University College London
Department of Computer Science
MSc Data Communications, Networks and Distributed Systems

User Manual

A Spatio-Temporal Middleware for Delay Tolerant Systems

MiDAS:

Middleware Implementation for DTN's with Autonomic Selection

Group Members:

Jessica Allen
Jiaxi Fu
Chukwuka Oyem
Dmytryo Stasyuk
Xueyu Jin

Supervisor:

Dr. Cecilia Mascolo

September 2, 2005

TABLE OF CONTENTS

1.1. Introduction

Welcome to the technical user manual of MiDAS. This technical manual covers the recommended system requirements needed to run MiDAS successfully. It also presents the MiDAS APIs provided and describes how they can be used to both implement applications to use the interfaces and to plug in new dissemination strategies to the middleware framework. Simple send and receive applications have also been provided with MiDAS as examples of how the applications can use the middleware and this technical manual will describe how these can be used.

1.2. System Requirements

MiDAS was implemented and tested on Java 2 standard development kit (SDK) 1.4.2 on Windows XP, Mac OSX and Unix environments and so is capable on running on all these platforms.

A minimum storage space of 317Mb is needed for the installation of Java SDK 1.4.2 and the installation instructions can be found at http://java.sun.com/j2se/1.4.2/install-windows.html. In addition to this, a minimum storage space of 1.2 MB is needed for the installation of MiDAS. This storage space is sufficient for mobile devices as MiDAS was successfully stored on a Compaq iPaq PDA although it was not tested on it. That being said, MiDAS uses only import packages that exist in the J2ME Personal Profiles (PP) libraries and so we believe that it is deployable on a real java enabled mobile device.

1.3. MiDAS API

MiDAS provides a set of APIs which application developers may use to interact with the middleware. The two main interface packages provided are the **ApplicationInterface** and the **DisseminationInterface** packages. Further to this an Autonomic Selection interface is provided in the event that developers wish to provide their own implementation of the decision semantics which takes place in the selection of dissemination strategies. The packages must be imported in the classes wishing to use them. Here is a list of the interfaces provided in each package and a brief description of their uses:

1.3.1 Application Interfaces

IApplication – The applications must implement this interface to force them to implement the methods called by the middleware to notify the user of a received message, acknowledgement or delivery failure.

IMiddleware – This interface is provided to allow an application to register or unregister with the middleware, and it also provides the interface operation to initialise the middleware.

IObjectSession – This is one of the most important interfaces presented to the application as it contains the send() and receive() primitives.

1.3.2. Dissemination Interfaces

IStrategy – All dissemination strategy components must contain a class which implements this interface. It forces the dissemination component to implement the disseminate() method which the middleware calls to use that component to disseminate the message. It also provides interface operations for the middleware to query the components for information used by the autonomic selection to make decisions.

IRegisterStrategy – This interface is provided to allow the dissemination strategy component to register with the middleware.

ITransmissionComponents – This interface represents the middleware transmission objects provided to the dissemination strategy component upon registration with the middleware. It provides interface operations to extract the message and beacon transmission objects.

IBeaconTransmission – This interface allows the dissemination strategy to obtain the host's current reachable list based on the middleware's beacon transmission.

IMessageTransmission – This interface is presented to the dissemination strategy components to allow them to use the middleware's message transmission function when the transmission of a message is required.

IMessage – This interface provides the dissemination components with the operations to get() and set() the parameters of the message structure.

IMsgHeader – This interface provides the dissemination components with the operations to get() and set() the parameters of the message header structure.

IReachableNode – This interface provides the dissemination components with the operations to extract the advertised information of a reachable node for use in transmitting a message to the host.

1.3.3. Autonomic Selection Interface

IAutonomic – This interface has been provided by MiDAS to allow for developers to modify the autonomic selection feature while still maintaining interoperability with the other components of MiDAS which use this feature.

