Wireless Agents in Ad Hoc Networks

Stephen Quirolgico[1], L. Jay Wantz[1], Michael Miller[1], Naveen Srinivasan[2], Vlad Korolev[1], and Michael Fay[1]

[1] Applied Research Group
Aether Systems, Inc.
Owings Mills, MD 21117
{squirolgico, jwantz, mmiller, vkorolev, mofay}@aethersystems.com
[2] Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County
Baltimore, MD 21205
nsrini1@cs.umbc.edu

Abstract. With the current trend toward ubiquitous computing comes wireless devices capable of direct, peer-to-peer communication. Such devices will be capable of forming the nodes of ad hoc networks. In the near future, it is expected that ad hoc networks will be saturated with heterogeneous hardware and software requiring mechanisms to facilitate interoperability. At the application layer, such interoperability may be facilitated using software agents. Unfortunately, it is currently impractical to design, implement, and study device-based software agents within ad hoc networks as the required hardware and software configurations do not yet exist. In this paper, we present a software framework for developing and studying device-based software agents in the context of ad hoc networks. We refer to this framework as the Wireless Agent Simulator.

1 Introduction

Using current wireless technologies, devices including laptops, PDAs and mobile phones can be equipped to communicate with each other directly and form the nodes of *ad hoc networks*. Ad hoc networks are local-area networks that are dynamically created through the connections between two or more nodes. Unlike other networks, ad hoc networks are formed and maintained in a decentralized fashion; that is, each node in the network is responsible for handling network management functions including establishing connections and maintaining routing protocols without a centralized controller. In addition, the topology of an ad hoc network continually changes based on the physical location of its nodes. Ad hoc networks may be used for a number of applications including file exchange and games, and may be especially useful in disaster relief situations or military theater operations where spontaneous network formations are necessary. In the near future, the continued proliferation of Bluetooth-, IrDA- and 802.11-equipped wireless devices is expected to result in ad hoc networks saturated with heterogeneous systems (i.e., heterogeneous devices, applications and configurations) requiring technology that can support interoperability between these systems.

In an ad hoc network, interoperability between wireless devices is required at both the network and application layers. Regarding the network layer, wireless technologies including IrDA and Bluetooth may be used to establish the low-level interoperability between devices. At the application layer, protocols including FTP, HTTP, and WAP may be used to facilitate interoperability between homogeneous applications. However, no such protocols exist to facilitate interoperability between heterogeneous wireless applications and services. This is problematic in an ad hoc network where such applications may need to interact. One way to facilitate interoperability between both homogeneous and heterogeneous wireless applications and services is to use software agents. We refer to such agents as *wireless agents*.

We envisage wireless agents playing an increasing role in facilitating interoperability at the application-layer in ad hoc network environments. Unfortunately, it is difficult to study wireless agents in the context of ad hoc networks as most OEM vendors do not yet fully support the required hardware and software configurations [8]. Even if such devices and applications were fully supported, it may be impractical to analyze the behavior of wireless agents between possibly numerous heterogeneous hardware and software configurations. To adequately design, implement, and test wireless agents, it is useful to have a framework to simulate the execution of such agents in an ad hoc network environment. In this paper, we present the *Wireless Agent Simulator* (WAS) system for simulating the execution of wireless agents. WAS facilitates the development of wireless agents by providing a framework within which developers may design, implement, and test wireless agents (e.g., their behaviors, services, GUI designs, and interaction with other agents) within virtual ad hoc network environments.

2 Wireless Agents

We define a wireless agent as a software agent that resides on a wireless device and directly communicates with agents on other devices. A wireless agent may be an application or service, or may provide a wrapper around, or interface to, an existing wireless application. We distinguish between wireless agents and *mobile agents* by noting that the latter migrate between hosts [10]. A wireless agent may be a mobile agent if it migrates between wireless devices, but this property is neither a requirement nor a focus of this work.

