MAGNET: AD-HOC NETWORK SYSTEM BASED ON MOBILE AGENTS

Nobuo KAWAGUCHI[†], Katsuhiko TOYAMA [†‡] AND Yasuyoshi INAGAKI[†]

†*Department of Information Engineering, Graduate School of Engineering, Nagoya University*
‡*Center for Integrated Acoustic Information Research (CIAIR)*
Furo-cho, Chikusa-ku, Nagoya, 464-8603, Japan

E-mail: {kawaguti|toyama|inagaki}@nuie.nagoya-u.ac.jp

This paper proposes an ad-hoc network system based on mobile agents. We regard some special mobile agent as an implementation of the network protocol. Mobile agents can move over the network while being carried by protocol agents. Implementation of the protocols by the mobile agents enables the dynamic extension of the network. We introduce **agent replication** for the simple development of the inter-agent communication. **Agent hierarchy** enables the component software design. **Link monitoring** agent enables the adaptations to the dynamic network change. We also describe the implementation of the ad hoc network protocol DSR using our prototype system MAGNET which is based on the framework. MAGNET is developed by Java and working both on the ad hoc network emulation environment and on the real infrared device in Win98/CE/Java OS PDA(BossaNova).

1 Introduction

Recently, mobile computing becomes popular in favor of the small sized, light-weight PDAs and notebook computers. Sometimes, it can happen that several PDAs and computers gather at the same place such as the conference site, or classrooms. However, without any centralized manager nor additional administrative work, it is not easy to exchange information between the portable computers directly under the mobile environment. The *ad hoc network* is a network which is constructed on demand to enable these communications. Several routing and management protocols for the ad hoc network are proposed so far[1, 2, 3]. These methods are evaluated by the simulation environment and shown to work well in some settings[11].

However, there are various situations in the communication under the mobile environment. The number of the hosts and the active links, host mobility, data length, traffic, communication speed, and quality are quite different depending on situations. Furthermore, desired function of the ad hoc network depend on who uses the network and where the network is utilized. So, it is almost impossible that the single protocol for the ad hoc network becomes the

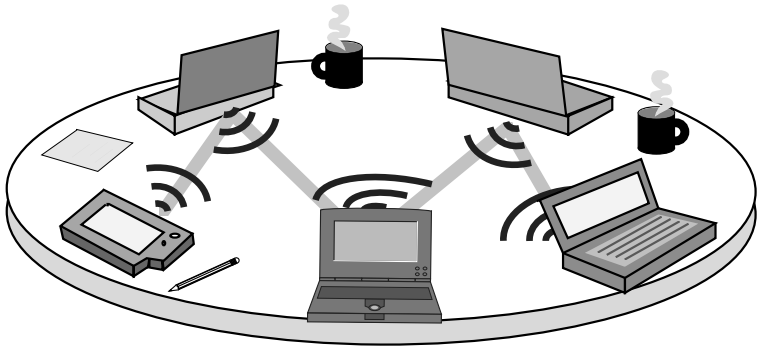

Figure 1. The ad hoc network on the table

optimal in all situations. Furthermore, it isn't realistic that preparing several protocols into all mobile hosts for various situations.

Recently, *the active networks* [4, 5, 6] have been proposed for dynamically configuring routers and switchs for the packet exchange based networks such as the Internet. Although these systems utilize a packet which contains a program for flexible network configuration, they do not consider about the ad hoc situations under the mobile environment.

We believe, under the nomadic situation, the network system should have the following two properties, (1)Dynamic adaptation to network topology change, and (2)Dynamic extension of protocols and services. The ad hoc network protocols developed so far have first property, but not second one. Active network systems have second property, but they do not mainly focused on first one.

This paper proposes a framework to utilize the mobile agent technology to construct the ad hoc network for adapting various situations and to enable the dynamic extension of the system. In our framework, we regard each packet as a set of mobile agents and each communication between hosts as a migration of the mobile agents. This key feature enables the framework to satisfy the required two properties. Ad hoc network protocols can be implemented as hierarchical structures of mobile agents for adapting the network topology change. To extend the protocol, it is not required to reinstall the protocol stacks into all mobile hosts. Just deploy the new extended protocol agent to the mobile hosts, each host can manage the new protocol on demand. Additionally, hence services on the mobile hosts are provided by the residential agents in the host, it can be easily extended or replaced by other mobile agents.

