A-GATE: A System of Relay and Translation Gateways for Communication among Heterogeneous Agents in Ad Hoc Wireless Environments

Leelani Kumari Wickramasinghe[1], Seng Wai Loke[2], Arkady Zaslavsky[2], and Damminda Alahakoon[1]

[1] School of Business Systems,
Monash University, VIC 3145, Australia
kumari.wickramasinghe@infotech.monash.edu.au
damminda.alahakoon@infotech.monash.edu.au

[2] School of Computer Science and Software Engineering,
Monash University, VIC 3145, Australia
seng.loke@infotech.monash.edu.au
arkady.zaslavsky@csse.monash.edu.au

Abstract. The devices in an ad hoc network are expected to perform network functionalities by themselves due to the absence of proper networking infrastructure. Generally the routing is multi-hop as nodes may not be within the wireless transmission range of each other. This paper describes a system named A-GATE to support the high-level communication needs of agents in such a network. Agents are used to support the interoperability among devices and the system is capable of handling heterogeneity in agent platforms. A-GATE proposes a novel routing mechanism for locating the intended recipient of a message. The system aims to be self-organizing and self-configuring to suit the dynamic nature of ad hoc networks.

1 Introduction

There is a surge of interest in mobile computing mainly due to the continued miniaturization of mobile devices and their ability to enable mobility. At the same time, wireless ad hoc networks are becoming quite popular due to their inherent feature of "the network" at user disposal. People wish to download a roadmap on their way so that they know what is available in close proximity to them or they wish to receive driving suggestions on the global positioning system (GPS) of their car [1].

A mobile ad hoc network is formed by a collection of mobile nodes. There is no established networking infrastructure for the mobile devices to rely on. As a result, all the networking functionalities have to be performed by the nodes themselves. Mobile devices take an active role in creating a network infrastructure and routing of data. Cooperation among nodes is an essential requirement: for example, if two nodes need to communicate, intermediate nodes are expected to forward the messages [1, 2].

When developing an application for ad hoc environments, there are challenges to be addressed [3]. The topology of the ad hoc network is dynamic and nodes join and leave the network spontaneously. There is no central server that knows the nodes' current locations and the networks to which the devices shifted. The cost of communication is high as the devices have a limited battery power.

Due to the distributed and dynamic nature of mobile ad hoc networks and the potential need for proactive, spontaneous and intelligent interaction, agent technology has been considered promising for building applications in such ad hoc wireless environments [3]. With the spread of distributed systems, a large number of agent systems has been developed. Each agent system has a different agent platform and it is really difficult to get agents on heterogeneous systems to work together. Most agent systems require homogeneity in agent systems for agents to communicate and migrate. When it comes to agents on ad hoc networks, they should be capable of interacting with agents on any other device no matter what the communication language or agent platform of the other agent.

Agent interactions are handled by standardizing the language of communication. KQML [4] and FIPA [5] are two such specifications. Before communication starts agents should have priori knowledge of where the other agents reside. It is an extra overhead if the communication initiation agent has to locate the target agent. Infrastructure support should be capable of locating the target agent, keeping the source agent away from network level issues.

The aim of this paper is to make it possible for agents running on devices interconnected ad hoc and wirelessly to communicate without concern for underlying network level issues or heterogeneity of agent platforms. The proposed system, A-GATE, will take responsibility for locating the target agent and delivering the message successfully, regardless of the above heterogeneity.

One possible scenario would be the exchange of business cards among the participants in a conference room using agents residing on mobile devices. Another scenario would be an infrastructure-less office environment composed of mobile devices. Communication among devices can be handled by agents residing in each device.

The approach is explained in the following sections: Section 2 describes some existing systems that handle heterogenous agent platforms and applicability of mobile agents for the communication needs of ad hoc networks. The proposed system, A-GATE is described in detail in Section 3. Experiments done to evaluate the applicability of A-GATE as a generic communication system for an ad hoc wireless network are presented in Section 4. Section 5 discusses the conclusion and future work.

