

# POLICY-BASED MOBILE AD HOC NETWORK MANAGEMENT FOR DRAMA \*

Ritu Chadha, Yuu-Heng Cheng, Jason Chiang, Gary Levin, Shih-Wei Li, Alexander Poylisher  
{chadha,yhcheng,chiang,gary,sli,sher}@research.telcordia.com  
Telcordia Technologies, One Telcordia Drive, Piscataway NJ 08854.

## ABSTRACT

*Ad hoc networking is the basis of the future military network-centric warfare architecture. Such networks are highly dynamic in nature, as mobile ad hoc networks are formed over wireless links that are susceptible to failure. Strict requirements on security and reliability combined with the dynamic nature of the network provide a strong motivation for self-forming, self-configuring, and self-healing capabilities in the network. This paper describes a policy-based mobile ad hoc network management system that addresses these needs. The system provides the capability to express networking requirements in the form of policies at a high level and have them automatically realized in the network by intelligent agents. Our system provides the following capabilities: flexible monitoring and reporting that enables collection of management information from network elements at configurable intervals; automated configuration and re-configuration of network elements based on reported network status; user-definable aggregation and filtering of monitored management information at the source of the data so as to reduce management station processing and network transmission overhead.*

## 1. INTRODUCTION

Today's military networks are mobile ad hoc wireless networks, where every node acts as a router and can route traffic to other nodes. Such networks pose stringent requirements for security and reliability. They are highly dynamic in nature, as mobile ad hoc networks are formed over wireless links. Links are susceptible to failures because of mobility of nodes, or loss of connectivity due to volatility of wireless links. Strict requirements on security and reliability combined with the dynamic nature of the network provide a strong motivation for self-forming, self-configuring, and self-healing capabilities in the network. In order to address these needs, we have developed a policy-based network management system

that provides the capability to express networking requirements at a high level and have them automatically realized in the network by configuration agents. This approach provides the network administrator with the capability to specify high-level policies that:

- Specify *long-term, network-wide* configuration objectives, e.g. all control traffic must get the highest level of QoS priority; all private communications must be encrypted.
- Provide an *automated feedback loop* so that information reported by monitoring agents can be used to automatically trigger correction of network problems based on policies, e.g. "If server response time > 5 seconds, determine whether to relocate the server".

Once policies such as those described above are defined, they are automatically enforced by the management system. These capabilities can provide military personnel with very powerful tools to configure and control their network, and to re-configure their network in response to network conditions. Some examples of added functionality enabled by policy-based management include:

- Dynamically changing the role of a node to act as a server (e.g. DNS server), based on relevant capabilities such as computing power, battery power, signal strength, hardware, etc. Capabilities that are important in a certain environment may not be relevant in others (e.g. battery power may be important at night but not during the daytime for solar-powered nodes); such constraints are captured as policies.
- Switching between proactive and reactive routing protocols to optimize performance depending on the known density of nodes in the network.

---

\* Prepared through collaborative participation in the Communications and Networks Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Contract DAAD19-01-C-0062. The U. S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

- Adjusting auto-configured address pool sizes based on density of nodes in the network to conserve address space.
- Setting network-wide values for auto-configuration parameters; etc.

This paper describes a policy-based mobile ad hoc network management system, developed under the U.S. Army CERDEC DRAMA (Dynamic Re-Addressing and Management for the Army) program [3], that addresses the special needs posed by mobile ad hoc networks. The system provides the capability to express networking requirements at a high level and have them automatically realized in the network by agents, without requiring further manual updates. Network management functionality is realized by policy agents that are organized in a hierarchy to provide both scalability and autonomy. Survivability is achieved by enabling any component to take over the role of another component in the case of failure.

### 1.1. Related work

The characteristics of mobile ad hoc networks (commonly called *MANETs*) are sufficiently different from commercial wireline networks to have generated a great deal of work in alternate management paradigms. The task of managing MANETs involves frequent reconfiguration, due to node mobility and consequently dynamically changing network topology. Network nodes often have limited battery power and storage capacity, and wireless radio link capacity and quality varies dynamically based on environmental conditions (weather, terrain, foliage, etc.). These differences have resulted in an emergence of paradigms particularly suited for managing MANETs. As an example, although SNMP is the management protocol of choice for monitoring the vast majority of network elements available on the market today in wireline networks, alternative protocols have sprung up that are specifically designed for MANETs. ANMP (Ad hoc Network Management Protocol) [16] is a management protocol that is compatible with SNMP [18], and uses hierarchical clustering of nodes to reduce the number of messages exchanged between the manager and the agents. In the management system described in this paper, we make use of YAP (Yelp Announcement Protocol) [7] for efficiently reporting management information. YAP relies on periodic information reporting from agent to manager instead of having a manager that regularly polls network elements for status (as is done with SNMP). Further, the U.S. Army CERDEC MOSAIC Ad hoc Mobility Protocol Suite (AMPS) [15] provides mechanisms for dynamically merging and splitting

networks, which we have adopted in our testbed. The work in this paper describes a management system for managing networks running AMPS protocols.