Like other types of agents, wireless agents embody a number of general properties including autonomy (i.e., control over their own actions and internal state), reactivity (i.e., responsiveness to the environment) and pro-activity (i.e., initiation of behavior to achieve their goals) [19]. Thus, wireless agents may function on their own with little or no user interaction and may also act as both a client and server simultaneously to facilitate peer-to-peer communication. Wireless agents exhibit additional properties to (1) facilitate interoperability at the application layer, (2) function on small-footprint wireless devices, and (3) operate in an ad hoc network.

2.1 Application-Layer Interoperability

Because the primary goal of wireless agents is to facilitate interoperability between heterogeneous applications and services, such agents may be required to conduct some form of reasoning to infer the beliefs, desires, and intentions of other agents. In a multiagent system where agents influence the behavior of one another through direct communication, an agent may use an agent communication language (ACL) to help communicate such mental states. One ACL that has recently gained attention within the agent research community is being specified by the *Foundation for Intelligent Physical Agents* (FIPA). This ACL, known as *FIPA ACL*, is a high-level, message-oriented communication language and protocol for supporting run-time knowledge sharing among agents. Like other ACLs, FIPA ACL is distinguished by its objects of discourse and semantic complexity [11].

Agents use FIPA ACL by passing FIPA messages that contain a *communicative act* [9]. A communicative act is a language primitive that identifies the message type (e.g., assertion, query, or command). Each communicative act has well-defined semantics that may be used by a receiving agent to help infer the intention of the sender. A FIPA message also contains a set of message elements that may be used by a receiving agent to further refine the context surrounding a received message. There are a number of reserved FIPA message elements including `content` (i.e., the content of the message), `ontology` (i.e., the ontology that describes the content), and `language` (i.e., the language used to encode the content).

Even using a standardized communication language, a wireless agent may be unable to exploit the knowledge encapsulated by a received FIPA ACL message. This is because FIPA ACL does not mandate specific reasoning methodologies to be used by agents. Ultimately, application-layer interoperability will entail additional mechanisms that must be defined by the agent developer. Such mechanisms may involve the use of representation schemes to structure knowledge as well as the use of ontologies to provide adequate semantics for reasoning about knowledge. An agent will also need to employ a suitable reasoning mechanism. Although knowledge representation, ontologies and reasoning mechanisms are beyond the scope of this paper, we briefly discuss how these may be used to facilitate application-layer interoperability in Section 3.3.

2.2 Platform Considerations

Although this paper focuses on wireless agents in a simulated environment, platform considerations must be taken into account in order to yield a valid simulation of the system as well as to assess potential portability of simulated wireless agents to actual devices.

One issue related to platform portability concerns the agent-based platform to be used for the simulation. Here, the goal is to identify a platform that supports the porting of simulated wireless agents to actual devices. Since virtually all FIPA-based platforms are implemented in Java, a platform is needed that allows agent-based components written in J2SE/EE to port to small-footprint devices that support KVM, PersonalJava, and J2ME [12]. In addition, a suitable layer of abstraction between simulated wireless agents and lower-level components is required to further support portability to actual devices.

Unlike *infrastructured* networks that utilize fixed and wired gateways, ad hoc networks are *infrastructureless*; that is, they are formed by mobile devices that can be connected dynamically in an arbitrary manner [16]. In an ad hoc network, the devices on which wireless agents reside must support a wireless, device-to-device communication technology such as IrDA or Bluetooth rather than technologies associated with infrastructured networks like wireless WANs (e.g., GSM, CDMA, and GPRS) and wireless LANs (e.g., HomeRF). In a simulated ad hoc network environment, care must be taken to ensure that the simulated behavior of wireless technologies is valid.

2.3 Ad Hoc Network Considerations

In addition to interoperability and platform issues, any framework for the development of wireless agents must take into consideration issues related to the nature of ad hoc networks. Such issues are typically concerned with the mobility of nodes within the network.[3] Because nodes are always moving in an ad hoc network, the topology of the network is continually changing.

