Published in the UBICOMM Conference October 2010

Coordination and Control in Mobile Ubiquitous Computing Applications Using Law Governed Interaction

Rishabh Dudheria, Wade Trappe
WINLAB
Rutgers University
North Brunswick, NJ 08902, USA
Email: {rishabh,trappe}@winlab.rutgers.edu

Naftaly Minsky
Department of Computer Science
Rutgers University
Piscataway, NJ 08854, USA
Email: minsky@cs.rutgers.edu

***Abstract*—This paper introduces a mechanism for regulating the interactions between the members of an ad hoc, heterogeneous and mobile multi-agent system, in order to ensure reliable and secure coordination between them. We demonstrate this mechanism, and its importance, by describing its application to a police team whose mission is to manage (i.e., monitor and control) the traffic in an area, by operating on a set of traffic-related devices, such as draw bridges, traffic lights, and road blocks. In particular, we demonstrate how the following critical aspects of the working of such a team are provided for: a) reliable coordination between the team members; b) the ability of the leader of the team to steer its subordinates; c) reliable auditing of the operations of the team; and d) robustness of the team under certain unexpected adverse conditions, such as the unpredictable failure of the team leader. Beyond developing suitable formalisms for local regulation of actions and communications, performance tests have been conducted with the proposed implementation on the ORBIT testbed and the results presented show the viability of this approach.**

***Keywords*-Law Governed Interaction; ad hoc coordination; decentralized enforcement; security**

I. Introduction

Current mobile ubiquitous technology supports reasonably reliable and secure communications between pairs of agents. It does not, however, provide adequate support for ad hoc, heterogeneous, multi-agent systems, whose members need to coordinate dynamically with each other in order to carry out their function—whether they operate in a collaborative or competitive mode, or some mix of the two. Such dynamic coordination is required in many application domains, such as in *law-enforcement*, and *military* applications, where an ad hoc team of diverse individuals is assembled to carry out a complicated and open-ended mission; coordination is also required in an *impromptu marketplace* where consumers may interact with each other via their wireless devices to share content and trade various digital tokens; and in various applications involving *vehicular communications*.

Effective and trustworthy coordination, however, requires participants to conform to a common coordination protocol. For example, for car drivers to survive their passage through an intersection, they must coordinate with other car drivers. But this requires all drivers to comply with certain traffic laws, like the one that requires stopping at a red light.

The goal of this paper is, therefore, a reliable and secure mechanism for establishing common coordination protocols over ad hoc multi-agent systems, whose members interact with each other via wireless communication. We illustrate the importance of such a mechanism, and some of its required characteristics, via the following example.

An Ad Hoc Police Team Mission: Consider a team of police officers, whose mission is to manage (i.e., monitor and control) the traffic in a certain region. In particular, the team is responsible for operating a set of traffic-related devices, such as draw bridges, traffic lights, or road blocks. This they can do via a collection of sensors and actuators distributed in their domain. Moreover, suppose that the team is managed by a leader, who assigns the team members to various tasks, monitors their progress, and exerts control over what each team member can do.

For such a team to operate effectively and safely, it must operate according to an appropriate protocol—which we shall denote by P—that regulates the interaction among the team members, and between them and the various actuators. This protocol should facilitate effective coordination between the team members, so that, for example, it would never happen that two policemen attempt to raise or lower a draw bridge at the same time. P should also regulate the interaction between the team members and their leader, providing the leader with a degree of control over the behavior of the team members, and ensuring that the leader gets from each member the information it needs to manage the team. Moreover P must facilitate proper handling of various exceptions, such as the disappearance of the team leader, which would require the employment of a careful leader election procedure.

Ensuring that such distributed agents operate properly is difficult as such a protocol cannot, practically, be hard-wired into the communication devices in the police cars, because a single police car may be required to participate in various missions, subject to different protocols. We need a far more flexible technique for establishing a given coordination protocol over ad hoc multi-agent systems. In this paper we

have explicitly modeled the types of control that are needed in a wireless ad hoc network and, using the police example as motivation, we have devised a flexible technique for coordinating an ad hoc collection of agents. Our approach leverages Law Governed Interaction (LGI) [1], which was originally developed for regulating transactions over the Internet. Under our version of LGI, each wireless device would have a built-in generic *controller* that can interpret an arbitrary interaction protocol, written in a special protocol-language. With such a generic controller, addressing the challenges of the police-mission becomes easy as all we must do is: (a) write our protocol P in a language recognized by the controllers; and (b) load this protocol into all the controllers built into the team member cars, and into the various actuators on the road.

The rest of this paper is organized as follows. Section II describes related work. Section III presents an overview of LGI, which provides the mechanism for implementing such applications. In Section IV, we provide a motivating example of an ad hoc team of traffic police officers whose mission is to monitor the traffic-related situation along with its implementation using the concept of LGI. The architecture of our proposed solution is described in Section V. We report various performance tests of our implementation conducted on the ORBIT testbed [2] in Section VI. Finally, we conclude and provide directions for future work in Section VII.

II. Related Work

DRAMA [3] is a policy-based network management system for mobile ad-hoc networks. The policies are represented by event-condition-action rules concerned with configuration, monitoring, and reporting of management events in a network. DRAMA policies are enforced in a distributed manner by Policy Agents that are co-located with the managed network elements. Policy operations–such as enabling, disabling, or introducing new policies–are propagated between Policy Agents in a peer-to-peer manner. DRAMA, however, is not concerned much with controlling the communication between managed network elements, and has only a rudimentary and stateless access control capability.