2 Related Work

In most agent systems, agents require a homogeneous platform to migrate. In [6] is described an approach where migration of agents in a heterogeneous network is possible by way of a blueprint. The blueprint consists of the agent's functionality and its state. The receiving platform regenerates a mobile agent as it migrates to its new location. In the A-GATE system only agents on heterogeneous systems are needed to communicate. The recreation of agents consumes time and resources whereas communication between agents should be faster and simple.

A model based on middleware or an interface between agents and platforms is proposed in [7]. This layer is visible to an application programmer on one side and on the other side there is a platform dependent layer.

There are guest interfaces of one per platform. According to the platform, the required java classes have to be downloaded. Our approach is quite similar to this one but the required agent platforms are loaded at configuration time as it is much faster with real time communication. Our goal is to provide a fast and secure communication support for agents in mobile devices.

A network protocol called Agent Platform Protocol (APP) is designed for agent interactions among heterogeneous platforms [8]. In this approach, agents are free from the management of network level issues. It uses peer-to-peer communication. But when agents are deployed in a mobile network, direct peer-to-peer communication among any two peers is not possible due to the short range coverage of the underlying wireless protocols. Peers need to forward messages via other peers as A-GATE facilitates.

The advantages of using mobile agents for the communication needs of an ad hoc networks is described in [9]. Mobile agents can function as "wrappers" on messages, which enable the messages to propagate themselves to the intended destinations. But the transmission overhead associated with mobility is high. Mobile devices need interoperability with minimum pay load in scenarios such as exchange of business cards in a conference or infrastructure-less office environments. The system proposed in this paper has a store and forward concept with a minimal transmission overhead.

3 Proposed System

A-GATE is based on the concept of a gateway (gw), as illustrated in Fig. 1, where agents on mobile devices interact with a gateway which is located within the reachable area of the device, to communicate with any other agent. The importance of this concept is that both the gateways and agents reside on mobile devices and gateways come into existence only when they are required, as will be described in Section 3.5.1

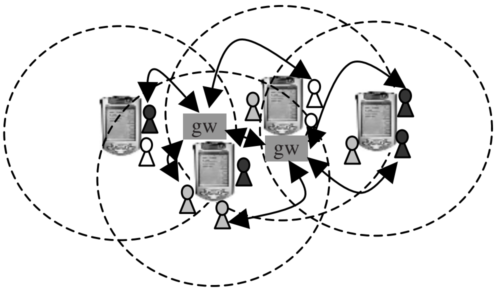


Fig. 1. A-GATE System Architecture

The block diagram of a gateway is shown in Fig 2. It consists of 5 components: Message Accepting Relay Agents, Message Extractor, Message Translator, Message Re-builder and Message Dispatching Relay Agents. The gateway application itself spawns relay agent platforms depending on the number of agent platforms the system needs to support while there is only one generic process known as METR consisting of Message Extractor, Message Translator, and Message Re-builder units.

A gateway provides two main functionalities to the agents on mobile devices: a message translation service and a message relaying service. When an agent sends messages in its own language without considering the language of the target agent, message translation service takes care of the translation to a language understood by the destination agent. The message relaying service consists of Message Accepting Relay Agents and Message Dispatching Relay Agents (see left and right hand corners of Fig 2). The Message Accepting Relay Agent accepts messages from a source agent or an intermediate message routing gateway while the Message Dispatching Relay Agent dispatches the message to the intended recipient.

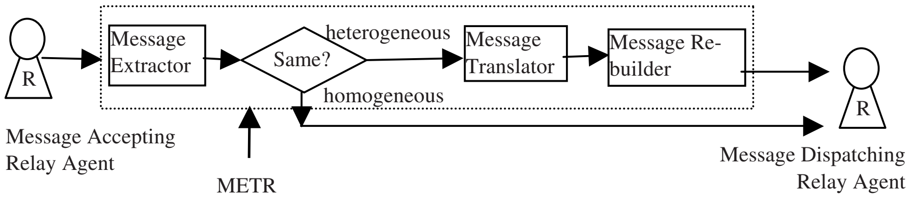


Fig. 2. Block Diagram of the Gateway Application

Each gateway maintains two registers: agent registry and gateway registry. Agent registry maintains information related to the agents reachable from the gateway while the gateway registry maintains information related to other gateways. The agent registry stores reachable agents' identifications, their platforms and communication languages while the gateway registry stores reachable gateways' identifications. The process to identify reachable agents and gateways is described in Section 3.4. The main units of the application are described in Sections 3.1 and 3.2.