One issue related to the movement of nodes within an ad hoc network concerns the maintenance of sessions between wireless agents. When two wireless agents reside on devices that move in and out of *connectivity* (i.e., the ability to communicate directly) with one another, mechanisms are needed in order to both quickly establish dialog as well as gracefully terminate interaction. In addition, wireless agents may need to be able to re-establish sessions in an appropriate manner if a connection is re-established.

A related issue to session maintenance is system performance. Because nodes may move quickly and unpredictably in an ad hoc environment, performance of wireless agents is crucial in order to complete interactions before the devices on which they reside move out of proximity. Often, performance will be restricted by a number of factors associated with wireless and mobile computing in general including platform constraints and limited bandwidth[18]. Thus, care should be taken to ensure that wireless agents utilize efficient mechanisms to carry out their functionality.

Another issue of ad hoc networks is how to route messages through a set of intermediate devices. Recently, a number of research efforts have proposed

[3] Here, the term 'mobility' refers to the movement of wireless devices in physical space and *not* the migration of agents between hosts (as is the case when we talk about mobile agents).

the use of *ad hoc network routing protocols*. Ad hoc network routing protocols are concerned with discovering and establishing multi-hop paths through the network to other nodes. Such protocols may allow communication between two wireless agents using intermediate agents as routers.

Though ad hoc networks require consideration of network security issues [20], the topic of security in ad hoc networks is beyond the scope of this paper.

3 Wireless Agent Simulator

The Wireless Agent Simulator (WAS) is a framework for building and simulating wireless agents in virtual ad hoc network environments. WAS is implemented on top of the FIPA-based *Java Agent DEvelopment Framework* (JADE) platform [4]. The motivation for the use of JADE over other FIPA-based platforms stems primarily from its potential for portability to wireless devices [1].

The WAS architecture is comprised of a *Simulator for Wireless Ad hoc Networks* (SWAN) and a set of agent-enabled, virtual devices. Figure 1 shows the general WAS architecture with SWAN and two agent-enabled virtual devices.

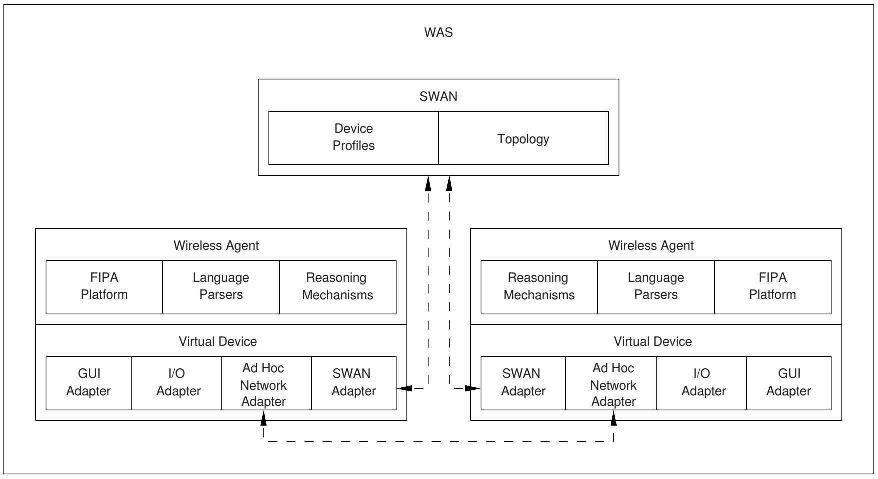

Fig. 1. General WAS architecture with SWAN and two agent-enabled virtual devices.

3.1 Simulator for Wireless Ad hoc Networks (SWAN)

In an ad hoc network, the ability of a wireless agent to initiate and maintain a conversation with another wireless agent is dependent upon a number of factors including the wireless hardware, protocols, and physical proximity of the devices on which the agents reside. In WAS, SWAN simulates these characteristics of an ad hoc network. SWAN is comprised of a *device profiles* component and a *topology* component.

The device profiles component receives and maintains information on wireless technologies and profiles used by virtual devices within the simulated ad hoc network. It also provides a set of services to virtual devices including notification services for alerting virtual devices of other devices with which they may communicate. Information about a virtual device is sent to the device profiles component when a virtual device is initialized. During initialization, a virtual device subscribes to the services of the device profiles component by sending it a device ID as well as information about its wireless technology.