Xu et al. proposed SATEM (Service-Aware Trusted Execution Monitor) [4], which is a partial realization of LGI at a lower layer running on a TPM. Notably, this implementation did not include statefulness. However, this work suggests the validity of our approach as they have shown the feasibility of implementing such enforcements using trusted platforms. Further, the authors have enhanced this work to provide a distributed mechanism that allows trusted nodes to create protected networks in [5]. Only nodes that can demonstrate their trustworthiness by proving their ability to enforce policies are allowed to become members of the protected MANET. This avoids attacks from untrusted nodes as well as prevents attacks from member nodes due to enforcement of network policy.

In [6], Viterbo et al. have proposed a system that applies regulatory mechanisms to coordinate the interaction among applications in ubiquitous computing. A Domain Regulation Service regulates the interaction between client and server applications based on an explicit set of rules and contextual data. This service in turn acts as a centralized entity and may become a bottleneck besides being a single point of failure. Their system does not support stateful policies.

Rei [7] is a policy language for pervasive computing applications that includes constructs for rights, prohibitions, obligations and dispensations (deferred obligations). Rei includes a representation of speech acts (delegation, revocation, request and cancel) that are used to decentralize control and support dynamic modification of policies. Rei is a flexible and an expressive policy language that allows various kinds of policies (such as security, privacy, management, conversation etc.) to be specified. However, to our knowledge, Rei does not provide any support for handling communication faults and stateful policies.

III. An Overview of LGI

We have used the LGI paradigm to define the regulation policies as *laws*. The most salient aspects of LGI laws are their *strictly local formulation* and the *decentralized nature* of their enforcement. In this section, we provide an overview of the LGI mechanism. The implementation of LGI for ad hoc networks is similar to the LGI implementation for the Internet by the *Moses toolkit* [8] with some modifications as described in Section V.

LGI is a mode of interaction that allows an *open* group of distributed heterogeneous *agents* to interact with each other with confidence that the explicitly specified policies, called the *law* of the open group, is complied with by everyone in the group [1]. The messages exchanged under a given law $\mathcal{L}$ are called $\mathcal{L}$-messages, and the group of agents interacting via $\mathcal{L}$-messages is called a *community* $\mathcal{C}$, or more specifically, an $\mathcal{L}$-community $\mathcal{C}_{\mathcal{L}}$.

The concept of "open group" has the following semantic: (a) the membership of this group can be very large, and can change *dynamically*; and (b) the members of a given community can be *heterogeneous*. LGI does not assume any knowledge about the structure and behavior of the members of a given $\mathcal{L}$-community. All such members are treated as black boxes by LGI. LGI only deals with the interaction between these agents. Members of a community are not prohibited from non-LGI communication across the Internet, or from participation in other LGI-communities.

For each agent x in a given $\mathcal{L}$-community, LGI maintains the *control state* $\mathcal{CS}_x$ of this agent. These control states, which can change dynamically subject to law $\mathcal{L}$, enable the law to make distinctions between agents, and to be sensitive to dynamic changes in their states. The semantic of the control state for a given community is defined by its law, and could represent such things as the role of an agent in this community, its identity, its privileges, or reputation, etc. The

Published in the UBICOMM Conference October 2010

$\mathcal{CS}_x$ is viewed as a collection of objects called *Terms*. For instance, under the $\mathcal{L}$ law (to be introduced in Section IV), a term with the value *role(officer)* in the control state of an agent denotes that the agent has been authenticated to be a genuine officer.

In the rest of this section we discuss the concept of law, its local nature, and describe the decentralized mechanism for law enforcement. The interested reader is referred to [1] for more detail regarding LGI.

A. The Concept of Law and Its Enforcement

The law of a community $\mathcal{C}$ is defined over certain types of events occurring at members of $\mathcal{C}$, mandating the effect that any such event should have; this mandate is called the *ruling* of the law for a given event. The events subject to laws, called *regulated events*, include (among others): the *sending* and the *arrival* of an $\mathcal{L}$-message; the *coming due* of an *obligation* previously imposed on a given agent; and the submission of a *digital certificate*. The operations that can be included in the ruling of the law for a given regulated event are called *primitive operations*. They include: operations on the control state of the agent where the event occurred (called, the *home agent*); operations on messages, such as *forward* and *deliver*; and the imposition of an obligation on the home agent. The ruling of the law is not limited to accepting or rejecting a message, but can mandate any number of operations, like the modifications of existing messages, and the initiation of new messages and of new events, thus providing the laws with a strong degree of flexibility. More concretely, LGI laws are formulated using an *event-condition-action* pattern. In this paper we will depict a law using the following pseudo-code notation:

upon <event> **if** <condition>
do <action>

where the <*event*> represents one of the regulated events, the <*condition*> is a general expression formulated on the event and control state, and the <*action*> is one or more operations mandated by the law. This definition of the law is abstract in that it is independent of the language used for specifying laws. Concretely, we used Java but note that Prolog is also a viable language for writing the laws. However, despite the pragmatic importance of a particular language being used for specifying laws, the semantics of LGI is basically independent of that language.

Thus, a law $\mathcal{L}$ can regulate the exchange of messages between members of an $\mathcal{L}$-community, based on the control state of the participants; and it can mandate various side effects of the message exchange, such as modification of the control states of the sender and/or receiver of a message, and emission of extra messages.