We have developed a mobile agent network system named MAGNET as a prototype system based on our framework. MAGNET is implemented by Java and mobile agents are Java objects. To exemplify the expressive power of the

framework, we have implemented the ad hoc network protocol DSR[3].

MAGNET is mainly designed to use under the small area network for notebook computers and PDAs. We adopt the infrared as a communication medium. For the medium such as an infrared device, conventional hard-wired protocol such as TCP/IP is not suitable for the ad hoc networking. So, our system does not directly depend on any existent network protocols. The link layer and network layer are managed by the mobile agents and the agent support link library.

We developed a wireless network emulator which has a GUI to simulate the dynamic change of the ad hoc networks. The emulator ease the development of mobile agents and simplify the test of the adaptation of the system. We also developed the system based on the real wireless communication, which works on Windows 95/98/CE and Java PDAs with additional infrared library using Java Native Interface. The same agent code can run on real environment in favor of compatible library.

This paper consists of the following sections. In section 2, we briefly explain the ad hoc networks and our experiences. Section 3 describes our framework of the mobile agent network. Implementation of the framework is presented in section 4. Section 5 shows an implementation of DSR protocol. Section 6 presents the related work. Section 7 concludes the paper.

2 Ad hoc Network

The ad hoc network is a network which is constructed on demand to enable the communications between the mobile hosts equipped with the wireless devices. It has following characteristics.

1. No centralized manager.
2. Before the construction of the ad hoc network, each mobile host has no information about other hosts nor links (No previous administrative work).
3. Network topology is changed dynamically by movement of hosts.

Different concepts of management and utilization are required for the ad hoc network operation. Currently, we employ the following assumptions for simplicity.

1. *Each mobile host has a unique host ID.*
   Without unique ID, each host should perform the anonymous network computations[7].

2. *Links are bidirectional.*
   It is difficult to distinguish between the unidirectional link and the link breakage just after the migration of the mobile agent.
3. *Mobile hosts can recognize the adjacent hosts.*
   Without this feature, it is impossible to start a communication with other hosts.

The dynamic network topology change and the communication error make the management of the network into very complicated one. We had tried to develop a distributed communication protocol for infrared ad hoc networks. In this protocol, the system is required to handle the complex event table[1] to adapt the dynamic change of the network topology. Most of the complexity of the protocol comes from the design policy which is based on status of hosts. For example, when host A send some message to host B and then wait reply, but if host B disappear from the network just after the acceptance of the message, host A may wait until recognizing it. This type of waiting mistakes occurs frequently in the network.

From this experience, we learned that using state and message based approach is not suitable for the distributed computing which have communication errors and dynamic changes. Mobile agents do not suffer from this kind of problems. Although they have the execution status, they do not have to assume other agents status nor waiting other agents. It can move to other hosts by itself. In the next section, we introduce the framework for utilizing mobile agents to form an ad hoc network.

3 Mobile Agent Network

This section explains our framework of the mobile agent system for constructing the ad hoc network. The core idea of our framework is using the mobile agent as a basic unit of the system. In this framework, each packet on the network is a set of mobile agents. Furthermore, all routers are also mobile agents [2]. Network links between the host and the other hosts are monitored and maintained by Link Manager agents. Routing is performed by mobile agent itself or under the communication with the stationary routing agents.

All mobile agent systems developed so far[8, 9, 10] are based on existent network systems. However, our framework do not based on any other network systems. Mobile agents itself form a network and provide the routing services.

[1] The event table has over 70 columns.

[2] Basically, routers do not move. Occasionally, routers are replaced or extended by other agents. This kind of agents are called stationally agents.