The topology component maintains information about the topology of a virtual ad hoc network. The topology component is used to (1) initialize and modify the geospatial locations of virtual devices, (2) manipulate the physical orientation of virtual devices, and (3) graphically display the virtual ad hoc network. When a virtual device registers with SWAN, the topology component generates an initial geospatial location for the device and displays the device's location graphically within a GUI that represents a bounded virtual geospatial region. Once a virtual device is registered, the GUI may be used to alter the location of the device by dragging an icon of the device within the bounded region.

In addition to simulating movement and physical orientation, SWAN assesses the potential for connectivity between virtual devices. Connectivity is dependent upon factors including: (1) the wireless technologies used by the devices, (2) the physical orientation of the devices, and (3) the physical proximity between the devices. For example, connectivity between two IrDA-enabled virtual devices may occur when both are in physical proximity and each is within line-of-sight of one another. In a simulated Bluetooth environment, however, establishing connectivity is more complex. Here, connectivity between two Bluetooth-enabled virtual devices is determined by factors including (1) maximum RF range allowed by the simulated Bluetooth receivers, (2) the number of virtual devices in the associated piconet, and (3) whether the simulated Bluetooth technology supports point-to-point or point-to-multipoint communication.

3.2 Virtual Devices

In WAS, wireless agents are deployed on virtual devices. Like device emulators that provide PC-based versions of a device for testing application-layer software, WAS virtual devices provide an environment for testing wireless agents prior to deploying on (JADE-enabled) devices. However, WAS virtual devices differ from device emulators in that the former implements an architecture that is not specific to any particular device manufacturer. A WAS virtual device is further distinguished from current device emulators as the latter do not typically emulate wireless transmissions nor embody the notion of movement in physical space. A WAS virtual device is comprised of a number of components including a SWAN adapter, ad hoc network adapter, GUI adapter, and I/O adapter.

A virtual device may join a simulated ad hoc network using a *SWAN adapter*. A SWAN adapter is used by a virtual device to subscribe to SWAN notification services and to receive updates on other virtual devices with which connections may be established. When a virtual device receives notification that a connection may be established with another device, it makes this information available to its *ad hoc network adapter*.

An ad hoc network adapter is an interchangeable component comprised of a set of APIs, protocol stacks, and hardware simulators that mimics the properties of a specific infrastructureless, wireless technology like Bluetooth, IrDA, or 802.11. Although an ad hoc network adapter may reflect a vendor-specific implementation of a wireless technology, custom adapters may be used to simplify or enhance the simulation environment. Virtual devices may use one of possibly several different ad hoc network adapters and thus may simulate a variety of wireless hardware and software configurations. To ease portability from the simulated environment to actual devices, however, virtual devices should use standardized APIs and protocol stacks whenever possible.

When a virtual device is initialized, it is configured to use a specific ad hoc network adapter. During subscription to SWAN notification services, a virtual device sends information to SWAN about its ad hoc network adapter. This information may include the class of wireless technology (e.g. Bluetooth, IrDA, or 802.11), the vendor of the implementation, and properties of the simulated hardware components. When a virtual device receives notification that a connection may be established with another device, it makes this information available to its ad hoc network adapter which, in turn, uses this information to provide a connection for wireless agents at the application-layer. Using protocols embodied by an ad hoc network adapter, a communication channel between two virtual devices may be established. Note that in addition to providing a communication channel between virtual devices, an ad hoc network adapter may also provide additional services. For example, an ad hoc network adapter for Bluetooth may also provide service discovery through the Bluetooth Service Discovery Protocol (SDP) or the Salutation service discovery and session management protocol.

In order to provide a simulation environment that resembles a typical handheld device, WAS virtual devices include GUI and I/O adapters. A GUI adapter provides access to the GUI of the virtual device. This access is provided by a set of APIs. By using the GUI component of a virtual device, developers may design, implement and test the behavior of GUIs for their wireless agent. An I/O adapter allows for the reading and writing of files by a wireless agent. Like an ad hoc network adapter, both GUI and I/O adapters are components that may be interchanged with other adapters to reflect different GUI and I/O environments.