1) The Local Nature of Laws: Although the law $\mathcal{L}$ of a community $\mathcal{C}$ is *global* in that it governs the interaction between *all* members of $\mathcal{C}$, it is enforced locally at each member of $\mathcal{C}$, by the following properties of LGI laws:

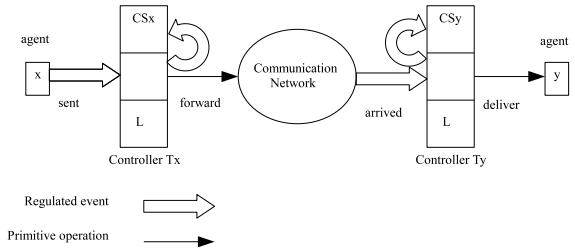

Figure 1. LGI framework achieves regulation of agents through controllers.

- $\mathcal{L}$ only regulates local events at individual agents.
- The ruling of $\mathcal{L}$ for an event e at agent x depends only on event e and the local control state $\mathcal{CS}_x$ of x.

The ruling of $\mathcal{L}$ at x can mandate only local operations to be carried out at x, such as an update of $\mathcal{CS}_x$, the forwarding of a message from x to some other agent y, and the imposition of an obligation on x. The fact that the *same law* is enforced at all agents of a community gives LGI its necessary global scope, establishing a *common* set of ground rules for the members of $\mathcal{C}$ and providing them with the ability to trust each other, in spite of the heterogeneity of the community. Furthermore, the locality of law enforcement enables LGI to scale with the size of the community.

2) Distributed Law-Enforcement: The law $\mathcal{L}$ of community $\mathcal{C}_\mathcal{L}$ is enforced by a set of trusted agents, called controllers that mediate the exchange of $\mathcal{L}$-messages between members of $\mathcal{C}_\mathcal{L}$. Every member x of $\mathcal{C}$ has a controller $\mathcal{T}_x$ assigned to it ($\mathcal{T}$ here stands for trusted agent), which maintains the control state $\mathcal{CS}_x$ of its client x. All these controllers, which are logically placed between the members of $\mathcal{C}$ and the communication medium as illustrated in Figure 1 carry the same law $\mathcal{L}$. Every exchange between a pair of agents x and y is mediated by their controllers $\mathcal{T}_x$ and $\mathcal{T}_y$, so that this enforcement is inherently decentralized.

3) The basis of trust between members of a community: For members of an $\mathcal{L}$-community to trust its interlocutors to observe the *same law*, one needs the following assurances: (a) Messages are securely transmitted over the network; (b) The exchange of $\mathcal{L}$-messages is mediated by controllers interpreting the same law $\mathcal{L}$; and (c) All these controllers are correctly implemented. If these conditions are satisfied, then it follows that if agent y receives an $\mathcal{L}$-message from agent x, this message must have been sent as an $\mathcal{L}$-message; in other words, that $\mathcal{L}$-messages cannot be forged.

We assume messages transmitted over the network are secured through proper cryptographic authentication and integrity mechanisms. To ensure that a message forwarded by a controller $\mathcal{T}_x$ under law $\mathcal{L}$ would be handled by another controller $\mathcal{T}_y$ operating under the same law, $\mathcal{T}_x$ appends the one-way hash [9][10] H of law $\mathcal{L}$ to the message it forwards to $\mathcal{T}_y$. $\mathcal{T}_y$ would accept this as a valid $\mathcal{L}$-message if and only if H is identical to the hash of its own law. As to the correctness of controllers, we assume here that every $\mathcal{L}$-community is willing to trust the controllers certified by a given $\mathcal{CA}$, which is specified by the law $\mathcal{L}$. In addition,

every pair of interacting controllers must first authenticate each other by means of certificates signed by this $\mathcal{CA}$.

B. Additional Features of LGI

Some of the further features of LGI are now discussed. For additional information, the reader is referred to [11].

1) The Treatment of Certificates: Certificates may be required by a given law $\mathcal{L}$ to certify the controllers used to interpret this law. Certificates may also be submitted by an actor x to its controller $\mathcal{T}_x$. The effect of such certificates is subject to the law in question. Typically, such submitted certificates are used to authenticate the identity of the actor, or the role it plays in the environment in which the community in question operates.

LGI currently supports the SPKI/SDSI model [12] for certificates. Under LGI, a certificate is a four-tuple *(issuer, subject, attributes, signature)*, where *issuer* is the public-key of the $\mathcal{CA}$ that issued and signed this certificate, *subject* is the public-key of the principal that is the subject of this certificate, *attributes* is what is being certified about the *subject*, and the *signature* is the digital signature of this certificate by the *issuer*. The *attributes* field is essentially a list of (attribute, value) pairs. For example, the attributes of a certificate might be the list [name(Joe), role(officer)], asserting that the name of the subject in question is Joe and its role in this community is that of a officer.

2) Enforced Obligation: Informally speaking, an *obligation* under LGI is a kind of *motive force*. Once an obligation is imposed on an agent (generally, as part of the ruling of the law for some event), it ensures that a certain action (called a *sanction*) is carried out at this agent, at a specified time in the future, when the obligation is said to come due, and provided that certain conditions on the control state of the agent are satisfied at that time. Note that a pending obligation incurred by agent x can be *repealed* before its due time. The circumstances under which an agent may incur an obligation, the treatment of pending obligations, and the nature of the sanctions, are all governed by the law of the community.