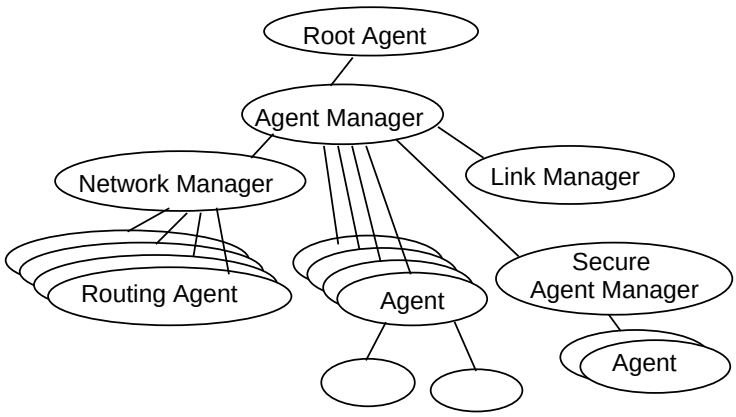

Figure 2. Agent Hierarchy

In our framework, an agent plays a role of the active packet or the user application. Occasionally, one single agent may play the both of them. For example, the enquiry agent can contain routing procedure and user interfaces for enquiry.

The framework has following characteristics: *agent replication* and *agent hierarchy*. These features make our framework quite different from other mobile agent systems.

3.1 Agent replication

To eliminate the effort to develop the inter-agent and the inter-host communication protocols, we only allow the communications among the agents which have same agent ID. [3] Agent replication enables the existence of the agents which have same ID on the different mobile hosts. [4] At the creation time of the agent, each agent has a unique ID. The agent can be replicated before the migration to the other host while keeping the same agent ID. The agent on the host has noticed about arrival of the replica agent, when the replica agent arrive the host. Then the agent on the host can perform the data synchronization with the arrived agent. The agents with same ID have same functionality and same authority. Faking agent ID can be easily avoided by shared secret key and code matching. So, agent replication can be regarded as a secure communication method.

[3] These agents are called "replica agents".

[4] Generally, other mobile agent systems use the unique global ID for each agent.

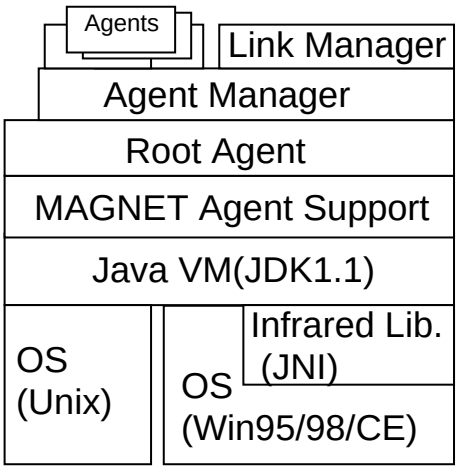

Figure 3. Configuration of MAGNET

For the network protocol agents, the agent ID for the protocol is identical in all mobile hosts. This works like a "port no" of TCP/IP implementation. But for TCP/IP, user must set up the host to understand the protocol. In out framework, unprepared hosts can obtain the protocol agent by the agent migration. So there is no need for set up of protocols.

3.2 Agent hierarchy

We introduce the hierarchy into agents to manage the mobile agents as software components. Each agent can contain other agents inside of itself. Child agent can be registered, replaced or disposed by parent agent. Agents can be replaced or disposed dynamically by its ascendant agents. When an agent moves to another host, its descendant agents move to the same host while keeping hierarchy. This feature enables the extensive system like the micro-kernel systems. Figure 2 shows a canonical configuration of agent hierarchy. `Root Agent` is the only agent which never moves. `Agent Manager` is a manager which deals agent registration and the communication between agents and other managers. Agent Manager contains sub managers such as Network Manager, Link Manager, and so on. It may also contain sub Agent Managers as a secure sand-box environment.

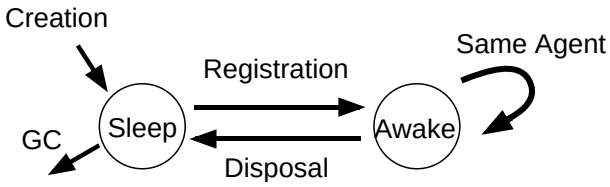

Figure 4. Agent Lifecycle

```
public class MagAgent implements Serializable{
  MagID id = MagID.getNewID();
  transient MagManager magMgr;
// called after the registration of Agent Manager
  void onRegister(){ }
// called before the disposal of the agent
  void onDispose(){  }
// called when replica agent arrived to the same host
  void onSame(MagAgent m){ }
}
```

Figure 5. MAGNET Agent

3.3 Adaptation to Network

As we mentioned previously, the dynamic extension and the adaptation to the dynamic network change are the core requirements of the mobile network systems. The introduction of mobile agents and agent hierarchy enable the dynamic extension of protocols and services of MAGNET. And, following agents enables the adaptation to the dynamic change of the network topology.