3.1 Message Accepting and Dispatching Relay Agents

Generally, agent platforms have an inbuilt messaging system to interact with agents from homogeneous systems. What is needed is a system in which heterogeneous agent systems communicate. To accomplish this task, the Gateway application spawns relay agent platforms to match the agent platforms used by the two communicating mobile devices. Relay agents are there to accept messages or deliver messages to the source and destination agent respectively. The Message Accepting Relay agent accepts the message from the source agent and forwards it to the Message Extractor. Once the message is rebuilt, the Message Dispatching Relay Agent forwards the message to the destination agent. The advantage of having platform specific relay agents is that, when a new agent platform is introduced to the system it is just a matter of deploying an agent of that platform as a relay agent without needing any modifications to the generic gateway application.

3.2 Message Extractor, Translator and Re-builder (METR)

Once a message is accepted by a Message Accepting Relay Agent, it is forwarded to the Message Extractor unit. This unit extracts the message and gets the recipient address. Then it checks whether the recipient is a reachable agent by going through its routing table (information about the routing table can be found in Section 3.4.2). If the recipient is a reachable agent, it checks for the agent platform of the recipient. If the agent platform of the sender is different from that of the receiver, the message is sent to the Translator unit. Otherwise, the message is directly sent to the Dispatching Relay Agent. Message Translator takes the content of the message and translates it to a format understood by the recipient agent considering the agent platform of the recipient. Once this is done, the message is forwarded to the Re-builder unit which attaches the appropriate headers to the message and forwards it to the Message dispatching relay agent. If the recipient is not reachable, the message is forwarded to a gateway which is capable of delivering the message. Identification of such a gateway is done by going through the contents in the routing table. But in this case, before forwarding the message, the Message Extractor unit adds another optional field called “sender’s agent platform” to the original message, which is needed by any other gateway to deliver the message in a format understood by the receiving agent. The complete algorithm of METR is listed below:

```
extract the sender address
extract the recipient address
check for the recipient address in the agent registry
if an entry found
    check for the optional field “sender’s agent platform”
    if field found
        extract the senders platform (P1)
    else
        check whether sender got any entry in the agent registry
        if an entry found
            get the senders platform (P1) from the list entry
        else
            drop the message
    check for the agent platform of the recipient (P2)
    if ( P1 != P2)
        translate the message to the P2 platform
        dispatch the message to the Message re-builder
    dispatch the message to the Message Dispatching Relay Agent
if an entry is not found
    go through the routing table
    check for a gateway which can route the message to the recipient
    if a gateway found
        sends the message to that gateway
    else
        drop the message
```

A somewhat similar algorithm is proposed in [10], where mobile hosts actively modify their trajectories to transmit messages. It involves trajectory modification of each host to approach the immediate host within the transmission range of the host, which is applicable for devices which have automated moving capabilities (for example moving cars and robots).

3.3 Address Schema

A unique way of identifying an agent on a mobile device is required. Each agent on a mobile device is given a unique identity according to the concepts in cellular phones, where each phone is assigned a unique identity referred to as Numeric Assignment Module (NAM) [11]. There can be number of agents acting on a single device. Therefore, an identifier for an agent would be:

Identifier for the mobile device + some identifier for the agent (x)

The ID stored in a digital certificate [12] could be considered as an identifier for the mobile device as security can also be built upon it.