Policy-based networking [4],[5] is a powerful approach to automating network management, as evidenced by the numerous industry efforts in this area dealing with diverse networking domains, e.g. quality of service control [1], [8], traffic engineering [6], etc. A policy implies a pre-determined action pattern that is repeated by an entity whenever certain system conditions appear. In 1997, the idea of modeling and using policies for network management resulted in the creation of the Directory-Enabled Networks (DEN) initiative, which was later incorporated into the Common Information Modeling (CIM) effort in the DMTF (Distributed Management Task Force) [14], [9]. More recently, the use of policy-based management for QoS management in wireless ad hoc networks has been studied; [17] proposes a solution suite including k-hop clustering, inter-domain policy negotiation, and automated service discovery.

The IETF Policy Framework work group [10] was created to leverage and extend standardization of policy information modeling from the CIM/DEN working groups in the DMTF. The output of this working group includes the Policy Core Information Model (PCIM) [11], [12], which defines object classes that represent generic constructs such as policy rules, conditions, actions, policy groups, etc.

### 1.2. Driving requirements

The purpose of this work is to develop a network management system for large mobile ad hoc networks. In order to achieve this, we have developed the following driving requirements:

- Scalability: The network management system must scale to be able to manage a mobile ad hoc network consisting of thousands of nodes.
- Autonomy: In face of the dynamic nature of the managed network, it is necessary that any resulting change in management role/ functionality happen autonomously. Such changes may be necessary to adjust to changes such as reduced bandwidth available for management traffic, mobility, etc. The network management system must be able to adjust to environment changes.
- Low bandwidth consumption: One salient characteristic of a mobile ad hoc network is scarce and variable bandwidth. This makes it essential that the network management system impose very little traffic overhead on the network. Another consequence of variable bandwidth is that the network management

system must be able to autonomously adjust its bandwidth consumption based on available bandwidth.

- **Survivability:** Given the nature of mobile ad hoc networks, it is necessary to make the network management system survivable. This is due to the dynamic nature of such networks, as nodes can move in and out of the network and may be destroyed for various reasons such as battlefield casualties, low battery power, etc. Thus there should be no single point of failure for the network management system.
- **Minimal human intervention:** The complexity of mobile ad hoc networks makes it imperative that any management system for such networks minimize the amount of human intervention required to achieve the desired network performance. This includes automatic fault detection and remediation as well as automation of component configuration.

### 1.3. System Overview

The high-level architecture of our system is shown in Figure 1. As shown here, a collection of Policy Agents manage all the nodes in the mobile ad hoc network. At the highest level, the Global Policy Agent, or GPA, manages multiple Domain Policy Agents, or DPAs. A DPA can manage multiple DPAs or Local Policy Agents (LPAs). An LPA manages a node. LPAs perform local policy-controlled configuration, monitoring, filtering, aggregation, and reporting, thus reducing management bandwidth overhead. Policies are disseminated from the GPA to DPAs to LPAs, or from DPAs to LPAs. Policy Agents react to network status changes on various levels (globally, locally, domain-wide) by automatically reconfiguring the network as needed to deal with fault and performance problems. In this architecture, any node can dynamically take over the functionality of another node to ensure survivability. A flexible agent infrastructure allows dynamic insertion of new management functionality.

This paper is organized as follows. Section 2 gives an overview of our system design and describes all the components of the management system. The kinds of policies that are used in the system are described in Section 3. Our implementation experience is described in Section 4, and future work is outlined in Section 5. We conclude with a summary in Section 6.