`Link Manager` plays a key role in the ad hoc network formation. Link Manager manages the discovery of the adjacent hosts and the communication on the link layer between these hosts and the host itself. Link Manager also manages a set of agents called Link Monitor which wants to be notified when the status of the links between neighbor hosts have changed. When new host appears or existing host disappears, Link Manager notify Link Monitors the event, and then each Monitor deals the event autonomously. `Network Manager` deals routing agents to manage the network layer. When Agent Manager calls Network Manager to send a agent to some destination, Network Manager requests the routing agent to deliver the agent to the destination. Each routing agent has a routing strategy which decide the travel path to the destination.

```
interface LinkMonitor {
// called when new neighbor host is appeared
  public void onNew(Vector ns);
// called when existing host is disappeared
  public void onBye(Vector ns);
}
```

Figure 6. Link Monitor

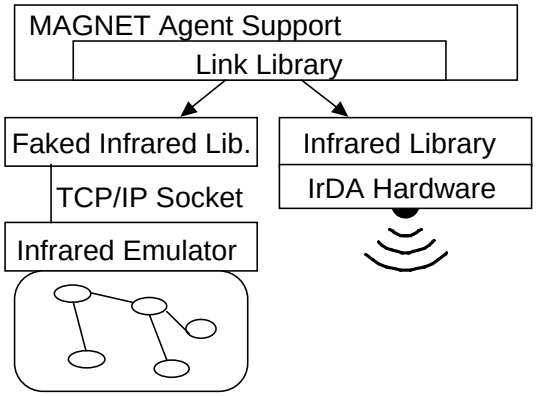

Figure 7. Infrared link libraries

4 Implementation

We developed a prototype network system named MAGNET(Mobile AGent NETwork) based on our framework. Currently, MAGNET is written in Java(JDK 1.1 and later). The system contains a wireless communication emulation environment and a "real" infrared using the java native interface. Figure 3 shows the configuration of MAGNET. Agent Support library contains convenient agent factories, link libraries, agent serializer/deserializer, class loader, etc.

4.1 MAGNET Agent

Agents in MAGNET are all subset of class `MagAgent`(Figure 5). We design `MagAgent` as simple as possible. So it has the minimum functionality for the mobile agent network. When the agent is registered to the Agent Manager, `onRegister()` is called. `onDispose()` is called when the agent is disposed by Agent Manager. `onSame()` is a key method of agent replication. When the replica agent arrives the host, this function is called with reference of the

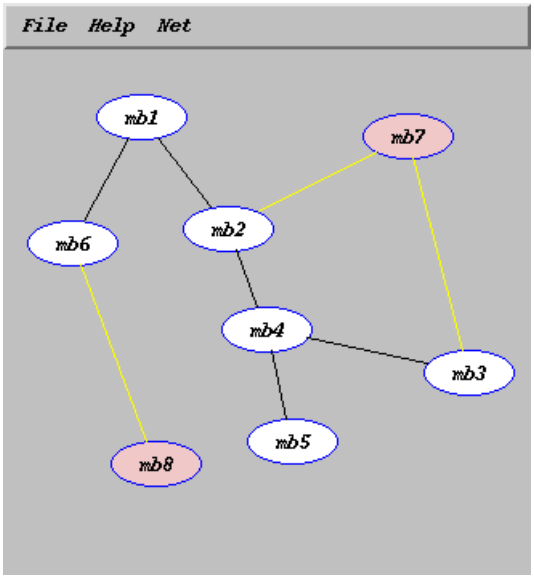

Figure 8. Wireless Network Emulator

replica agent. So, the agent can easily perform the information sharing or the communication with the new agent. Figure 4 shows the lifecycle of MAGNET agents. Variable `MagManager` keeps the reference of Agent Manager.

Figure 6 shows the interface for Link Monitoring agents. The agent which implements the interface are called by Link Manager when the status of the neighbor links are changed.