3.3 Wireless Agent

In WAS, a wireless agent is a user-defined agent that provides peer-to-peer functionality between it and other wireless agents. A wireless agent resides on top of a WAS virtual device and utilizes APIs from that device in order to access GUI, I/O and wireless technology components. Although WAS provides a framework for building wireless agents, it does not mandate the use of any particular application-layer functionality. Some particularly important applications of wireless agents may include service discovery and ad hoc routing.

Using FIPA ACL, wireless agents may communicate in a standardized fashion, thereby facilitating interoperability. However, since neither WAS nor FIPA ACL dictate a specific reasoning methodology, agent developers may need to extend their agents in order to support the level of interoperability that they require. Although various methods may be used to support enhanced interoperability, we have implemented some proof-of-concept wireless agents that utilize standardized knowledge representation schemes, ontologies and reasoning mechanisms.

One emerging standard for representing knowledge is the Resource Description Framework Schema (RDFS). RDFS is an XML-based language that provides a means to define vocabulary, structure and constraints for expressing metadata about Web resources. Although RDFS is sufficient for structuring knowledge, it does not embody the formal semantics and expressivity required to support ontological modeling and reasoning. However, an extension to RDFS to allow for the proper representation of ontologies has been specified in the Ontology Inference Layer (OIL) [7]. By using FIPA ACL in conjunction with OIL, wireless agents may exchange semantically richer knowledge and improve the potential for heterogeneous interoperability. Inferencing on such knowledge requires an appropriate parser and reasoning mechanism (e.g., Prolog or the Java Expert System Shell (JESS)).

4 Related Work

Recently, a number of research efforts have looked at using agents in ad hoc networks. Most of these efforts have been focused on using mobile agents to facilitate ad hoc routing protocols [14, 2, 13, 17]. Some efforts have also used agents for service discovery in ad hoc networks [3]. The work described in this paper is distinguished from these other efforts by its use of agents that utilize FIPA ACL to promote interoperability between agents in ad hoc network environments.

A few research efforts are also using FIPA ACL in wireless environments. Two specific efforts include the Lightweight Extensible Agent Platform (LEAP) project [5] and the CRUMPET project [12]. The former is concerned with porting JADE to wireless devices while the latter involves porting the FIPA-OS framework [15] to wireless devices. Although these projects entail the use of FIPA-based agents on wireless devices, they are currently designed for use in wireless, infrastructured networks with fixed gateways. The work described in this paper is distinguished from these efforts through its focus on FIPA-based agents in wireless, infrastructureless networks (i.e., ad hoc networks).

There are also a number of research efforts looking into the development of simulators for ad hoc networks. One such effort concerns an extension to the *ns* discrete event simulator for providing multi-hop ad hoc network simulation [6]. Although such simulators provide very detailed implementations of low-level networking components, they do not target issues at the application-layer. The work presented in this paper is distinguished from these other efforts primarily by its focus on interoperability between wireless agents at the application layer.

5 Conclusions and Future Work

In this paper, we presented WAS, a framework for developing wireless agents in a simulated ad hoc network environment. The motivations for developing WAS stems from a lack of suitable wireless hardware and software configurations for deploying wireless agents, and the lack of a framework for studying the behavior between wireless agents on possibly numerous hardware and software configurations.

Currently, the implementation of WAS is undergoing refinement and the addition of new features. Refinements to WAS will include the integration of more standardized APIs for ad hoc network adapters to further facilitate portability to wireless devices. New features of WAS may include support for GPS and mapping functionality to facilitate the development of wireless agents for location-based services. Refinements to SWAN will include support for additional ad hoc network characteristics. In addition, we intend to verify the portability of wireless agents to appropriate wireless devices as such devices become available.