3) The Treatment of Exceptions: Primitive operations that initiate messages, like *deliver* and *forward*, may end up not being able to fulfill their intended function. For example, the destination agent of a *forward* operation may fail by the time the forwarded message arrives at it. Such failures can be detected and handled via a regulated event called *an exception*, which is triggered when a primitive operation that initiates communication cannot be completed successfully. It is up to the law to prescribe what should be done to recover from such an exception. The syntax of an exception event is *exception(op, diagnostic)* where *op* is the primitive operation that could not be completed, and *diagnostic* is a string describing the nature of the failure. The home of the exception event is the home of the event that attempted to carry out the failed operation. For instance, if a message m, forwarded by an agent x to an agent y operating under law $\mathcal{L}$ cannot reach its destination, then an event *exception(forward(x,m,[y,$\mathcal{L}$]),'destination not responding')* would be triggered at x. Commonly, exceptions are triggered by the *forward* and *deliver* primitive operation, as well as other communication primitives.

IV. An Ad Hoc Police Team Mission

This case study involves the police team mission introduced in Section I. We now elaborate on the structure and operations of this team. The team, whose purpose is to manage traffic in a given region by operating on a set of traffic-related devices (sensors and actuators), involves the following participants: (a) the *officers* who query the various sensors on the road, and operates on the various actuators; (b) a *leader* who monitors the activities of the officers participating in the mission and grants them access control rights to various devices; (c) a *supervisor* who maintains the information about the current leader, and has the ability to appoint a new leader, if the current leader fails, and to notify all team members of the new leader; (d) an *auditor* who maintains a log of messages sent to the various devices and provides this information to the leader whenever it requests for it.

We classify the messages sent by the officer into the following: *control messages* sent to the various devices (such as a command to raise a draw bridge, or to change the color of a traffic light), and *conversation messages* sent to any team member.

The members of the team and the sensors and actuators managed by them—collectively referred to as *agents*—operate according to protocol P specified informally below:

1) *Authentication of Identity*: For an agent to participate in the mission it must authenticate itself and its role via a certificate issued by a specific certification authority (CA) known as admin.
2) *Steering of the team*: The team of officers can be regulated by the leader in the following way: (a) the leader can grant and withdraw permission to an officer to access a particular device; (b) the leader has the right to query any required information from any of the officers taking part in the mission; (c) the leader has the power to stop the officer from taking part in the mission; and (d) the leader can assign a conversation message budget to any officer, which would restrict the number of arbitrary messages circulating in the network, thus reducing the possibility of congestion.
3) *Control Messages*: An officer is allowed to issue control messages to a device to which it has access rights. The copy of such a control message must be sent to the auditor.
4) *Fault tolerance*: The supervisor has the power to appoint a new leader, if the current leader fails to send him heart-beat messages.
5) *Control State Content and Conversation Messages*: Any member taking part in the mission should be able to access its control state to know the various terms

stored in it. The leader, auditor and supervisor are able to send arbitrary messages to each other, and to the plain officers. Each officer can also send an arbitrary conversation messages, to any team member, provided that it has sufficient budget for such messages.

A. Realization of Policy P via an LGI Law $\mathcal{L}$

To ensure that our mission team operates as required, we will have all team members, and all traffic related devices to be managed, operate under an LGI law $\mathcal{L}$ that realizes the policy P described above. They are, accordingly, called $\mathcal{L}$-agents, or simply agents. (Note that such agents can recognize each other as bona fide $\mathcal{L}$-agents.)

Before we get to law $\mathcal{L}$ itself we make the following preliminary comments: First, terms in each agent's control state are used to represent the role played by this agent. In particular, the control state of the current leader should contain a term, *role(leader)*. Likewise, the presence of term *budget(B)* in the control state of an officer x means that x has a budget of amount *B* and is entitled to send *B* conversation messages. An acting leader is forced to announce its identity periodically to the supervisor after every T_{report} seconds. If the current leader does not report to the supervisor within a predetermined time T_{fail} seconds, then it is assumed that the current leader has failed and the supervisor appoints a new officer as the leader of the mission.

Law $\mathcal{L}$ itself consists of two parts namely the *preamble* and the *body*. The preamble of $\mathcal{L}$ consists of the following clauses. First, there is the law clause that identifies the name of this law and the $\mathcal{CA}$ *admin* whose public key is used for the authentication of the controllers that mediate the messages of this system. Second, there is an *authority clause* that identifies the $\mathcal{CA}$ *admin* (represented by the keyed hash of its public key) for certifying the roles played by the different actors in this community. Third, the *initialCS clause* defines the initial control state of all actors in this community—it is empty in this case. Finally, the two *alias clauses* provide shorthand for the identifier (id) of supervisor and auditor respectively.

The law is now presented as a list of fragments along with their pseudo code, and explained in English.