4.2 Infrared implementation

Figure 7 shows the link library of the MAGNET agent support libraries which implements the infrared link layer based on IrDA. One can dynamically change the binding between the faked library and the real library. The same agent can utilize the both libraries without any modifications. The faked library is used to connect with *the wireless network emulator* which is also implemented by Java. Several MAGNET hosts can connect to one emulator using TCP sockets.

Figure 8 shows the display of the wireless network emulator. On the emulator display, one can dynamically connect or disconnect the link between hosts by using the pointing device. Each oval node denotes the mobile host, and line denotes a link. Pale color of the node means the host is sending some agents.

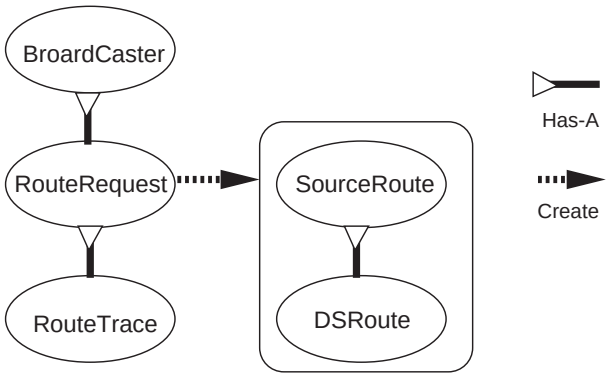

Figure 9. Agent structure for DSR protocol

By using the emulator, development and testing of the agents are significantly helped.

The real infrared library utilizes Java Native Method to activate IrDA functions offered by operating systems(Windows95/98/CE). The transfer speed of the infrared is upto 4Mbps by using IrDA FIR[12].

The link layer of MAGNET can also implemented on other medium or existent network protocols. Currently, we implement the link layer over the infrared, TCP, UDP, and serial lines.

4.3 Performance

We do not design the system for the high-performance networkings.

5 Example:DSR protocol

To show the expressive power of MAGNET, we implement DSR(Dynamic Source Routing[3]) on the MAGNET. DSR is based on two mechanisms: Route Discovery and Route Maintenance. Route Discovery is the mechanism by which a source host S wishing to send a packet to a destination host D obtains a source route to D. Route Maintenance is the mechanism by which a packet's sender S detects if the network topology has changed such that it can no longer use its route to D.

```
class BroadCaster extends MagAgent {
  Vector nodes =new Vector();  //for keeping visited nodes
  MagAgent mes;
  BroadCaster(MagAgent m){  mes = m;  }
  void onRegister(){
     Vector newNodes = magMgr.getNeighbor();
     newNodes.removeAll(nodes);
     nodes.addAll(newNodes);
     nodes.addElement(magMgr.currentNode());
     magMgr.seqRegister(mes);  //Register Message
     new DirectMulticast(this,newNodes,magMgr);
  }
  void onSame(MagAgent bc){  //replica Agent arrives!
     nodes.addAll(((BroadCaster)bc).nodes);
  }  // Just sharing Information
}
```

Figure 10. Broadcast Agent

5.1 Design of Agent Hierarchy

Using our framework, protocols are divided into several mobile agents. We first show the structure of agents for DSR(figure 9) . DSR protocol is composed by five kind of mobile agents. We briefly describe them as follows, `BroadCaster`:Broadcasting `RouteRequest` to all neighbors, `RouteRequest`: When arrive the destination, create `SourceRoute`. `RouteTrace`: Preserving routing path, `SourceRoute`: Bring back `DSRouter` using preserved route by `RouteTrace`. and `DSRouter`: Router agent for DSR protocol containing the route to the destination.

From the limitation of the space, we only explain the action of `BroadCaster` by example. Figure 10 shows the main part of the Java code for `BroadCaster` agent. In figure 11, there are 4 hosts connected each other. In the beginning, only host A contain a `BroadCaster` with a message agent `x`. First, `onRegister()` of `BroadCaster` is called. Then it checks new neighbor hosts using saved `nodes`[5], then calls `DirectMulticast` to copy itself to all new hosts(B,C in Figure 11).