We believe WAS will provide an important framework for studying software agents within the context of ad hoc networks. Some areas of research that may be furthered using WAS include negotiation of ad hoc routing protocols among FIPA-based wireless agents, modeling ad hoc services, wireless agents for m-commerce, and interoperability assessment between heterogeneous wireless agents.

References

1. Adorni, G., Bergenti, F., Poggi, A., Rimassa, G.: Enabling FIPA Agents on Small Devices. Fifth International Workshop on Cooperative Information Agents. Modena, Italy, 2001
2. Bandyopadhyay, S., Paul, K.: Evaluating the Performance of Mobile Agent-Based Message Communication among Mobile Hosts in Large Ad Hoc Wireless Network. Proceedings of the 2nd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems. Seattle, WA, 1999
3. Barbeau, M.: Service Discovery Protocols for Ad Hoc Networking. CASCON 2000 Workshop on Ad Hoc Communications. Toronto, Canada, 2000
4. Bellifemine, F., Poggi, A., Rimassa, G.: Developing Multi-Agent Systems with a FIPA-compliant Agent Framework. Software - Practice and Experience, 13, 2001
5. Bergenti, F., Poggi, A.: A FIPA Platform for Handheld and Mobile Devices. Eighth International Workshop on Agent Theories, Architectures, and Languages. Seattle, Washington, 2001
6. Broch, J., Maltz, D.A., Johnson, D.B., Hu, Y., Jetcheva, J.: A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. Fourth ACM/IEEE International Conference on Mobile Computing and Networking. Dallas, TX, 1998

7. Broekstra, J., Klein, M., Decker, S., Fensel, D., van Harmelen, F., Horrocks, I.: Enabling Knowledge Representation on the Web by Extending RDF Schema. Proceedings of the 10th World Wide Web conference. Hong Kong, China, 2001
8. Finin, T., Joshi, A., Kagal, L., Ratsimore, O., Korolev, V., Chen, H.: Information Agents for Mobile and Embedded Devices. First International Workshop on Cooperative Information Agents. Modena, Italy, 2001
9. Foundation for Intelligent Physical Agents: FIPA Communicative Act Library Specification, 2000
10. Gray, R.S., Cybenko, G., Kotz, D., Rus, D.: Mobile agents: Motivations and State of the Art. In: Bradshaw, J. (ed), Handbook of Agent Technology. AAAI/MIT Press, 2001
11. Labrou, Y., Finin, T., Peng, Y.: Agent Communication Languages: The Current Landscape. IEEE Intelligent Systems, 14(2), 1999
12. Laukkanen, M., Tarkoma, S., Leinonen, J.: FIPA-OS Agent Platform for Small-footprint Devices. Eigth International Workshop on Agent Theories, Architectures, and Languages. Seattle, Washington 2001
13. Li, Q., Rus, D.: Sending Messages to Mobile Users in Disconnected Ad-Hoc Wireless Networks. Proceedings of the Sixth International Conference on Mobile Computing and Networking. Boston, MA, 2000
14. Minar, N., Hultman Kramer, K., Maes, P.: Cooperating Mobile Agents for Dynamic Network Routing. In: Hayzelden, A. (ed), Software Agents for Future Communications Systems. Springer-Verlag, 1999
15. Poslad, S., Buckle, P, Hadingham, R.: The FIPA-OS Agent Platform: Open Source for Open Standards. Fifth Internation Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents. Manchester, UK, 2000
16. Royer, E., Toh, C-K.: A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks. IEEE Personal Communications, 1999
17. Rus, D., Gray, R.S., Kotz, D.: Transportable Information Agents. Journal of Intelligent Information Systems, 22(3), 1997
18. Satyanarayanan, M.: Fundamental Challenges in Mobile Computing. Fifteenth Symposium on Principles of Distributed Computing. Philadelphia, PA, 1996
19. Woolridge, M., Jennings, N.: Intelligent Agents: Theory and Practice. Knowledge Engineering Review, 10(2), 1995
20. Zhou, L, Haas, Z.J.: Securing Ad Hoc Networks. IEEE Network, 13(6), 1999