

A NOVEL SOFTWARE AGENT FRAMEWORK WITH EMBEDDED POLICY CONTROL¹

Cho-Yu Jason Chiang, Ritu Chadha, Yuu-Heng Cheng, Gary Levin, Shihwei Li, Alex Poylisher
Telcordia Technologies
One Telcordia Drive, Piscataway, NJ 08854, USA

Abstract

In general, agent frameworks allow the functionality of their agents to be augmented with plug-ins, and agents in a multi-agent system collaborate with other agents to achieve their collective and/or individual goals. This model poses a few issues in terms of function coordination: (i) to allow the insertions of plug-ins on-the-fly, an agent framework needs to properly schedule the execution of plug-ins in a timely, coordinated fashion; (ii) plug-ins may conflict with each other in their behavior; and (iii) agents may need to adapt their behavior based on their environments, in addition to coordinating their actions according to their objectives.

Recent advances in policy-based management research suggest a promising direction to remedy the deficiency of existing software agent frameworks. We envision that the coordination between the plug-ins within an agent as well as the compliance of agents with environmental requirements can be achieved via policy control. With policy control, agents can adapt their behavior dynamically to meet the ever-changing environmental requirements such as those posed by military wireless ad hoc networks.

In this paper, we present a novel software agent framework¹ with seamlessly integrated policy control. We first describe the concept of policy control and how it is integrated into this framework. Next we explain the process of policy in action, discuss in detail the functions and benefits of policy control, and elaborate on the agent behavior adaptation that enables agents to achieve their goals while meeting dynamic environmental requirements.

1 Introduction

A software agent is a piece of cognitive or reactive, teleonomic or reflex, computer software [10]. An agent framework defines the programming standard for its agents, the internal component organization of its agents,

as well as the mechanism(s) enabling inter-agent communications. The programming standard of an agent framework describes the interfaces that a piece of software needs to implement in order to become an agent under that framework. The internal component organization specifies the components comprising an agent, the relationships among the components, the allowable function augmentation patterns, and the mechanisms that these components use to communicate with each other. To ensure that the communications syntax and semantics are consistent and well-understood among agents, an agent framework typically specifies an agent communication language and provides communication modules to be used for inter-agent message exchange. A group of communicative software agents may form a multi-agent system [10], and an agent framework typically provides mechanisms to organize agents in such systems autonomously.

Many agent frameworks have been reported in the literature. The most prominent ones include Jade [13], Grasshopper [14], Aglets [15], and Cougaar [12], to name a few. Their common goal is to enable agents to function autonomously in a distributed environment. These agent frameworks endow their agents with traits including autonomy, extensibility, self-organization, and unified communication mechanisms. In the military context, Cougaar, which was initially funded by DARPA, has been used for building a logistics application prototype. It has also been explored for use in other military projects [11]. According to the report in [11], the main differentiator that distinguishes Cougaar from other agent platforms is scalability. A logistics application built with Cougaar was demonstrated to scale up to 1000 nodes under the DARPA UltraLog program [16].

Although Cougaar has shown success in terms of scalability, this result was based on running a specific application in a presumably Internet-like, relatively stable testing environment. In contrast, the wireless ad hoc network environment is much more dynamic and unstable than the Internet environment due to its lack of fixed infrastructure, volatile radio connectivity, and node mobility. Because such networks do not require any pre-installed, fixed infrastructure to enable networking services, they are suitable for supporting military

¹ The research reported in this document/presentation was performed in connection with contract number DAAD19-01-C-0062 with the U.S. Army Research Laboratory. The views and conclusions contained in this document are those of the authors and should not be interpreted as presenting the official policies or position, either expressed or implied, of the U.S. Army Research Laboratory, or the U.S. Government unless so designated by other authorized documents. Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

operations. As a result, they have been designated as the foundation for the future US Army FCS [6] and WIN-T [7] networks.

The widespread use of wireless ad hoc networks in the military context has posed a lot of unprecedented challenges. Unreliable network connectivity and insufficient bandwidth cause some well-known Internet protocols such as UDDI and HTTP to be prohibitive, due to their dependency on TCP as well as excessive bandwidth requirement. Besides, wireless ad hoc networks represent a wide range of communications environments, which, as combined with the situation awareness requirements from the military, mandate agents to consider the requirements from their surrounding physical environment when they try to achieve their goals.