1) Authentication of Identity: The fragment of the $\mathcal{L}$ law in Figure 2 shows how the authentication of identity takes place. When a participant engages in the system, it does so by sending an *adoption* message to its LGI controller, a message that can carry its certificate. When the message arrives at the controller, it invokes an *adopted event*. If an actor submits a certificate, then the controller verifies it with the public key of the $\mathcal{CA}$ *admin* and challenges it with the private key of the subject as shown by rule $\mathcal{R}1$. If the subject is not the one who presented the certificate, or if the issuer is not the $\mathcal{CA}$ *admin*, then no role and no identity is assigned to the actor and it is forced to quit. If the attributes of the certificate contain the role of *supervisor, leader, device or officer*, then this role of the

```
Preamble:
        law(name(L),authority(admin)).
        authority(admin,keyHashOfAdmin).
        initialCS( ).
        alias(supervisor,"supervisor@192.168.10.1").
        alias(auditor,"auditor@192.168.10.2").

 R1)  upon adopted(Self,Issuer,Subject,Attributes)
            if(Subject!=Self or Issuer!=Admin)
               do Quit
            if(Attributes.role = supervisor)
               do Add(role(supervisor))
               do ImposeObligation(failure,600)
            if(Attributes.role = leader)
               do Add(role(leader))
               do Forward(Self,currentLeader,supervisor)
               do ImposeObligation(report,300)
            if(Attributes.role = auditor or device)
               do Add(role(Attributes.role))
            if(Attributes.role = officer)
               do Add(role(officer))
               do Add(budget(10))

 R2)  upon adopted(Args)
            do Quit
```

Figure 2. Authentication of Identity: Fragment of the $\mathcal{L}$ Law

actor is extracted from the attributes and saved in the control state maintained by the controller on behalf of the actor. The leader reports its identity to the supervisor and an *obligation* is imposed on the controller of the leader to come due after T_{report} period (for example, we use a reporting time of 300 seconds). Also, an initial budget of $B_{initial}$ messages (say 10 messages) is provided to all the officers for initial arbitrary communication. The controller of the supervisor keeps a check on the status of the current leader through the obligation *failure*, which comes due after every T_{fail} period (assumed to be 600 seconds) since the last time a successful reporting was made. On the other hand, if no certificate is provided in the adoption message, then the actor will be automatically forced to *quit* as shown by rule $\mathcal{R}2$.

2) Steering of the Team: Figure 3 show the fragment of the $\mathcal{L}$ law that handles this process. Policy $\mathcal{P}$ allows the leader to *steer* the messaging activity of all the officers by suitably modifying their budgets. This provision is implemented by rules $\mathcal{R}3$ to $\mathcal{R}6$, which allows the leader to send a message of the form *incrementBudget(Amount)* or *decrementBudget(Amount)* to an officer, resulting in an increase or reduction in their corresponding budget by the specified amount. The leader can grant any participating officer the right to access a device pertaining to the mission (such as bridge, traffic lights, cameras, etc.) through rule $\mathcal{R}7$. According to rule $\mathcal{R}8$, when the controller of an officer obtains the right to access a device, the corresponding *permission* is added to its control state and the message is then delivered to the officer. Similarly, the leader can cancel any officer's right to access a particular device, which results in the removal of the corresponding permission term from the control state of the officer (rules $\mathcal{R}9$ and $\mathcal{R}10$).

The leader is authorized to request any desired information from an officer by sending a *requestInfo* message as shown by rule $\mathcal{R}11$. By rule $\mathcal{R}12$, an officer is obligated to

```
R3) upon sent(C,incrementBudget(Amount),X)
      if (CS has role(leader))
        do Forward

R4) upon arrived(C,incrementBudget(Amount),X)
      if(CS has role(officer))
        do Replace(budget(B),budget(B+Amount))
        do Deliver

R5) upon sent(C,decrementBudget(Amount),X)
      if (CS has role(leader))
        do Forward

R6) upon arrived(C,decrementBudget(Amount),X)
      if(CS has role(officer))
        if (B>Amount)
          do Replace(budget(B),budget(B-Amount))
        else
          do Replace(budget(B),budget(0))
        do Deliver

R7) upon sent(C,grantAccess(Device),X)
      if (CS has role(leader))
        do Forward

R8) upon arrived(C,grantAccess(Device),X)
      if (CS has role(officer))
        do Add(permission(Device))
        do Deliver

R9) upon sent(C,repealAccess(Device),X)
      if (CS has role(leader))
        do Forward

R10) upon arrived(C,repealAccess(Device),X)
      if (CS has role(officer))
        do Remove(permission(Device))
        do Deliver

R11) upon sent(C,requestInfo(I),X)
      if (CS has role(leader))
        do Forward

R12) upon arrived(C,requestInfo(I),X)
      do ImposeObligation(requestInfo(C),180)
      do Deliver

R13) upon sent(X,replyInfo(I),C)
      if (CS has obligation(requestInfo(C)))
        do RepealObligation(requestInfo(C))
        do Forward(X,reply(replyInfo(I),
          controlState(Terms)),C)
      else
        do Deliver("Info_not_requested_by_this_
          destination")

R14) upon arrived(X,reply(replyInfo(I),controlState(
      Terms)),C)
      do Deliver

R15) upon obligationDue(requestInfo(C))
      do Forward (Self,notResponding(
         controlState(Terms)),C)

R16) upon arrived(X,notResponding(controlState(Terms)),
      C)
      do Deliver

R17) upon sent(C,stop,X)
      if (CS has role(leader))
        do Forward

R18) upon arrived(C,stop,X)
      do Deliver
      do Quit
```