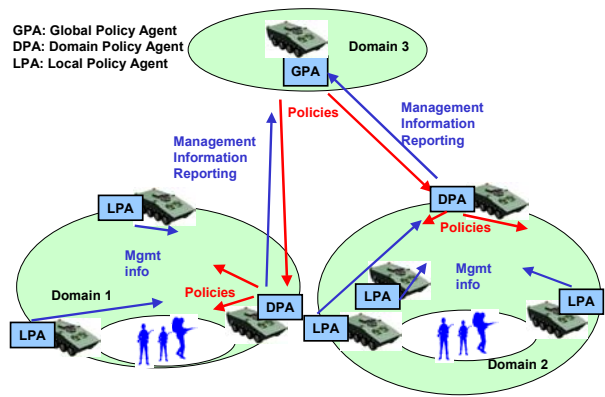


Figure 1. High-level architecture

## 2. INTELLIGENT AGENT-BASED NETWORK MANAGEMENT SYSTEM DESIGN

As shown in Figure 1, the management system is organized into a number of policy domains. A policy domain is a collection of entities subject to the same set of policies and managed by a policy agent, known as the Domain Policy Agent. A policy domain may contain one or more policy domains, organized in a hierarchical structure.

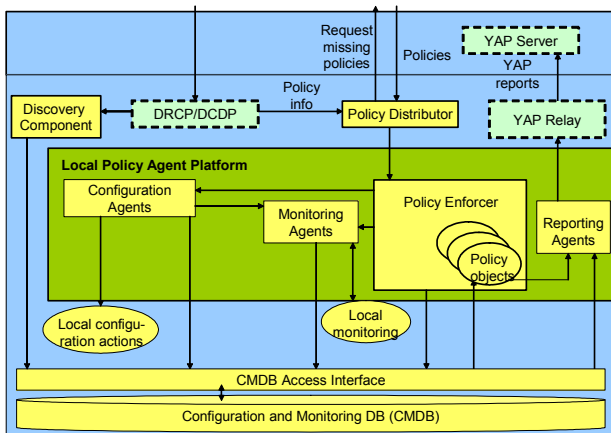
A Policy Agent is an entity responsible for managing and enforcing the policies of a policy domain. The Policy Agent of an atomic policy domain (i.e. one that does not contain any other policy domain) is referred to as a Local Policy Agent (LPA). The Policy Agent of the top-level policy domain is referred to as the Global Policy Agent (GPA). Intermediate Policy Agents are referred to as Domain Policy Agents (DPA). The Data Distribution Service (DDS) provides the mechanism for distributing policies and for reporting information. Policies created at the GPA are distributed to the DPAs which, in turn, distribute them to the LPAs. The *scope* of a policy determines whether the policy is applicable to a particular policy domain. The LPAs collect local management information and report the information to higher-level policy agents using an enhanced version of the YAP protocol [7]. This section provides an overview of the components of our policy-based management system, depicted in Figure 2.

**Policy Agent Platform:** A GPA, DPA, or LPA have the same basic structure and consist of the same code base. They have similar functionality but different scope. Different agents can be installed within a GPA, DPAs, and LPAs to enable different functionality.

**Policy Enforcer:** The Policy Enforcer is the entity responsible for enforcing policies. Policies consist of event, condition, action, and scope, and are described in more detail in Section 3. The Policy Enforcer monitors events and evaluates conditions to decide which agent should be instructed to perform its management actions.

When a policy is activated, the Policy Enforcer listens for the event associated with that policy, if the policy definition includes an event. When this event is received, the Policy Enforcer identifies all the policies that are triggered by that event. For each of these policies, the Policy Enforcer evaluates the policy condition, if one exists. If the condition evaluates to true, the corresponding action is performed by instructing the appropriate agent to perform the action. The Policy Enforcer can receive events published by other system components via an Event Bus. For a policy that does not contain any event or condition, the Policy Enforcer immediately launches the appropriate agent to perform the policy action. This design provides an easy and flexible way of adding new management functionality. Each agent implements a standard interface that enables the Policy Enforcer to communicate with the agents. The Policy Enforcer is then able to launch agents, set/modify their parameters, and terminate them. Management policies can therefore be defined in terms of the actions that could be performed by these agents. The standard interface also enables other systems to query the agents to determine their functionality and associated parameters.