Recent advances in policy-based management research suggest a promising direction to remedy the deficiency of the existing software agent frameworks. We envision that the coordination between plug-ins within agents as well as the compliance of agents with their environments can be provided via policy control. With policy control, agents can dynamically adapt their behaviors to meet the varying requirements from the ever-changing operating environments, such as wireless ad hoc networks. Agents that operate in such environments therefore must provide better control and coordination among their plug-ins as well as between themselves to achieve their goals.

We have designed and implemented a novel software agent framework integrated with policy control. Our aim is to address the following issues in function coordination: (i) to allow the insertions of plug-ins on-the-fly, an agent framework needs to properly schedule the execution of plug-ins in a timely, coordinated fashion; (ii) plug-ins may conflict with each other in their behavior; and (iii) agents may have to observe the requirements from their environments, besides coordinating their actions according to their objectives.

The rest of this paper is organized as follows. In the next section we introduce the concept of policy control. We then describe the architecture of an agent in this framework and introduce the components of this framework. We will also describe how policy control is integrated into this framework, explain how a policy is enforced, and discuss in detail the benefits of policy control as compared to other well-known agent frameworks. In the sections on applications and benefits, we share our experience of building a distributed network management system with this agent framework, and elaborate on the agent behavior adaptation that enables agents to achieve their goals while contending with dynamic network environments.

2 Policy Control

Policies can be regarded as goals that are to be observed by the entities to which the policies are applied. They can also be regarded as rules, which take the following canonical form, “*if condition holds true, then action is performed.*” In a nutshell, policy control is to perform one or more control functions in accordance with a given set of rules. Systems that can be controlled by policies generally adopt modular designs so that the control logic can be extracted out of the individual modules comprising the systems. Depending on the policies that are enforced in a system, the behavior of the system could vary drastically. Systems that exercise their control over their components by means of policies are known as *policy-based systems*.

Under the following circumstances, using policies to control the behavior of a system is particularly advantageous:

- When the system is to be used in many different environments and their characteristics are either not well-known, or hard to specify precisely at system design time;
- When the system is expected to exhibit different behaviors when it is used by different users and/or under different situations.
- When the development and deployment of the system is assumed to be incremental; as new modules are inserted into a system, the control of the system needs to be adjusted without changing any existing module.

Policies take effect only when they are enforced, which means that their actions are executed as their conditions evaluate to true. Although conceptually the simple “if condition, then action” model works just fine, in practice a policy-based system needs to determine when the rules need to be examined. To make the enforcement of policies scalable to the number of policies in the system, the definition of a policy additionally includes a triggering *event* component. The occurrence of an event indicates that the policies using this event as their trigger needs to be checked to determine whether they should be enforced.

Policy control has been adopted as a requirement under both the FCS and the WIN-T programs of the U.S. Army. Battlefield situations can neither be exactly predicted nor exhaustively enumerated. Systems need to provide a certain degree of freedom for the administrators to adjust the behavior of the systems according to the situational demands. Besides, combat units are assembled based on the battlefield situations. Systems may need to behave differently when different devices are thrown into the mix per the decisions of the commander. As a result, *scope* could be added as another component for specifying a

policy in the military context. The scope of a policy determines under what circumstances this policy will be included in the policy set for enforcement.

Note that policy control and cognitive capability of agents are completely orthogonal concepts. An agent may be classified as *reactive*, that is, all its components function use specific business logic that will not evolve by itself over time. An agent may be classified as *cognitive*, that is, some or all of its components rely on both sensing and actuation to change their business logic over time. In either case, policy control can be exercised to determine what components should be taking their actions based on the given set of policies.

3 Framework

In this section we describe a new *Policy Agent Framework*, (*PAF*), that has policy control integrated into its agent component architecture. The design of this policy agent framework has been transitioned to the U.S. Army FCS program as the building foundation for its network management system. This agent framework endows its agents with the following major functionalities: policy enforcement, community service, asynchronous event transport, and adaptive communications service. In the following, we first give an overview of these components and describe how policy control is exercised within agents of this framework. We then elaborate on each of the major functionalities in detail.