1.4. Using the Applications

To run MiDAS and the sample applications provided, transfer the MiDAS folder to a working directory and then open a command prompt application. This could be MS-DOS if using Microsoft Windows, Terminal if using Mac OS or a Unix shell if using Unix. Then perform the following steps:

Step 1: To access the source files type:

cd directory/MiDAS

Step 2: Then to compile the source files using a batch file simply type:

Windows Installation: directory\MiDAS>mscompile
Mac OS and Unix Installation: directory\MiDAS>sh unixcompile

Step 3: Once it has been compiled, type the following to run MiDAS:

directory\MiDAS>java Launcher xxxxx

Where the xxxxx is the host ID you wish to use.

In the example provided, once the Launcher has been started, it initialises the middleware and creates two application types each containing a send and receive application box. The two application types used as examples are a weather report application and a cinema listings application. These applications are registered with the middleware and the user input boxes are displayed.

If we focus on one application type which is the weather report application, the input box portrayed in figure 1.4 (a) is presented to the user to specify the parameters of the send primitive. As can be seen, the user may specify the following fields:

- **Recipient ID:** ID of the host the sender wishes to transmit the message to. The use of a "*" may also be inputted representing the desire to send the message to all hosts in the network.

- **Recipient Location:** The location the recipient should be in to receive the message. May also be a "*" representing a desire to receive a message independent of the host's location.

- **Sender Location:** The location the sender host should be in to transmit the message. May also be a "*" representing a desire to receive a message independent of the host's location.

- **Expiration Time:** The expiration time (validity period) of the message in seconds.

- **Blocking Time:** The period the application should wait to receive an acknowledgement of successful delivery of the message.

- **Reliability:** The reliability value which expresses the desired reliability of the delivery process.

Note that if the user inputs a "*" as the recipient ID and a reliability value of 60%, MiDAS interprets this as an "anycast" send and the message will be sent to 60% of the hosts in the network.