Host B obtain a new `BroadCaster` agent from A. Then the agent recognize a new host D. So it send itself to D. Host D obtain a new `BroadCaster` agent from B. The agent cannot find new host. So `BroadCaster` agent just stay in D. Host C also obtain a new `BroadCaster` agent from A. Then the agent recognize a new host D. So it send itself to D. Host D obtain same `BroadCaster` agent from C. Agent manager in host D may find that same agent is already there,

[5] At this point, variable `nodes` is empty.

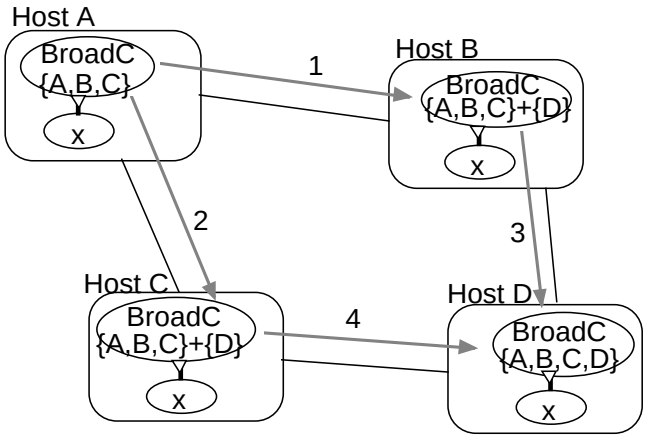

Figure 11. Example of Broadcasting

so call `onSame` with the arrived agent reference. `BroadCaster` just merge the `nodes` of new agent. This example shows there will be no loop in `BroadCaster` agent. Finally, agents can be disposed by employing TTL(Time To Live) into each agents.

5.2 Extension for Ad Hoc Situation

At this point, previous DSR implementation is not suitable for ad hoc, dynamic situation. Because `BroadCaster` does not deliver the message to the new host after the first multicasting. We also implement the ad hoc extension to the protocol. By adding `onNew` method and registering as a Link Monitor, `BroadCaster` agent can be easily extended to adapt the ad hoc situations. In favor of agent hierarchy(component module), modification on the agent is easy and little. Figure 12 shows the extended version of `BroadCaster` for adapting ad hoc situation.

Table 1 shows the number orginal and modified lines of on agent source codes for adhoc extension and TTL(Time To Live) extension ("-" means no need for extension). To implement the DSR protocol, only 141 lines of Java code are required. Modifications are very easy and only small part of the code (12 lines in `BroadCaster`)is required to change. Finally, the DSR protocol agents which has the ad hoc and the TTL feature can be written in only 226 lines. This result exemplifies the productivity and the extensibility of MAGNET.

```
//Broadcast agent adapting
// to the ad hoc situation
class BroadCasterAdhoc extends MagAgent
 implements LinkMonitor{       //Implement LinkMonitor
  Vector nodes =new Vector();  //for keeping visited nodes
  MagAgent mes;                // Message Agent
  BroadCasterAdhoc(MagAgent m){  mes = m; }
  void onRegister(){
     Vector newNodes = magMgr.getNeighbor();
     newNodes.removeAll(nodes);
     nodes.addAll(newNodes);
     nodes.addElement(magMgr.currentNode());
     magMgr.seqRegister(mes);          //Register Message
     magMgr.registerLinkMonitor(this); // Register as LinkMonitor
     new DirectMulticast(this,newNodes,magMgr);
  }
  public void onNew(Vector newNodes){ // New mobile host
     newNodes.removeAll(nodes);
     nodes.addAll(newNodes);
     new DirectMulticast(this,newNodes,magMgr);
  }
  void onSame(MagAgent bc){ //replica Agent arrives!
     nodes.addAll(((BroadCasterAdhoc)bc).nodes);
  }  // Just sharing Information
}
```

Figure 12. Ad hoc Broadcast Agent

6 Related Work

For supporting host mobility in the network, There are many studies about routing issues. DSDV[1] is a distance vector ad hoc routing protocol. This protocol avoids loop using sequence number from each host. Our system can also avoid loop by keeping information in the agent and using agent replication. DSR[3] is a dynamic source routing protocol described in last section. AODV[2] can be regarded as a combination of DSR and DSDV. These protocols satisfy the adaptation to the dynamic network change in the ad hoc networks, but do not satisfy the dynamic extension requirement.