Since the behavior of agents under this framework is controlled by policies, hereafter we will refer to them as *policy agents*. The behavior of a policy agent is determined by a set of plug-in actions that are associated with policies. Plug-in actions are Java programs inheriting a set of characteristics from their base *action* class, which implements a policy control interface and provides the base implementation of this interface. The set of policies ready for enforcement by a policy agent is dynamically configurable. Therefore, the behavior of an agent can be adjusted according to the consolidation of the goals of the agents and the requirements from their network environment. Policies that are in *activated* state are ready for enforcement by the *Policy Enforcer*, a component of an agent, which coordinates and controls the execution of plug-in actions associated with the enforced policies. During the life cycle of a plug-in action, it may publish events based on monitoring, aggregation, or deduction. The events are published to an asynchronous event bus where an event distributor will notify the appropriate subscribers that have registered to receive the events of interest to them. The primary function of the *Community Service* is to determine a *role* for an agent in the community that it joins. The community service runs in the

background, maintains the role of an agent dynamically by exchanging messages with peers in the same community, and publishes an event to the *Asynchronous Event Bus* whenever a role change event occurs. All communications by the *Plug-in Actions*, *Community Service*, and any other component of an agent use an *Adaptive Communications Service* as their only transport vehicle, which can also be controlled by policies. The purpose is to let the system communications behavior be adjusted dynamically, so that the communications performance can be enhanced and the situational requirements can be observed. We use Figure 1 to illustrate the major components that constitute a policy agent under this policy agent framework.

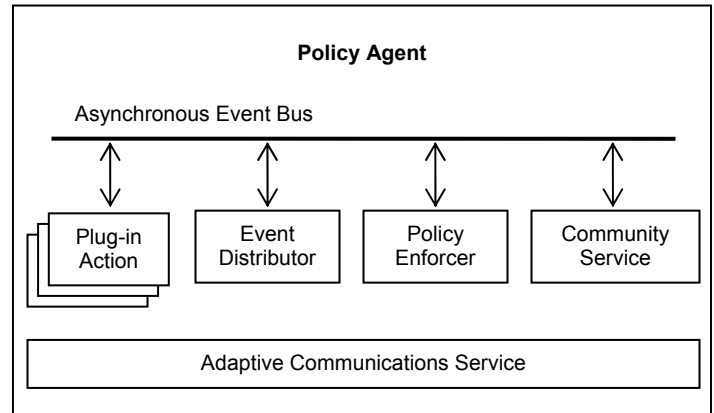


Figure 1. Major components in a PAF agent

3.1 Policy Enforcement

A policy is represented by a four-tuple: $\langle \text{event, condition, action, scope} \rangle$. The policy enforcer component constantly monitors the events that could trigger the enforcement of the activated policies. When the triggering event of an activated policy is received, the policy enforcer evaluates the condition of this policy. If the condition evaluates to true, then the action of this policy is loaded and signaled for execution.

The policy enforcer is also responsible for keeping track of the state of policies. Only policies that are in *activated* state will be enforced. The activations and deactivations of policies influence the behavior of a policy agent system, and could originate from an external authority such as the policy issuer of a policy agent community, or from the environments in which the policy agents are situated.

In any reasonably large set of policies, policy conflicts are bound to occur. Therefore, conflict detection mechanism is needed to determine whether any conflicts can arise when the activated policies set is changed due to the activation of a policy or the deactivation of an already-activated policy. Unfortunately, this detection can only catch a subset of all possible conflicts. Online conflict detection

and resolution mechanisms are also needed to detect and resolve conflicts during the course of policy enforcement.

Plug-in actions are individual software components that have their own specific business logic. They can be inserted into this framework if they conform to the PAF agent programming interface. This conformance allows their run-time behavior to be controlled by policies. When this policy agent framework was designed, its main purpose was to enable the construction of a distributed, agent-based ad hoc network management system. Because many plug-in actions that are needed in this network management system share some common functionality, we have designed an action class inheritance hierarchy to facilitate the development of plug-in actions. By taking advantage of this class inheritance hierarchy, for example, one can easily write new plug-in actions for various monitoring and reporting purposes.

3.2 Asynchronous Event transport

The components of a PAF agent communicate via an asynchronous event bus with traditional publish/subscribe semantics. As described earlier, events are the key to the enforcement of policies. The current implementation does not support the archival of events, nor can it recognize a set of events as one single transaction. This is due to a design philosophy that regards the occurrence of events as sequential, unrepeatable phenomena. The occurrence of any event could change the state of the system immediately and permanently.

From the architectural viewpoint, the inter-agent event transport service is provided by a component external to the agent. This design allows an event to be published to its local asynchronous event bus as well as to those on the other agents, as long as the component responsible for relaying events has been configured to do so. The event relaying component can be configured by the enforcement of policies. Such functionality is essential for a policy agent system in a military wireless ad hoc network because there are situations where the inter-agent event transport service needs to be disabled. As of now the current implementation only allows events to be transported along a hierarchy built by the Dynamic Role Setting Service. This is because global and regional management decisions are generally made by a node with information collected through events from the nodes below it in the hierarchy.