- **Configuration Agents:** These agents perform configuration on a local managed element, or node.
- **Monitoring Agents:** These agents collect information at configurable intervals from the local node and store it in the CMDB (Configuration and Monitoring Database).
- **Reporting Agents:** Reporting agents report two types of information: monitoring and configuration information. The information is reported at configurable intervals via YAP. Information monitored by Monitoring Agents or produced by Aggregation Agents may be reported on demand by Reporting Agents.
- **Aggregation Agents:** These agents perform aggregation of locally collected data before the information is reported. For example, an Aggregation Agent may compute the average value of an attribute over a given period of time using the last five monitored values of the attribute.
- **Filtering Agents:** Filtering Agents report information via YAP only when certain conditions are met. For example, a policy can be created that specifies that battery power should only be reported when it drops below a certain threshold.
- **General-Purpose Agents:** General-Purpose Agents are designed to perform special management functions. For example, a General-Purpose Agent may implement an algorithm that determines which node to move a server to, when the current node hosting the server is experiencing performance degradation.



**Figure 2. System components**

**Management Agents:** Management Agents are instantiated when policies are activated and triggered as described in the previous section. These management agents perform different types of actions. Currently, the following types of management agents have been defined:

**Policy Distributor:** The function of the Policy Distributor is to receive policy updates from a remote node and to send these updates to the Local Policy Agent on its node. Policies are disseminated to all the nodes that need to implement the policies in accordance with the defined policy scope. Policies will typically be created via a GUI, or by other related management systems. The Policy Agent Interface at the GPA disseminates policies to the LPAs using a survivable mechanism described in [2].

**YAP Server and Relay:** YAP (Yelp Announcement Protocol) [7] is a lightweight management protocol designed for the MANET environment and used for reporting management information to a higher-level manager. Rather than using an SNMP-like polling paradigm for gathering management information, YAP enables a managed element to periodically or asynchronously report management information. YAP is designed to efficiently and robustly relay (via *YAP relays*) network management information about a node to a

designated node running a *YAP server* that collects and stores this information in a database. Other applications can then query this information as needed.

**DRCP/DCDP:** The Dynamic Configuration Distribution Protocol (DCDP) [19] is a scalable, lightweight protocol designed to distribute configuration information. DCDP relies on the Dynamic and Rapid Configuration Protocol (DRCP) [19] to actually configure the interfaces. DRCP borrows heavily from DHCP, but adds features critical to roaming users. DRCP can automatically detect the need to reconfigure (e.g., due to node mobility) through periodic advertisements.

**CMDB:** The Configuration and Monitoring Database (CMDB) is used to store configuration and monitoring data on a node. This information is stored locally on every node in a repository implemented as a MySQL database server. Configuration data is written to the CMDB by Configuration Agents, and monitoring data is written to the CMDB by Monitoring Agents.

The CMDB database schema is the same on an LPA, DPA, or GPA. The difference is that on a node that is not acting as a DPA or GPA, the CMDB will only contain information about the local node; whereas on a node that is acting as a DPA or GPA, the CMDB will contain information about the local node as well as about all the nodes within its management jurisdiction. On a DPA/GPA, data about nodes other than the local node is obtained from remote nodes via YAP reporting and is stored in the CMDB by the YAP server.

**Discovery Component:** The Discovery Component is an entity responsible for initializing the Local Policy Agent (LPA) configuration and monitoring database (CMDB), for updating the LPA CMDB when values are changed by external software components, and for accessing the CMDB on behalf of these components.

### 3. POLICIES

Policies provide a means of specifying the desired behavior of a network at a high level. They are implemented and enforced by Policy Agents, and stored in a Policy Repository implemented as a MySQL database server. Policy enforcement involves translating the specification of desired network behavior into appropriate monitoring, evaluation, filtering, aggregation, configuration and reporting actions that establish and maintain the desired network behavior. Policies can have four components:

**Event:** An event is a named event in the local policy domain that triggers execution of a policy. Whenever the

event occurs, the policy condition (if present) is evaluated. If it evaluates to true, the action is executed.

**Condition:** A condition is a binary expression that is evaluated by a Policy Agent to determine if the corresponding action needs to be executed for a given scope of the policy.

**Action:** An action describes a task to be executed. Actions may need additional parameters to define actual operations that will be carried out by the action. Thus an action can be sub-divided into two components: action name (sometimes referred to as action *agent*) and action parameters. Note that actions are performed by executing agent code (agents were described in the previous section). Thus new actions can be introduced into the system by writing code to implement new agents.

**Scope:** The scope of a policy indicates its targets, i.e. at which nodes it should be enforced. Currently, two scopes are supported: *GPA* and *All\_LPAs*. A scope of *GPA* indicates that the policy is to be enforced only at the GPA; whereas a scope of *All\_LPAs* indicates that the policy is to be enforced at all LPAs.