Figure 3. Steering of the team: Fragment of the $\mathcal{L}$ law

respond to the request made by the leader within $T_{request}$ period (say 180 seconds). If the officer responds to the query posed by the leader within $T_{request}$ period, then the obligation is repealed, the control state terms are appended to the reply and forwarded to leader (by rule $\mathcal{R}13$). Such a reply message is simply delivered to the leader as per rule $\mathcal{R}14$. According to rules $\mathcal{R}15$ and $\mathcal{R}16$, if an officer does not respond to the obligation within $T_{request}$ period, then a *notResponding* message containing the control state terms is sent to the leader. The actions to be taken by the leader in such a circumstance are left to the discretion of the law of the mission at hand. Our law simply provides the ability to inform the leader of such a non responsive officer. The leader can dismiss any officer from taking part in the operations of the mission by issuing a *stop* message via rule $\mathcal{R}17$. By rule $\mathcal{R}18$, when a *stop* message arrives at the controller of the officer, the message is delivered to the officer and it is forced to quit.

```
R19) upon sent(X,operation(Parameters),D)
       if(CS has role(officer))
        if(CS has permission(D))
          do Forward
          do Forward(X,message(X,operation(
             Parameters),D),auditor)
        else
          do Deliver("do_not_have_permission_to_
             access_this_device")

R20) upon arrived(X,operation(Parameters),D)
       do Deliver

R21) upon arrived(X,message(X,operation(Parameters),D),
     auditor)
       do Deliver

R22) upon sent(C,query(Device),auditor)
       if(CS has role(leader))
         do Forward

R23) upon arrived(C,query(Device),auditor)
       do Deliver

R24) upon sent(auditor,queryResponse(R),C)
       do Forward

R25) upon arrived(auditor,queryResponse(R),C)
       do Deliver
```

Figure 4. Control Messages: Fragment of the $\mathcal{L}$ law

3) Control Messages: The monitoring function is carried out via the fragment of the $\mathcal{L}$ law shown in Figure 4. An officer can issue a *control* message (such as *operation(bridge(raise,speed))*) to operate on one of its accessible device, as shown by rule $\mathcal{R}19$. The necessary action to be carried out in response to this message is left up to the destination device. Our law simply provides the ability to deliver such a message to the device through rule $\mathcal{R}20$. According to rule $\mathcal{R}21$, a copy of such a control message is delivered to the auditor fulfilling the monitoring requirement. The leader can query the status of any device by sending a request to the auditor via rule $\mathcal{R}22$. By rule $\mathcal{R}23$, such a query message is simply delivered to the auditor. The auditor's response to the query is delivered to the leader

through rules $\mathcal{R}24$ and $\mathcal{R}25$.

```
R26) upon obligationDue(report)
          if (CS has role(leader))
             do Forward(Self,currentLeader,supervisor)
             do ImposeObligation(report,300)

R27) upon arrived(C,currentLeader,supervisor)
          do RepealObligation(failure)
          if(CS has leader(A))
             do Replace(leader(A),leader(C))
          else
             do Add(leader(C))
          do ImposeObligation(failure,600)

R28) upon obligationDue(failure)
          do Deliver(Self,appoint,Self)
          do Add(readyToAppoint)
          do Remove(leader(A))
          do ImposeObligation(failure,600)

R29) upon sent(supervisor,appoint,N)
          if (CS has readyToAppoint)
             do Remove(readyToAppoint)
             do Add(leader(N))
             do Forward

R30) upon arrived(supervisor,appoint,N)
          if(CS has role(officer))
             do Remove(role(officer))
             do Remove(budget(B))
             do Add(role(leader))
             do ImposeObligation(report,300)
             do Deliver

R31) upon exception(supervisor,appoint,N)
          do Remove(leader(N))
          do Add(readyToAppoint)
          do Deliver(Self,exception(appoint),Self)
```

Figure 5. Fault Tolerance: Fragment of the $\mathcal{L}$ law

4) Fault Tolerance: Figure 5 introduces the fault tolerance fragment of the law $\mathcal{L}$, which would allow our police team to recover from an unpredictable failure of its leader. We will consider the failure of the leader along with its controller to be of a *fail-stop* kind. We also assume that the supervisor and auditor do not fail. A broader perspective on such treatment of failures as part of self-healing under LGI can be obtained by referring to [13].

We adopt the concept of a *guardian* originally proposed by Tripathy et al. [14] to handle the failure of the leader. We assume that the supervisor acts as a guardian for the mission and is responsible for appointing an officer to the post of the leader whenever the current leader fails.

The leader is forced to report its status to the supervisor after every T_{report} period via an *obligation* as shown by rule $\mathcal{R}26$. The supervisor suitably updates the current leader information stored in its control state on receiving such an update (by rule $\mathcal{R}27$). In the absence of such a timely reporting, the obligation *failure* comes due at the controller of the supervisor as shown by rule $\mathcal{R}28$. The supervisor is then asked to appoint a new leader. This state of supervisor is characterized by the presence of the term *readyToAppoint*. According to rule $\mathcal{R}29$, when the supervisor sends a message to appoint some officer as the new leader, a new leader term for this appointee is inserted in its control state and the message is sent to the appointee. By rule $\mathcal{R}30$, when the forwarded *appoint* message arrives at an officer, it becomes the new leader. If an *exception* occurs while the supervisor is trying to appoint a new officer to the position of leader, then the corresponding leader term is once again removed from the control state of the supervisor and the term *readyToAppoint* is added back to the control state. Then, the supervisor is prompted again to appoint a new leader as shown by rule $\mathcal{R}31$.

```
R32) upon sent(X,getCS,Y)
          do DiscloseCS(all)

R33) upon sent(X,AnyOtherMessage,Y)
          if(CS has role(leader or supervisor or
            auditor))
           do Forward
          if (CS has role(officer) and budget(B))
           if(B > 0)
             do Replace(budget(B),budget(B-1))
             do Forward
           else
             do Deliver("Message_blocked_due_to_
                insufficient_budget")

R34) upon arrived(X,AnyOtherMessage,Y)
          do Deliver

R35) upon exception(E,D)
          do Deliver(Self,exception(E,D),Self)
```