3.3 Adaptive Communications Service

For policy agent systems to operate in a dynamic network environment, it is crucial to use a common communications service that is designed to relieve the communicating entities from dealing with unstable and unreliable networks. Under PAF, all communications must

interface with the adaptive communications service to ensure that the communications behavior of the policy agents is synchronized across the system.

The adaptive communications service features an API for use by the components of policy agents. The API provides an abstraction of the underlying network stack so that it can shield the communicating entities from being concerned about the network dynamicity and agent community changes. For example, assume that the communicating entities need to use reliable message transport so that they can ensure message delivery without having to worry about duplicated packets or packet retransmissions. The entity initiating the conversation only needs to specify its communications requirements via API invocation; in this case it specifies that it needs reliable message delivery. The actual selection of the transport protocol is the responsibility of the adaptive communications service. If the current network conditions do not favor the use of TCP [3], the adaptive communications service can use some other available reliable transport service instead, as long as the specified communications requirements can be satisfied. Moreover, the choice of a suitable transport protocol can change in the course of a communications session without the communicating entities being aware of the switch. Another example of the use of the adaptive communications service is as follows: rather than having to specify the destination of a message, a sender can specify the intended recipient(s) of a message with their abstract community identifiers. The communications service will map the identifiers to network identifiers on the fly so that callers of the API do not need to be concerned about changes to the mappings of abstract community identifiers to the network identifiers, which likely will occur in a dynamic environment.

The other salient feature of the adaptive communications service is its ability to adapt. It has a behavior control interface that accommodates external control from authoritative sources such as policy actions. The advantage of providing this behavior control interface is that it allows the communications behavior of policy agents to be changed dynamically based on mission requirements. In the military context, mission requirements could change dynamically due to unexpected changes in battlefield status, changes in mobility patterns, etc. Different situations may require different communications behavior, such as varying compression, encryption, authentication requirements, or a need to enable or disable store and forward message relay services [1][2], etc.

A detailed description of the design of the adaptive communications service can be found in [17].

3.4 Community Service

Our policy agent framework was designed to support large scale, fully distributed, multi-policy agent systems. Therefore, it needs to provide an array of community services to enable robust and resilient operations for communities of policy agents. Each community shares one common set of policies as its global goals/requirements. The community services that are currently provided under this agent framework include the following: Dynamic Role Setting Service, Naming Service, and Policy Distribution Service. Below we describe each of these services.

Dynamic Role Setting Service: In our PAF model, policy agents have *roles*, which are used to determine their responsibilities. This design decision is related to the “*scope*” component that is included in the specification of a policy (recall that a policy was defined as a four-tuple: $\langle \text{event}, \text{condition}, \text{action}, \text{scope} \rangle$). The role of an agent determines which policies it will enforce. Our current implementation creates a hierarchical tree-structured relationship among policy agents; the position of a policy agent within this tree structure is determined by its role. One of the design objectives of our PAF is to enable distributed agent operations in wireless ad hoc networks. As a consequence, we have built a resilient role setting service that can dynamically adjust the roles of agents in the face of unstable network connectivity and unpredictable community participation in the military ad hoc environments. Dynamic agent role adjustment is necessary because agents can dynamically enter and leave the network, thus requiring mechanisms to dynamically rebuild the policy agent hierarchy in response to such events.

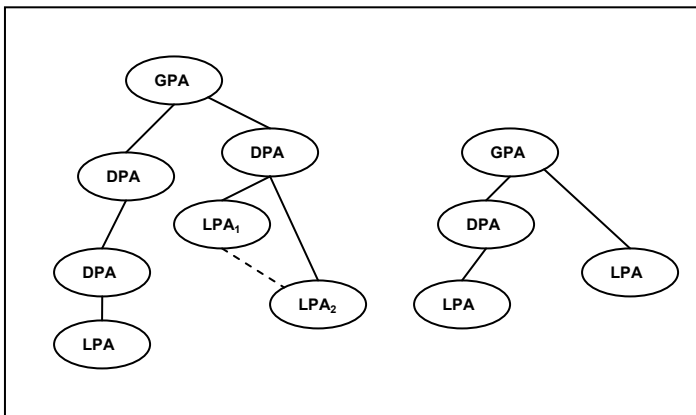


Figure 2. Policy agents in two communities

Here we use Figure 2 to illustrate the concept of policy agent hierarchy, policy agent community, and the dynamic role setting service that builds and maintains a policy agent hierarchy. There are two hierarchies shown in Figure 2 and each forms its own agent communities. The policy agent on the top of the hierarchy is the Global Policy Agent

(GPA), the ones that are neither the root node nor leaf nodes are Domain Policy Agents (DPAs), and the ones that are leaf nodes are Local Policy Agents (LPAs). The role of a policy agent can change on the fly, depending on the algorithm being used to autonomously form a policy agent hierarchy. For example, suppose the hierarchy maintenance algorithm takes the network connectivity into account. If LPA_2 loses network connectivity to its GPA due to its movement and later reconnects to the network with LPA_1 , LPA_1 may turn itself into a DPA that has LPA_2 under its management.