The agent unique identifier could be matched to IP:port relationship in networking. There are well known port addresses [13] such as 80 for web 21 for ftp and so on. Value of x could be based on this well known port address technique in networking. "Well known agents" are required by relating agents to the services they perform such as information retrieval, database accesses and exchanging information: for example, if the mobile user has a digital certificate for his email address user1@company.com and the well known agent for information exchange is 20, then the unique identifier of the information exchanging agent on user1's mobile device would be user1@company.com:20

3.4 Communication Process

Communication is achieved through agents and gateways residing on mobile devices. Agents are originators and consumers, while gateways are intermediate processes to transfer messages from originators to consumers. For the system to function properly, agents should have an understanding about which gateways they can directly communicate, while gateways should know about other reachable gateways. Facilitating this, agents send periodic network broadcasts in a neighbour to neighbour fashion on a pre-identified port address that the gateways and other agents monitor. Similarly, gateways send periodic broadcast messages on another pre-identified port address that only the gateways monitor.

3.4.1 Agent-Gateway Communication. A broadcast sent by an agent is considered as a notification of the existence of the agent. It consists of agent identification, agent platform and communication language. Once the broadcast is received by the gateways, they reply with their identifications. Agent stores this gateway identification to be used in future communication needs. If the agent receives replies from more than one gateway, it can either store the address of only one gateway or store all the gateway addresses so that, if one gateway goes down or changes its location, it can use the next gateway to accomplish its communication needs. In the latter case, deciding factors would be the basis of service such as First Come First Served (FIFS) or the signal strength of each gateway. Consequently, agents reply to the chosen gateway or gateways so that gateways can insert or update the entries in their agent registries.

3.4.2 Gateway-Gateway Communication. The broadcasts sent by gateways consist of the gateway identification and information about the reachable agents and gateways. This broadcast is considered to be a technique of exchanging routing tables among gateways. A routing protocol for ad hoc networks based on routing tables is presented in [14]. All the gateways which receive this message update their routing tables accordingly, which means each gateway on the ad hoc network has a complete picture of how the agents

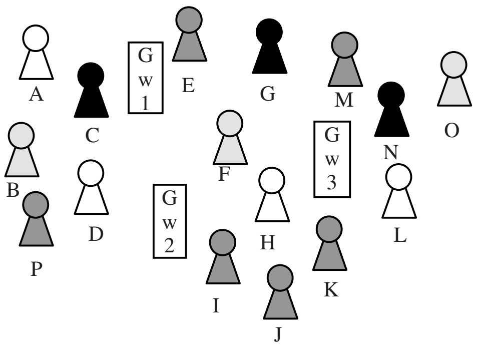


Fig. 3. Agents and Gateways in a Network

reside on the network at a given time. Gateways are proactive, meaning when a sender needs to send a message the next hop to the intended recipient is already known by the gateways. Maintaining the complete routing table is a reasonable approach as there are a limited number of devices on the ad hoc network.

Fig. 3 outlines the concept of routing tables. If it is assumed that the agents C and D are reachable by gateways 1 and 2, agent G is reachable by gateways 1 and 3, agent H is reachable by gateways 2 and 3 and agent F is reachable by all three gateways. In addition, gw1 and gw2 are reachable and gw2 and gw3 are reachable by each other.

Routing tables of gateways Gw1, Gw2 and Gw3 are presented in Tables 1, 2 and 3 respectively. As shown in Table 1, Gateway 1 (gw1) is within the reachable area of agents A, C, D, E, F and G, listed under "Direct agents". Gateway gw2 is located within the reachable area of Gw1; therefore it is listed under the "Direct gateways". Agents B, F, I, H, J and P can reach gw2 while agents M, N, O, L and K can reach gw3. They have two separate entries as "Via gw2 agents" and "Via gw3 agents". Gateway gw3 is contactable via gw2; so gw3 is listed in "Via gw2 gateways". The other two routing tables are also based on this concept.

Due to the dynamic nature of the ad hoc network, gateways and agents can appear and disappear with time and they can move to new locations. Gateways exchange their routing tables periodically so that each receiving gateway can update their routing tables to suit the current state of the ad hoc network.