Figure 6. Control State Content and Conversation Messages: Fragment of the $\mathcal{L}$ law

5) Control State Content and Conversation Messages: The participants of the system can check the terms stored in their control state and exchange various other messages via the rules given in Figure 6. Any participant can check the terms stored in its control state by sending a *getCS* message to its controller (by rule $\mathcal{R}32$). According to rule $\mathcal{R}33$, any participant (except the devices) can send any conversation message to another participant in the community. An officer can send a conversation message only if it has sufficient budget; the cost of which will be deducted from its budget. On receiving such a conversation message, the controller of the recipient simply delivers it to the actor as per rule $\mathcal{R}34$. If any other *exception* is raised, then the corresponding message and the reason for its failure is delivered to the sender by rule $\mathcal{R}35$.

B. Discussion

The law can be extended to achieve coordination in such a way that it would never happen that two officers issue contradictory control messages at the same time (for example, two officers should not be able to raise and lower the bridge at the same time) without knowing about each other. It is also possible to impose a restriction of changing the traffic light in front of a bridge to red before lowering the bridge. Further, it may be desired for certain missions to have the various devices (such as bridges, cameras, traffic lights etc.) work under their own law so that they can be operated independently by the officers (irrespective of the

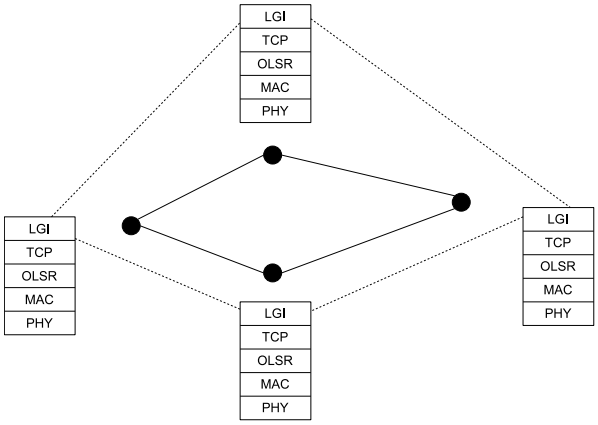

Figure 7. Our system runs LGI on a wireless ad hoc network protocol stack using TCP and OLSR.

law under which they are working). LGI supports this feature by allowing different policies to cross-interact, in a regulated manner, giving rise to *interoperability* between different LGI-communities. If a given community (like our police team) is required to operate in a context that imposes some global constraints on all wireless communication taking place in it, then the LGI policies can be organized in a *conformance hierarchy* [15]. We do not address these issues due to lack of space. Finally, a failure of the supervisor or the auditor can be addressed by replicating them. If we assume that the failure of the auditor or supervisor is very rare, then such replication would not adversely affect the scalability of the system.

V. LGI Architecture for Manet Applications

An example ad hoc network with LGI is shown in Figure 7. The LGI application runs on top of TCP. TCP is used instead of UDP for reliable delivery. The routing protocol used in the current implementation is Optimum Link State Routing (OLSR) [16]. We have chosen a proactive routing protocol such as OLSR over reactive routing protocol like Ad hoc on demand Distance Vector (AODV) routing [17] in order to minimize the message transmission delay. It should be noted that LGI as a mechanism is independent of the network routing protocol. We assume that each node has a trusted implementation of the LGI controller. Such an implementation requires the use of Trusted Platform Module (TPM) as specified by the Trusted Computing Group (TCG) [18], which we intend to do in future.

An actor adopts its controller with a particular law to become a member of the corresponding community. Laws can be made freely available on a server so that a user can download the appropriate law whenever it has internet connection. Another option is that the host can obtain the required law from a *certification authority* ($\mathcal{CA}$) when it applies for a certificate. The actor then communicates with the other participants in the community via its controller. We assume that mechanisms are in place to ensure that the nodes participating in the network can form a protected community so that outsider attacks can be prevented. This can be achieved using a secured routing protocol [19][20]. We disregard physical layer attacks as such consideration is beyond the scope of this paper.

To deploy LGI in an ad hoc network environment and test the implementation on the ORBIT grid [21], it was necessary to introduce subtle changes from the Moses Toolkit that operates over Internet via TCP/IP. The *preamble clause* can no longer specify the URL of the law in the *portal clause* in this implementation of LGI for MANET. Similarly, the *addPortal primitive* can no longer specify the URL of the law. Certifying Authorities can no longer be included in the law by specifying the URL of their public key. Human actors, controller service pool, and controller manager are no longer supported in this implementation. Features of the Moses toolkit that employed URL are no longer supported in the current implementation of LGI for ad hoc networks.

VI. Performance of LGI in a Wireless Ad Hoc Environment

The first part of this section introduces a model for the relative overhead of LGI-regulated communication based on a performance model published in [1]. The second part of this section reports on the *event evaluation time*, the *actor to controller communication time* and computes the *relative overhead* for messages by evaluating the *unregulated message transfer time* and the *LGI-regulated message transfer time* for a 5-hop topology. The results reported here are for laws written in Java rounded off to the nearest integer wherever appropriate. The experiments have been conducted on the ORBIT grid with nodes having a processor speed of 1 GHz on a Linux 2.6.12 platform. We have used the OLSRD 0.4.10 [22] implementation and 802.11a radio for communication. Further, a 5-hop topology is created using ORBIT tools to estimate the overhead due to LGI.