Event, Condition and Scope are optional. If neither Event nor Condition are present in a policy, the policy action is immediately executed. If Event is not present but Condition is, an implicit event is assumed which triggers condition evaluation every time the variable operand of the condition changes. If a Condition expression is not provided, it defaults to *true* so that the Action is executed immediately after policy execution is triggered. In the absence of a Scope, the policy scope defaults to "*All\_LPAs*", i.e. universal scope.

## 4. IMPLEMENTATION

The management system described in this paper has been implemented under the U.S. Army CERDEC DRAMA program and demonstrated at Technology Readiness Level 5 (TRL-5). The software was implemented on Red Hat Linux release 8.0. Our testbed is shown in Figure 3 and consisted of two wireless 802.11b ad hoc routing domains running OLSR (Oscar and Emmy in the figure below) connected by a wired backbone network running RIP. We installed the MOSAIC AMPS protocols in our MANET testbed. Infrastructure servers such as a SIP server (for mobility management as described in [15]) and a Bandwidth Broker (for QoS management as described in [13]) were placed on machines in the wired backbone. Two desktop machines (Oscar and Emmy) were used as border routers for the two ad hoc routing domains, and the other two desktop

machines (DRAMA-gw and Grammy) were used to host the infrastructure servers described above. Two laptop computers were used in each ad hoc routing domain as the mobile nodes. Each laptop had an 802.11b wireless interface card, as did the two border routers Oscar and Emmy. The GPA was running on the DRAMA-gw machine, and LPAs were running on all other machines in the network. A Policy Manager GUI was used on DRAMA-gw to create and modify policies. For a detailed description of the demonstrated scenarios, see [2].

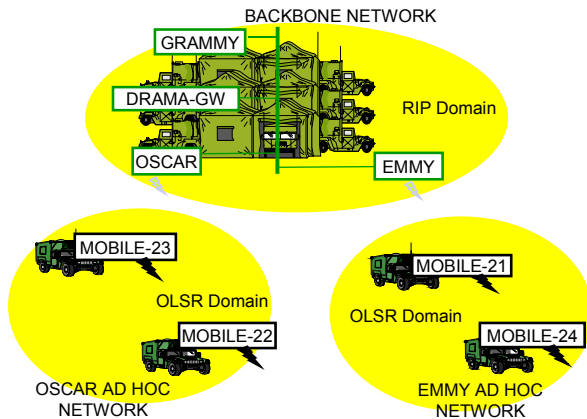


Figure 3. Mobile ad hoc network testbed

## 5. FUTURE WORK

The system described in Section 4 implements a two-level management hierarchy and does not allow dynamic formation and maintenance of a multiple-level management hierarchy. A dynamically adjustable multi-tier hierarchy enhances the scalability of the management system by expanding or shrinking the number of tiers in the hierarchy depending on network conditions. The basic idea is to form “clusters” of nodes for management purposes and then link clusters into a tree-like structure. This limits the number of nodes under the control of a single manager, thereby providing scalability. Cluster members include one single cluster leader and a set of children, known as cluster associates. Leaders of several clusters can form another cluster. A domain is the set of all associates in a subtree of this hierarchy. An example of this is shown in Figure 4.