3.4.3 End to End Message Delivery. When an agent needs to send a message, it sends a delivery request message to the gateway it is registered with, or to one of the gateways it got registered with. The message consists of the sender, intended recipient and the content to be delivered. As with any other device in an ad hoc network, gateways may appear and disappear. Therefore, when a gateway receives a delivery request, it sends an acknowledgement back to the sender to indicate it has received the delivery request. But the acknowledgement does not guarantee an end-to-end delivery of the message. The delivery of the message is handled as described in the Sections 3.1 and 3.2.

Table 1. Routing Table of Gw1

Direct agents	A, C, D, E, F, G
Direct gateways	gw2
Via gw2 agents	B, F, I, H, J, P
Via gw3 agents	M, N, O, L, K
Via gw2 gateways	gw3

Table 2. Routing Table of Gw2

Direct agents	B, D, P, C, F, I, H, J
Direct gateways	gw1, gw3
Via gw1 agents	A, E, G
Via gw3 agents	G, M, N, O, L, K

Table 3. Routing Table of Gw3

Direct agents	G, M, N, O, L, K, H, F
Direct gateways	gw2
Via gw1 agents	A, C, E
Via gw2 agents	B, D, P, C, I, J
Via gw2 gateways	gw1

3.5 Gateways

Gateway is the main application which handles the communication process. It resides on the mobile device and come into existence only when needed as described in Section 3.5.1. Once a gateway comes into existence, how to retain it for further communication needs is explained in Section 3.5.2 and deciding factors for a gateway to shut down is explained in Section 3.5.3. Gateways can be used for load balancing as detailed in Section 3.5.4.

3.5.1 Creation of Gateways. When an agent needs to communicate, it usually generates the message and sends it to a gateway it is registered with. But if the agent has not received a reply from any of the gateways for its initial broadcast, it will retry with the same broadcast in random time delays. After three such consecutive attempts, if there is no reply and if there is any mobile device within the reachable area, the device itself can voluntarily become a gateway.

Using an arbitrary node 'X' intending to communicate with the node 'Y', if the node X has not registered with a gateway at time t1, it sends a network broadcast informing it of its existence. Then it waits for some time to receive a reply from the gateway. After waiting for a random time period, if it does not get any reply it will do another broadcast for the second time at time t4. If still no reply, then a third broadcast is done at time t8. If the same condition remains, it will listen to check whether there is any notification of existence messages from other devices in the network. If there are any such devices, X would voluntarily become a gateway and send a network broadcast informing its existence as a gateway.

Also if a node detects there exist two gateways which do not know about each other then that node should become a gateway. This is needed in order to facilitate the communication between the nodes which have got registered with one of the two gateways. To determine whether one gateway knows about other one, the node can try to send a message to one gateway via other one. If the first gateway drops the message, then that means there is no route from that gateway to the second one and vice versa and the node itself can become a gateway.

3.5.2 Retention of Gateways. Once a gateway is formed it is better to retain it for some time as a gateway for the other devices on the network to communicate with. But in an ad hoc network where all the nodes do not belong to the same authority, each node tries to maximise the benefits it receives from the network. Perhaps nodes are not willing to provide gateway functionalities to the other nodes. Nodes might become selfish to save limited resources such as battery power, memory and CPU cycles. But, considering the network as a whole, gateway functionality is essential.

The nuglet counter concept described in [15] can be considered as a technique for retaining gateways. If a device voluntarily becomes a gateway it will earn 3 nuglets. That means it can send 3 of its own messages without acting as a gateway to other agents. As long as the gateway has enough nuglets it can send its own messages. Whenever it sends its own message the nuglet counter gets decreased. But it can earn more nuglets by acting as a gateway to other messages. This would become useful if the device acting as a gateway knew that it may want to send more of its own messages in the near future. In that case, till the time comes, it can collect nuglets by being a gateway to the neighbouring devices.