In addition, the sets of policies enforced at these two policy agent communities do not have to be identical, and actually they can be totally different. This reflects the command structures in the battlefield. The armored units and the infantry units may use different sets of policies for their networks and form their own policy agent hierarchies.

Naming Service: A *Naming Service* is needed to work in conjunction with the *Dynamic Role Setting Service*. As mentioned above, policy agents may change their roles during the course of their activities. Instead of providing naming service in a traditional sense, that is, mapping agent names to IP addresses via DNS, the Naming Service under PAF maps agents to the roles that they are assuming. The major advantage of using such a design is to separate the business logic from the need to keep track of connections based on network IDs in a dynamic environment like a wireless ad hoc network. Instead, communications can be established based on agent roles. For example, the current implementation of the Naming Service stores the following types of roles relative to the hosting policy agent: *parent*, *children*, *neighbors*, and *root*. If a policy agent needs to communicate with its parent, it will use “parent” as the message recipient, and leave the work of sending the message to the parent policy agent to the adaptive communications service. The latter looks up the recipient’s network identifier (e.g. IP address) using the Naming Service.

Policy Distribution Service: A Policy Distribution Service is provided to facilitate the dissemination of policies and synchronize their state changes. Policies can be disseminated to all policy agents so that they all behave in a consistent manner, e.g. to implement policies for a given mission. The current implementation assures dissemination robustness by embedding a synchronization mechanism into policy agents to handle unreliable and possibly intermittent network connectivity. This service enables policy agents in a community to operate with the same set of policies whenever they have network connectivity between them.

4 Application

We have implemented this policy agent framework and used it to build a wireless ad hoc network management system prototype under the DRAMA program of the U.S. Army CERDEC [4]. This management system was designed to provide distributed, policy-based network management to manage mobile ad hoc networks. Due to the design decision of building a management system exhibiting deterministic behavior, only reactive agents and not cognitive agents were constructed for this system. Through a number of TRL-5/TRL-6 demonstrations, this system was shown to have achieved the following objectives:

- *Enable modular and customizable distributed management:* The ability to flexibly insert plug-in actions, in conjunction with the policy control, allows this system to perform any specified mission objectives in a dynamically configurable, distributed fashion.
- *Provide autonomous and adaptive management functions:* By integrating policy control with agent technologies, human intervention is reduced due to the ability of this multi-agent system to autonomously adapt itself to a wide range of environmental condition changes.
- *Accommodate large-scale deployment through self-organization and self-healing:* Dynamic role setting in conjunction with an adaptive communications service allows the system to scale in the wireless ad hoc environment.

5 Benefits

In this section, we compare the policy agent framework presented in this paper with the other well-known agent frameworks. The main benefits of our PAF are discussed below.

First, to the best knowledge of the authors, this is the first report on incorporating policy control into an agent framework that has then been used to build a real-life distributed application. The advantages of using policy-controlled agents for this application are:

- *Dynamic objectives adjustment.* By using policies as an overarching coordination mechanism between the plug-in actions, the objectives of agents can be adjusted based on need.
- *Separation of application and network concerns:* When agents are used in a stable network environment, the effect of network dynamics can be more or less ignored by the agents. However, in the wireless ad hoc military context, network dynamics have to be accounted for by applications. Using policies is one promising approach to make a compromise between the autonomy of agents

and the desired deterministic behavior from the perspective of system administration.

Second, this agent framework was designed from the ground up with a stringent set of requirements derived from the wireless ad hoc network environment. In contrast to Cougar and other agent frameworks, we exclude Internet protocols that are known to have excessive bandwidth requirements from being building blocks for this agent framework. This is because the typically scarce network bandwidth and intermittent network connectivity in wireless ad hoc settings very likely will cause these protocols to perform poorly. Our approach is to introduce an adaptive communications service as the substrate for supporting inter-agent communications. This enables agents to be deployed in dynamic environments without adding the extra implementation complexity of handling communications. The other benefit is that the communications service is also subject to policy control. As a result, communications behavior can be adapted to achieve the objectives of both performance optimization and responsiveness to mission changes.