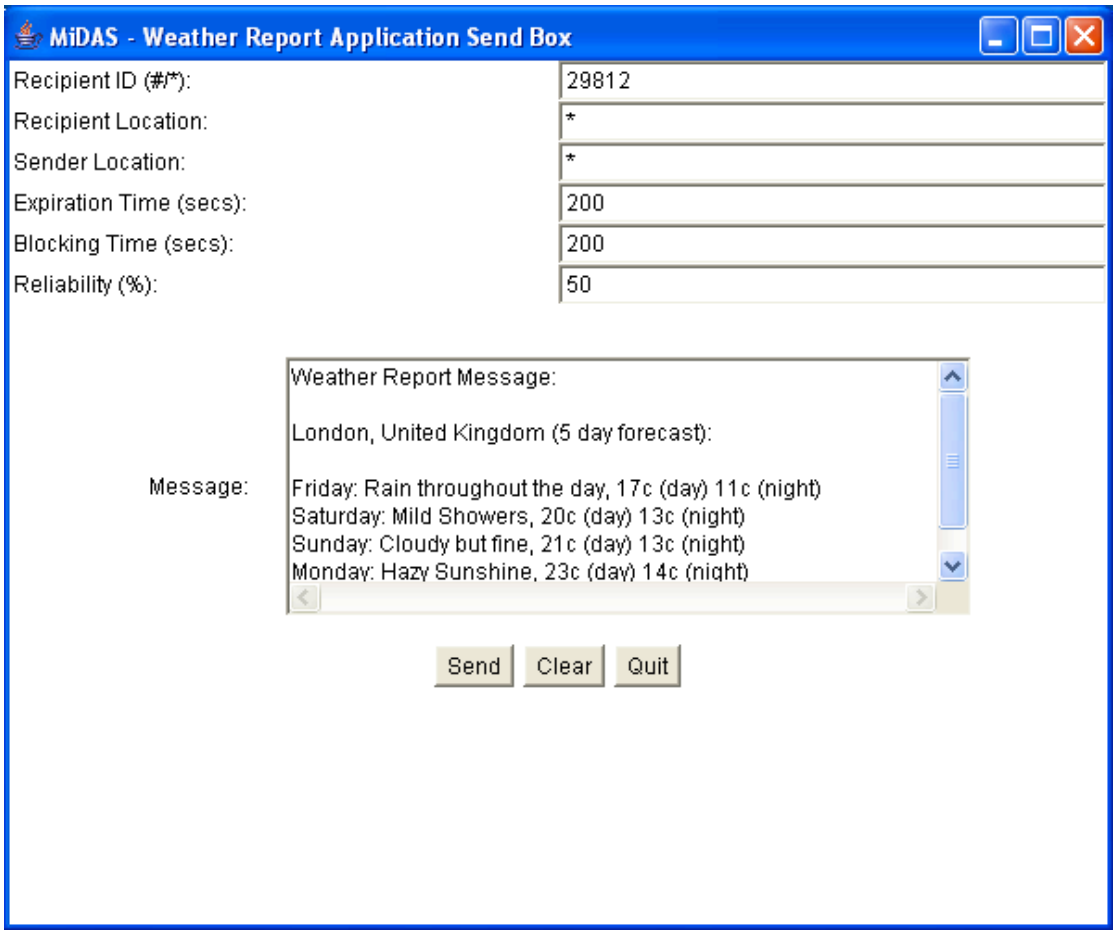

Figure 1.4 (a): Send box containing the parameters of the send primitive

The input box presented to the user to specify the parameters of the receive primitive is portrayed in figure 1.4 (b). As can be seen, the user may specify the following fields:

- **Sender ID:** ID of the host the receiver wishes to receive a message from. The use of a "*" may also be inputted representing the desire to receive the message from any host in the network

- **Sender Location:** Expresses the desire for the receiver to receive a message sent from a particular location. May also be a "*" representing a desire to receive a message independent of where the message was sent from.

- **Receiver Location:** The location the recipient should be in to receive the message. May also be a "*" representing a desire to receive a message independent of the host's location.

- **Blocking Time:** The period of time the application should wait to receive the message.