3.5.3 Shut down of Gateways. Two situations have been identified for shutting down

a) Isolated gateways: In an ad hoc network, as each device can move, gateways can get isolated. If a gateway does not receive any notification of existence broadcasts it is an indication as to no neighbouring devices within the reachable area of that gateway. Once a gateway identifies itself as an isolated device it can terminate its functionality and shut down.

b) Redundant gateways: There can be situations where more than one device is acting as a gateway to the same set of nodes as shown in Fig 4. As all the agents are within the reachable area of each gateway, it is enough to have one gateway rather than two. Gateways themselves can identify this issue by going through their routing tables. In that case, they can negotiate with each other and come to a conclusion as to which gateway to shut down. Metrics to consideration could be the available resources, the processing power and memory.

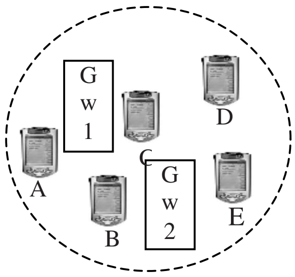


Fig. 4. Redundant Gateways

3.5.4 Load Balancing Using Gateways. If it is assumed that, after the negotiation, gw2 went down, but later gw1 finds that there is a huge traffic between agents A and E which is difficult to be handled alone by gw1, it can send a disaster message indicating it is overloaded. In that case gw2 or some other device can come into existence to balance the load.

4 Experiments

To evaluate A-GATE as a generic agent interaction system, the gateway application was implemented for the process of exchanging business cards in an ad hoc network. Heterogeneous agent systems, Grasshopper [16] and Kaariboga [17] were used as the test agent platforms. Grasshopper is a commercial agent platform while Kaariboga is an open source agent platform.

4.1 Implementation

The system consists of a Message class which needs to be used by the agent systems to send messages. This Message class is portable to many programming languages. At the initial implementation level, message class contains three main attributes: sender, receiver and content.

The communication between platform-specific relay agents and gateway application is handled via TCP/IP sockets. The system is implemented on java platform using java socket programming [18] for communication. Relay agents and gateway application listen on specific ports so that whenever a relay agent receives a message, it can forward it to METR. Similarly once the address resolution for routing and translations are done at METR, it can forward the message to the correct platform specific agent on the receiving side. Fig 5 is a simplified diagram of the Grasshopper and Kaariboga test system.

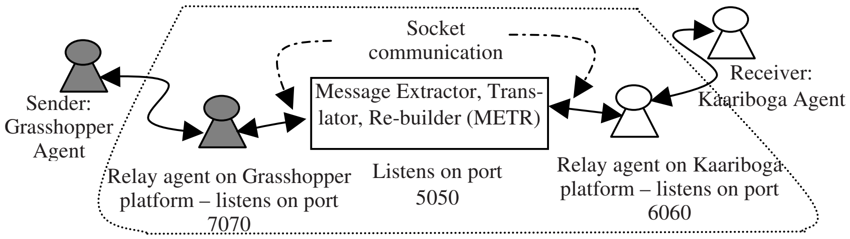

Socket communication
Sender: Grasshopper Agent
Message Extractor, Translator, Re-builder (METR)
Receiver: Kaariboga Agent
Relay agent on Grasshopper platform – listens on port 7070
Listens on port 5050
Relay agent on Kaariboga platform – listens on port 6060