6 Discussion

Although we have made some progress in terms of incorporating policy control into an agent framework, there are many research issues that await further study.

- *System-wide event transport:* In our experience, there are situations where a more generic event transport system may be needed for relaying events. However, there are many issues with such a system, including its semantics, unreliable network connectivity, message delivery delay, the cost of rollback operations, and so on. Basically this could be regarded as a generic problem of building a distributed system over unreliable networks. Moreover, the participants of the distributed system may not be fixed. This is a big research area that calls for further investigation.
- *Policy language:* Although the policy language that we have currently implemented allows us to exercise any policies we need, there is room for further enhancement in order to make the language more expressive.
- *Communication service API:* We have defined the framework for this API. However, the networking API spans a wide array of disciplines. More modules need to be added to cover additional communications requirements in the areas of security, authentication, data compression, transmission reliability, and so on.
- *Policy synchronization:* The current implementation only allows policies to be injected into the system at the root of a policy agent community. Future work will examine the possibility of a more distribution policy creation mechanism.

- Role Setting Service algorithms: The outcome of the Role Setting Service algorithm dictates the performance of a policy agent system in the wireless ad hoc network environment. We have implemented a few candidate algorithms and currently are in the process of evaluating their performance in the target environment.
- Conflict detection and resolution. We have partially implemented the conflict detection logic to identify static policy conflicts. More elaborate online conflict detection and resolution mechanism is still under study.

7 Summary

In this paper, we presented a novel software agent framework with seamlessly integrated policy control. We first described the concept of policy control and how it is integrated into this framework. This was followed by a description of policies in action and a detailed discussion of the functions and benefits of policy control. Finally, we described agent behavior adaptation that enables agents to achieve their goals and simultaneously deal with dynamic network environments and changing mission plans.

References

- [1] K. Fall, "A Delay Tolerant Network Architecture for Challenged Internets," Proceedings of SIGCOMM'03, August 25-29, 2003, Karlsruhe, Germany.
- [2] "Disruption Tolerant Networking", BAA from DARPA 2004, <http://www.darpa.mil/ato/solicit/DTN/>.
- [3] H. Balakrishnan, S. Seshan, E. Amir and Randy H. Katz, "Improving TCP/IP Performance over Wireless Networks," In Proceedings of the 1st ACM International Conference on Mobile Computing and Networking (Mobicom), November 95.
- [4] R. Chadha, Y-H Cheng, C. Chiang, S. Li, and G. Levin "Policy-Based Mobile Ad Hoc Network Management", Proceedings of the IEEE 5th International Workshop on Policies for Distributed Systems and Networks, Yorktown Heights, New York, June 7-9 2004.
- [5] C. S. Ram Murthy, B. S. Manoj, "Ad Hoc Wireless Networks: Architectures and Protocols," Prentice Hall, May 2004.
- [6] US Army, "Future Combat Systems", <http://www.army.mil/fcs/>
- [7] US Army, "Warfighter Information Network-Tactical", <http://peoc3t.monmouth.army.mil/WIN-T/WIN-T.html>
- [8] J. Li, C. Blake, D. S. J. De Couto, H. I. Lee, and R. Morris, "Capacity of Ad Hoc Wireless Networks", MobiCom, 2001.
- [9] Information Operations Condition, <https://infosec.navy.mil/Documents/>
- [10] J. Ferber, "Multi-Agent Systems, An Introduction to Distributed Artificial Intelligence," Addison Wesley, 1999.
- [11] A. Helsing, M. Thome, and T. Wright, "Cougaar: A Scalable, Distributed Multi-Agent Architecture",

- Proceedings of the International Conference on Systems, Man and Cybernetics, 2004.
- [12] Cougaar website, <http://cougaar.org>
 - [13] JADE website, <http://jade.cselt.it>
 - [14] Grasshopper website, <http://www.grasshopper.de>
 - [15] Aglets website, <http://aglets.sourceforge.net>
 - [16] DARPA UltraLog program website, <http://ultralog.net>
 - [17] C. J. Chiang, R. Chadha, G. Levin, S. Li, Y.-H. Cheng, A. Poylisher, "AMS: An Adaptive Middleware System for Wireless Ad Hoc Networks," to appear in Proceedings for Milcom'05, October 17-20, Atlantic City, USA