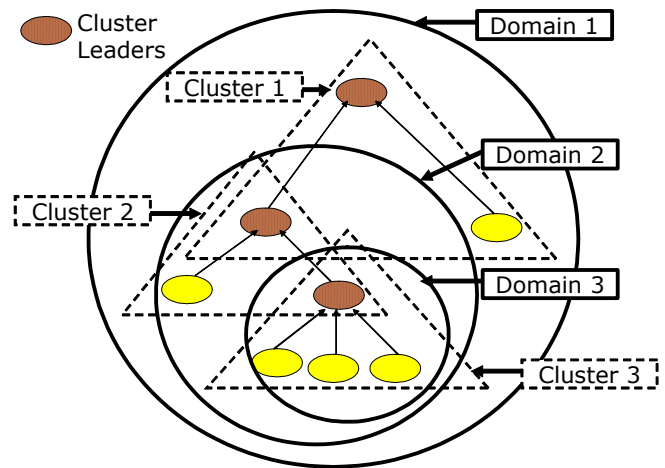


Figure 4. Multi-tier management hierarchy

Formation and maintenance of this management hierarchy is accomplished as follows. For *initial cluster* formation, clusters and their leaders are specified by a pre-defined network plan (i.e. manually). After the initial formation of clusters, it is necessary to *maintain* the management hierarchy for distributing policies and collecting management information. This is achieved using the following mechanisms. The leader of a cluster periodically sends heartbeat messages to all its cluster associates, which is acknowledged by each cluster associate. If a cluster associate misses a number of consecutive heartbeat messages from its leader, it assumes that the leader no longer exists and tries to join the cluster of an ancestor of its current parent. If it fails to do so, it assumes the role of the GPA, i.e. the root of the management hierarchy. A cluster leader may decide to *split* its current cluster into two clusters, based on network conditions (e.g. if a cluster becomes geographically dispersed and management performance is adversely impacted). The cluster leader then appoints one of its associates as the leader for a subset of its associates. The new leader remains a child in the current cluster. If an associate detects the existence of another GPA, it notifies its own GPA, who then initiates a negotiation session with the other GPA. The other GPA signals the session initiator its decision to remain a GPA or to become a child of the session initiator. Once the session initiator acknowledges this decision, the two domains merge and one of the GPAs steps down to become a child of the other.

## 6. SUMMARY

The management system described in this paper was prototyped and demonstrated in a realistic environment. Our implementation results showed that a substantial reduction in management traffic overhead is achievable by controlling the content and frequency of management information reporting, and by aggregating and filtering management information at the source before sending it across the network. We also demonstrated the ability to automatically trigger reconfiguration actions in the network in response to network status, thus achieving a high degree of management automation, and improving the performance of the network. Future work will focus on implementation of the multi-tier management hierarchy described in the previous section. This work is also being transitioned to the U.S. Army under the SDD (System Design and Development) phase of the Future Combat Systems program.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U. S. Government.

## REFERENCES

- [1] R. Bhatia et al., "Policy Evaluation for Network Management", INFOCOM 2000.
- [2] R. Chadha et al., "Policy-Based Mobile Ad Hoc Network Management", Policy 2004, Yorktown Heights, New York, June 2004.
- [3] R. Chadha et al., "System Design Report", CERDEC Intelligent Agent-based Management for the Army Environment Project Deliverable, Contract DAAD19-01-C-0062, CDRL A003, Sept. 2003.
- [4] R. Chadha et al., "PECAN: Policy-Enabled Configuration Across Networks", Policy 2003, Como, Italy, June 2003.
- [5] R. Chadha, G. Lapiotis, S. Wright, "Policy-Based Networking", IEEE Network, March/April 2002, Vol. 16 No. 2, guest editors.
- [6] R. Chadha et al., "Policy Framework MPLS Information Model for QoS and TE", <http://search.ietf.org/internet-drafts/draft-chadha-policy-mpls-te-01.txt>, Dec. 2000.
- [7] C. Chiang et al., "Generic Protocol for Network Management Data Collection and Dissemination", MILCOM 2003, Oct. 2003.
- [8] A. Chiu, S. Civanlar, R. Rajan, "A Policy based approach for QoS on demand over the Internet", 8<sup>th</sup> Intl. Workshop on QoS (IWQoS), Pittsburgh, 2000.
- [9] DMTF CIM Standards, <http://www.dmtf.org/spec/cims.html>.
- [10] IETF Policy Framework, <http://www.ietf.org/html.charters/policy-charter.html>.
- [11] B. Moore et al., "Policy Core Information Model Extensions", IETF RFC 3460, January 2003.
- [12] B. Moore et al., "Policy Core Information Model - Version 1 Specification", IETF RFC 3060, February 2001.
- [13] I. Sebuktekin et al., "Initial DS/BB Protocol Design Document, Version 1.0", CECOM MOSAIC Project, Contract DAAB07-01-C-L534, January 2002.
- [14] J. Strassner, "Directory Enabled Networks", Tech. Series, New Riders Publishing, 1999.
- [15] K. Young et al., "Ad Hoc Mobility Protocol Suite for the MOSAIC ATD", MILCOM 2003.
- [16] W. Chen, N. Jain and S. Singh, "ANMP: Ad hoc Network Management protocol", IEEE Journal on Selected Areas in Communications 17(8) (August 1999) 1506-1531.
- [17] K. S. Phanse, "Policy-Based Quality of Service Management in Wireless Ad Hoc Networks", Ph.D. thesis, 2003, <http://scholar.lib.vt.edu/theses/available/etd-09082003-110529/>.
- [18] D. Harrington, R. Presuhn, B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", RFC 3411, 2002.
- [19] K. Manousakis et al., "Routing domain autoconfiguration for more efficient and rapidly deployable mobile networks", 23<sup>rd</sup> Army Science Conference, Dec. 2002.