Fig. 5. Test System with Grasshopper and Kaariboga Agents

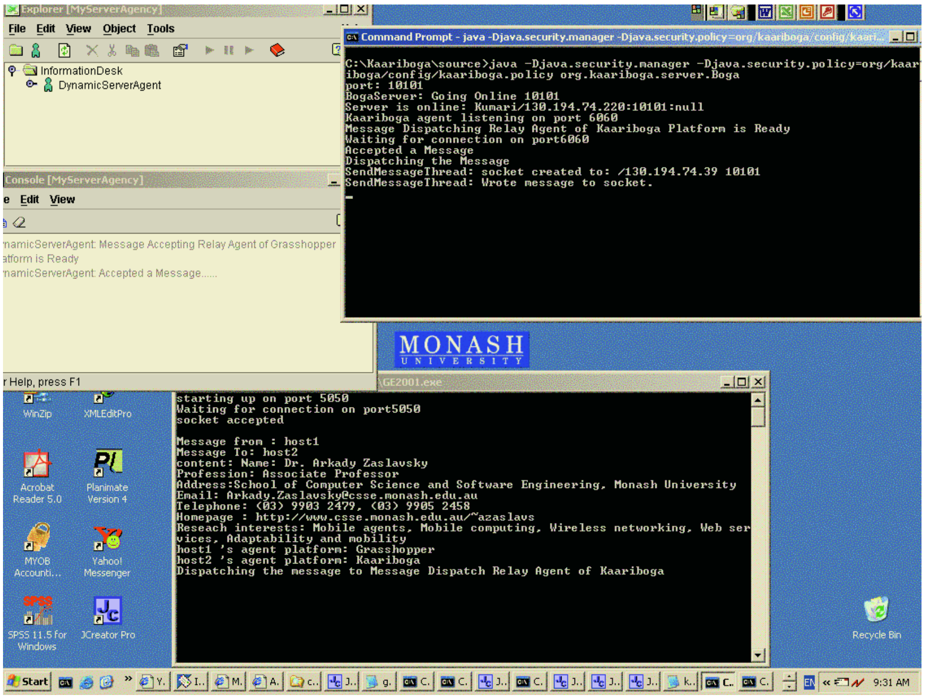
For testing, a generic message class was imported to both agent platforms. Using the agent creation concepts in Grasshopper platform, a dynamic client agent and a dynamic server agent were created to act as the message sender and the corresponding message accepting relay agent respectively. Changes had to be made to IDynamicServerAgent.java and IDynamicServerAgentP.java to use the generic message class. Similarly a message receiver and message sender agents were created in the Kaariboga platform. KaaribogaMessage.java class was configured to handle the messages of our generic message class. Fig 6 illustrates Gateway application, where a Grasshopper agent on one device sends the business card details to a Kaariboga agent in another device.

4.2 Performance Testing

To evaluate the performance of A-GATE, it was tested with the traditional client server architecture as it can support agent communication. The result was encouraging as both the systems take the same amount of time to send a message from a source agent to destination agents. In addition, the byte overhead of the generic message class is limited to the two address fields: senders and receivers, which is acceptable with any communication system. Currently the system is being tested in a network with around 50 nodes (some of which are mobile) and the performance has to be evaluated as to what happens when the number of gateways is increased. There should be a threshold number of gateways after which the performance of the system would not be significant.

5 Conclusion and Future Work

The A-GATE system, as presented in this paper can be used for the agents on an ad hoc network to communicate without concern for the underlying network level issues or heterogeneity of agent platforms. The technique used for creating, retaining and shutting down gateways provides a novel solution to the infrastructure-less ad hoc networks where the devices on the network have to handle all the required networking functionalities. We believe our work complements existing work on low level ad hoc networking, as our work is at the application or agent level, where more sophisticating reasoning concerning when and how to relay and high level semantics of translations can be considered.



```
C:\Kaariboga\source>java -Djava.security.manager -Djava.security.policy=org/kaar
iboga/config/kaariboga.policy org.kaariboga.server.Boga
port: 10101
BogaServer: Going Online 10101
Server is online: Kumari/130.194.74.220:10101:null
Kaariboga agent listening on port 6060
Message Dispatching Relay Agent of Kaariboga Platform is Ready
Waiting for connection on port6060
Accepted a Message
Dispatching the Message
SendMessageThread: socket created to: /130.194.74.39 10101
SendMessageThread: Wrote message to socket.
_
```

```
namicServerAgent: Message Accepting Relay Agent of Grasshopper
atform is Ready
namicServerAgent: Accepted a Message......
```

```
starting up on port 5050
Waiting for connection on port5050
socket accepted

Message from : host1
Message To: host2
content: Name: Dr. Arkady Zaslavsky
Profession: Associate Professor
Address:School of Computer Science and Software Engineering, Monash University
Email: Arkady.Zaslavsky@csse.monash.edu.au
Telephone: (03) 9903 2479, (03) 9905 2458
Homepage : http://www.csse.monash.edu.au/~azaslavs
Reseach interests: Mobile agents, Mobile computing, Wireless networking, Web ser
vices, Adaptability and mobility
host1 's agent platform: Grasshopper
host2 's agent platform: Kaariboga
Dispatching the message to Message Dispatch Relay Agent of Kaariboga
```

