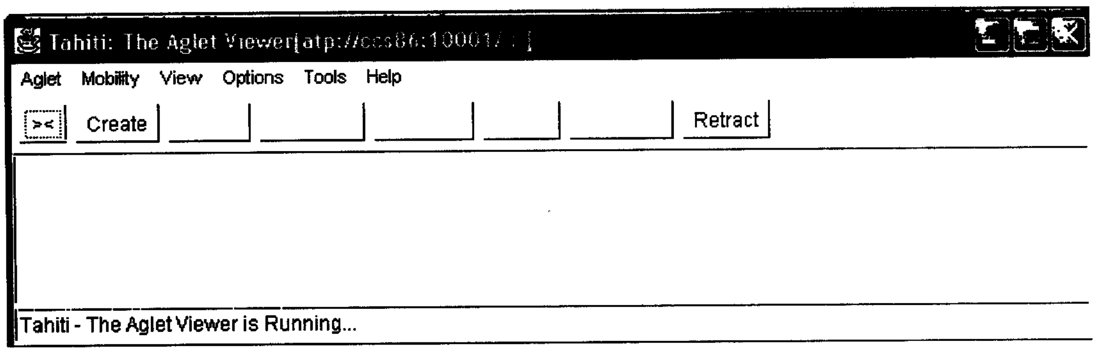
Capacity : 2 Need : 7


Tahiti: The Aglet Viewer|atp://ccs87:10000/ : |
Aglet Mobility View Options Tools Help
>< Create Retract

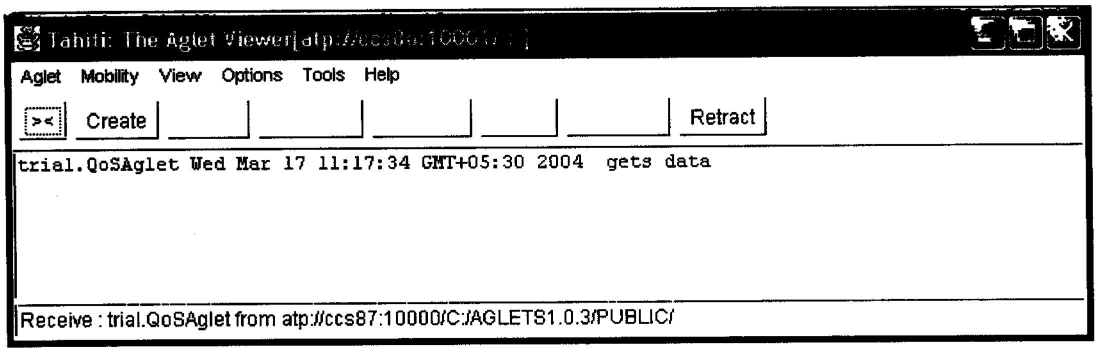
```
trial.QoSAglet Wed Mar 17 11:24:34 GMT+05:30 2004  I returned
```

Receive : trial.QoSAglet from file:/C:/AGLETS1.0.3/PUBLIC/

9.3.2 Destination 1

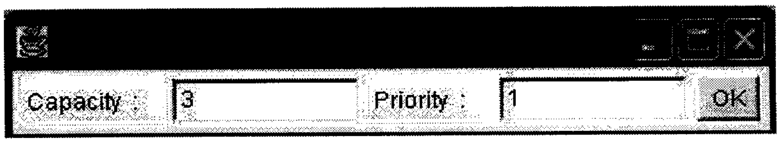

Tahiti: The Aglet Viewer|atp://ccs86:10001/ : |
Aglet Mobility View Options Tools Help
>< Create Retract

Tahiti - The Aglet Viewer is Running...


Tahiti: The Aglet Viewer|atp://ccs86:10001/ : |
Aglet Mobility View Options Tools Help
>< Create Retract

```
trial.QoSAglet Wed Mar 17 11:17:34 GMT+05:30 2004  gets data
```

Receive : trial.QoSAglet from atp://ccs87:10000/C:/AGLETS1.0.3/PUBLIC/



Capacity : 3 Priority : 1