PLAN[4] is a functional language for active networks. They divide the services and the packets, and provide separate way of extensions. This language did not consider about dynamic change of the network. SmartPacket[6] is a active network project which is using special language Sproket. They also did not consider the ad hoc situation nor the mobile environments. ANTS[5] is a Java based active network system. This seems very close to our system. But this system does not pay attention to the inter-packet(agent) communication.

Table 1. Modified Lines for Agents

Agent Name	Org. code leng(lines)	Adhoc mod.	TTL mod.
`BroadCaster`	53	12	12
`RouteRequest`	29	5	12
`RouteTrace`	16	-	-
`SourceRoute`	22	17	12
`DSRoute`	21	3	12

Figure 13. Meeting supported by MAGNET

So, they need some protocols between Nodes and Capsules. Our system does not need any inter-agent protocols in favor of agent replication. The on demand code distribution can be introduced into our system. It is also not designed for the mobile ad hoc environment.

7 Conclusion

This paper proposes a mobile agent framework which can form the mobile ad hoc networks and exemplify that the framework satisfies the requirements for the mobile networks by developing a prototype implementation MAGNET. MAGNET satisfies the both requirements of the dynamic extension and the adaptation to the dynamic network. Agent replication is one of the key features of our system to simplify the inter-agent communications. Link monitoring agents enable the adaptation to the environment.

Using our prototype system MAGNET, one can easily construct an ad hoc network, and sharing information on the network. Our prototype implementation MAGNET can be regarded as a noble combination of Mobile Agent and Active Network with ad hoc extension. Implementation of DSR protocol shows that MAGNET can be used for non-trivial network programs as well as mobile application programs. Required extensions for adapting ad hoc situation were very simple and intuitive in favor of agent hierarchy and `Link Monitor` feature.

We believe MAGNET is a first active network system for mobile computers including PDAs using the infrared communication device. For the PDAs, there are strong requirement of on demand installation of various applications such as news reading, guidance system, and personal information exchange. We are currently implementing meeting support agents for ad hoc on-demand meeting(Figure 13). Some prototype agents are working such as sharing text files or spread sheets. A lot of future works are remained such as more concrete security, on-demand code distribution, dynamic code update, and performance issues.

References

[1] Charles E.Perkins and Pravin Bhagwat, Highly Dynamic Destination-Sequenced Distance-Vector Routing(DSDV) for Mobile Computers,*Proc. of the SIGCOMM'94* (1994) pp.234–244.

[2] Charles E.Perkins, Ad Hoc On Demand Distance Vector (AODV) Routing , Internet Draft,draft-ietf-manet-aodv-01.txt,(1998). work in progress.

[3] David B. Johnson and David A. Maltz, Dynamic Source Routing in Ad Hoc Wireless Networks, *In Mobile Computing*, Tomasz Imielinski and Hank Korth eds., Chapter 5, Kluwer Academic Publishers (1996) pp.153–181.

[4] Michael Hicks, Pankaj Kakkar, Jonnathan T. Moore, Carl A. Gunter, and Scott Nettles, Network programming with PLAN,*In Proceedings of the IEEE Workshop on Internet Programming Languages*(1998).

[5] David J. Wetherall, John V. Guttag and David L. Tennenhouse, ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols, IEEE OPENARCH'98(1998).

[6] Beverly Schwartz, Wenyi Zhou, Alden W. Jackson, W. Timothy Strayer, Dennis Rockwell, Craig Partridge , Smart Packets for Active Networks, BBN Technologies (1998).

[7] M. Yamashita and T. Kameda, Computing on Anonymous Networks Part I, Characterizing the Solvable Cases, IEEE Trans. Parallel and Distributed Systems 7, No. 1 (1996) pp.69–89.

[8] Danny B. Lange, Mitsuru Ohshima, Programming and Deploying Java Mobile Agents with Aglets, Addison Wesley(1998).

[9] General Magic, Odyssey(1998).
(http://www.genmagic.com/technology/odyssey.html)
[10] ObjectSpace, Voyager
(http://www.objectspace.com/Voyager)
[11] Hosh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva, A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols, Mobicom98(1998).
[12] Infrared Data Association : Serial Infrared Link Access Protocol, Version1.1 (1996). (http://www.irda.org)