Fig. 6. Screen Output of Gateway Application

The immediate target is to evaluate the performance of the system in a large network of mobile devices connected via short-range wireless networks. The amount of flooding in the network when there is large number of devices in the network needs to be measured.

Future work will include building a Semantic Translator Engine and a Content Translator Engine to the Message Translator unit of the Gateway. Semantic Translator Engine is to handle multilingual messages where both the performative and content need to be translated. Once it is done agents can communicate in their native language, be they English, Chinese or Japanese. Content Translator is to handle the contents of the message, for example to translate the "Prolog facts" in a message to "KIF facts".

References

1. Perkins, C.E., *Ad hoc networking*. 2001, Boston: Addison-Wesley. xii, 370.
2. Corson, S., J. Freebersyser, and A. Sastry, *Mobile Networks and Applications (MONET)*. Special Issue on Mobile Ad Hoc Networking, 1999.
3. "Agents in Ad Hoc Environments", *http://www.fipa.org/docs/input/f-in-00068/ f-in-00068A.htm*

4. Finin, T., Y. Labrou, and J. Mayfield, *KQML as an agent communication language*, in *Software Agents*, J. Bradshaw, Editor. 1997, MIT Press: Cambridge. p. 291-316.
5. "Foundation for Intelligent Physical Agent Specification", *http://www.fipa.org*,
6. Brazier, F.M.T., et al. *Agents, interactions, mobility and systems: Agent factory: generative migration of mobile agents in heterogeneous environments*. in *2002 ACM symposium on Applied computing*. 2002. Madrid, Spain: ACM Press New York, NY, USA.
7. Magnin, L., et al. *Our guest agents are welcome to your agent platforms*. in *2002 ACM symposium on Applied computing*. 2002. Madrid, Spain: ACM Press New York, USA.
8. Takahashi, K.i., G. Zhong, and D. Matsuno. *Interoperability between KODAMA and JADE using Agent Platform Protocol*. in *The First International Joint Conference on Autonomous Agents and Multi agent systems*. 2002. Italy.
9. Kotz, D., et al., *AGENT TCL: targeting the needs of mobile computers.* Internet Computing, IEEE, 1997. **1**(4): p. 58-67.
10. Qun Li and D. Rus. *Message relay in disconnected ad-hoc networks*. in *Mobility and Wireless Access Workshop, 2002. MobiWac 2002. International*. 2002.
11. Bates, R., *Cellular Communications*, in *Wireless networked communications: concepts, technology, and implementation*. 1995. p. 73-95.
12. Feghhi, J., J. Feghhi, and P. Williams, *Digital Certificates:Applied Internet Security*. 1999: Addison-Wesley.
13. "Port Numbers", *http://www.iana.org/assignments/port-numbers*,
14. Basagni, S., et al. *A distance routing effect algorithm for mobility (DREAM)*. in *Fourth Annual ACM/IEEE International Conference in Mobile Computing and Networking (MobiCom)*. 1998. Dellas,TX.
15. Buttyan, L. and J.-P. Hubaux, *Stimulating cooperation in self-organizing mobile ad hoc networks.* MONET Journal of Mobile Networks, 2002.
16. "Grasshopper - The Agent Platform", *http://www.grasshopper.de./*,
17. "Kaariboga Mobile Agents", *http://www.projectory.de/kaariboga/*,
18. Heaton, J., *Programming Spiders, Bots, and Aggregators in Java*. 2002: Richard Mills.