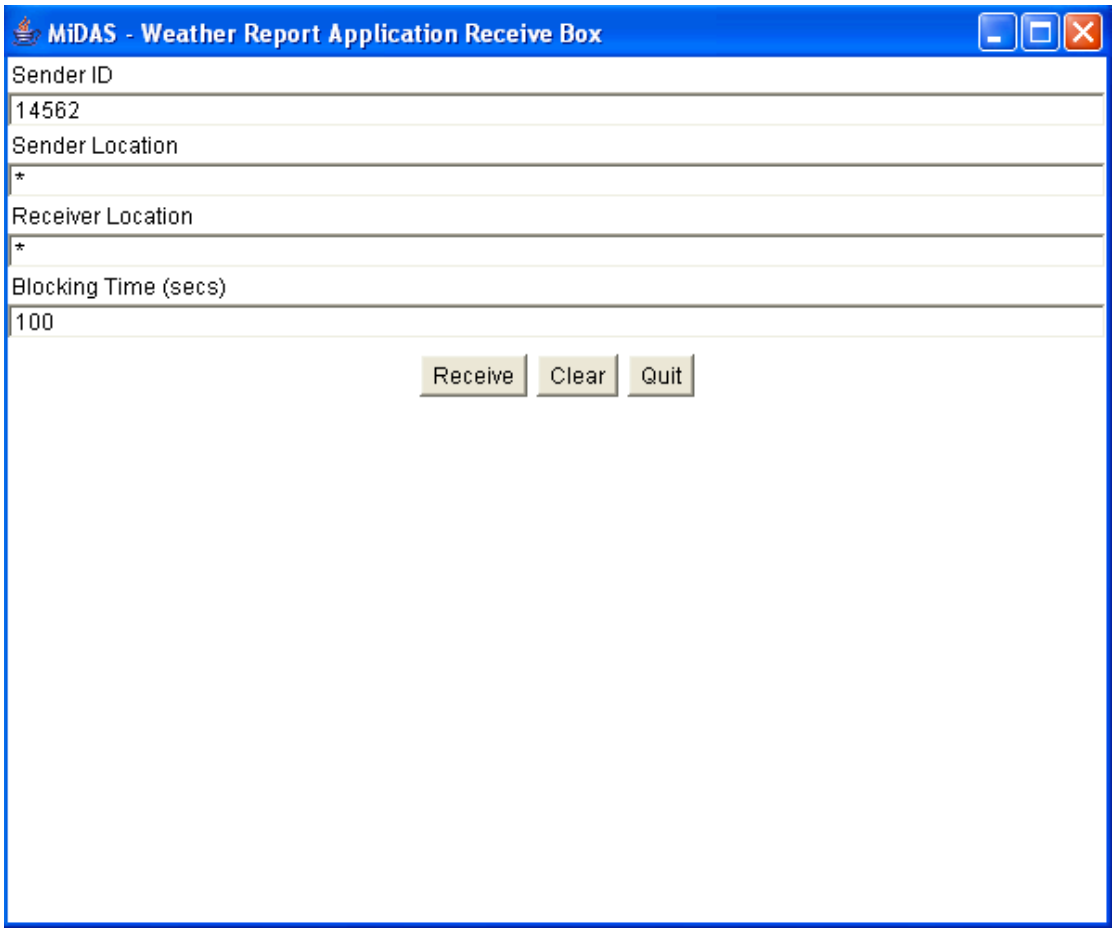

Figure 1.4 (b): Receive box with the parameters of the receive primitive

When a weather report has been issued on the sender side and a receive request has been made on the receiver side specifying the parameters which matches the send primitive, MiDAS checks if the message has been received and if so it alerts the user that the message has been received by a pop-up window displaying the message. This is presented in figure 1.4 (c).

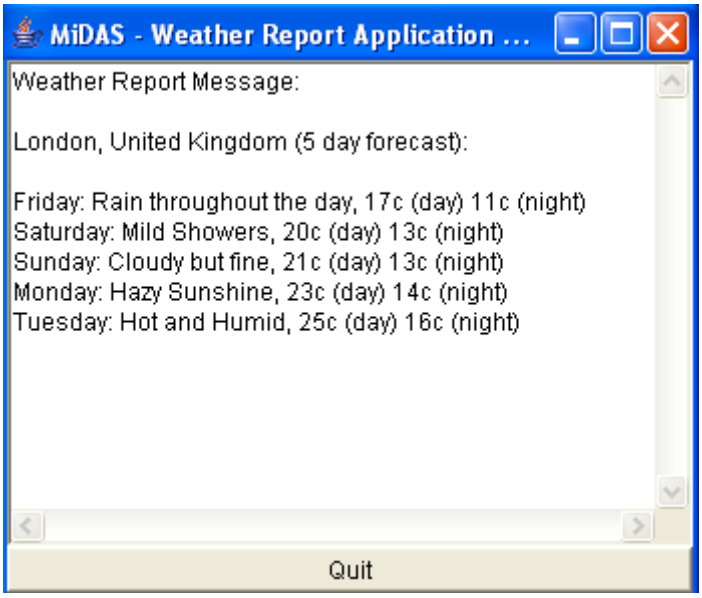

Figure 1.4 (c): Received Message pop-up window

When the message is transmitted on the sender side, if it is received by the recipient, then an acknowledgement is issued to notify the sender that the message had been delivered successfully. On reception of an acknowledgement, the user is notified via the use of a pop-up window expressed in figure 1.4 (d).

Figure 1.4 (d): Received Acknowledgement pop-up window

If an acknowledgement of successful delivery is not received by the sender within the specified "Blocking Time" parameter or if the message is not received within the specified time on the receiver side, then the user is notified of a failure. An example of this is expressed in figure 1.4 (e).

Figure 1.4 (e): Received failure notification pop-up window

When the user wishes to quit an application, they can click the "quit" button in the boxes in figure 1.4 (a) or 1.4 (b). This de-registers the application from the middleware, but does not stop the middleware as other applications may be using it.