A. A Model for the Relative Overhead of LGI

Consider an LGI message m sent by an actor x to a destination actor y. This message is mediated by the controller of x ($\mathcal{C}_x$), which sends the message to controller of y ($\mathcal{C}_y$). (We denote controllers by the letter C here instead of the letter T used before, in order to avoid confusing it with the notation for time). Therefore, this message is converted to three consecutive messages: (1) from x to $\mathcal{C}_x$, (2) from $\mathcal{C}_x$ to $\mathcal{C}_y$, and (3) from $\mathcal{C}_y$ to y. The overhead $o_{x,y}$ due to the extra messages and the law-evaluations involved, is given by the following formula:

$$o_{x,y} = t_{com}^{x,C_x} + t_{eval}^{sent} + t_{com}^{C_x,C_y} + t_{eval}^{arrived} + t_{com}^{C_y,y} - t_{com}^{x,y} \quad (1)$$

where t_{eval}^{e} is the time it takes a controller to compute and carry out the ruling for the event e, and $t_{com}^{a,b}$ is the communication time from a to b. The relative overhead

$ro_{x,y}$ of an LGI message from x to y (as compared to the unregulated transmission of such a message) is defined as:

$$ro_{x,y} = o_{x,y}/t_{com}^{x,y}. \quad (2)$$

B. Measurements

1) Event evaluation time (t_{eval}): This experiment measures the time required by the controller to evaluate an event under LGI written as Java law. In this experiment, an actor adopts its controller with a sample law and then sends a message to its controller. The controller on evaluation of this sample law forwards the message to the same actor generating an *arrived* event, which in turn loops 100,000 times before getting delivered to the actor. The event evaluation time was averaged over 10 experiments, and gave:

Avg t_{eval} = 118 μs, Standard deviation = 2 μs.

This event evaluation time includes the time needed for local communication from actor to controller and back from controller to actor (within the same host), which is ignored as the experiment consists of 100,000 loops of actual event evaluations. Further, we have ignored the dependency on the events corresponding to different rulings of any LGI law. The variance of the event evaluation time for the different rules of the police team law introduced earlier is negligible.

2) Actor to Controller Communication time (t_{local}): This experiment finds the time it takes for a message sent by an actor to reach its own controller. A message is sent by the actor to its own controller, which on evaluation of the law, simply delivers the message back to the actor . This process is executed 100,000 times. The time obtained is averaged over 10 such experiment to get an accurate value. The average t_{delay} is 500 μs with a standard deviation of 4 μs. The delay measured in this experiment consists of the time taken for the actor to send a message to its controller (t_{local}) within the same host, event evaluation time (t_{eval}) corresponding to the ruling of the law at the controller and the time it takes for the message to be delivered back to the actor. Thus,

Avg t_{local} = (500 - 118)/2 = 191 μs.

On average, when an actor communicates with its controller on the same host, it takes t_{local}+ t_{eval} = 191+118 = 309 μs to receive (or to send a message) and to handle the associated event. Thus, the average throughput rate for the controller is 3236 events per second.

We now apply the model in [1] to evaluate the relative overhead of LGI-communication for a 5-hop topology.

3) Unregulated Message Transfer time ($t_{unregulated}$): In this experiment, we calculate the average time it takes for two hosts (separated by a 5-hop topology) to communicate with each other, i.e., the time required to transfer an unregulated message. A simple client is run on the sending node and a server application is run on destination node (both written in Java). The client program sends a message to the server program. This process is repeated 10,000 times. The resulting unregulated message transfer time is averaged over 10 such experiments to get an accurate value.

Avg $t_{unregulated}$ = 1.97 ms,
Standard deviation = 0.0015 ms

The unregulated message transfer time depends on many factors, such as message length, communication protocol, and distance between nodes. The distance between the nodes could not be varied much as these tests were run on the ORBIT indoor grid where nodes are spread over a distance of 80 ft by 70 ft [21]. In general, the delay caused by the message length and the distance between the nodes is negligible and has been neglected in these calculations. This unregulated message transfer time measurement does, however, take into account the overhead caused by the routing protocol (OLSR in our case).

4) Regulated Message Transfer time ($t_{regulated}$): This experiment used the same 5-hop topology. The actors on the sending and receiving nodes adopt their respective controllers with a Java law that simply forwards any message that is being sent and delivers any message that is received. The actor on the sending node sends a message to the actor on the receiving node. The in between nodes of the 5-hop topology act as routers and forward the message to the destination.

Avg $t_{regulated}$ = 2.4 ms, Standard deviation = 0.1 ms

The regulated message transfer time consists of the event evaluation for the ruling of the current law and local communication delay at the two end nodes along with the message transmission delay. Thus, the relative overhead is

$ro_{x,y}$ = (2.4 - 1.97)/1.97 = 0.22.

This overhead is far from prohibitive for most applications.

VII. Conclusion and Future Work

We have introduced a model of interaction control for the *regulation of wireless communication* in ad hoc networks using LGI to regulate the dynamic behavior of the interacting wireless agents. The power of the proposed mechanism resides in its ability to handle *statefulness*, *obligations*, *exceptions* and *locality*. There are many practical applications of such a system (e.g., police personnel at a sports event, medical personnel at an accident scene, emergency responders to a natural disaster, secure electronic commerce [23], manageable and robust multi-agent systems [24], etc.), yet little prior work exists that addresses these scenarios. We have prototyped an example based on a team of police officers in an ad hoc mission to control traffic. We have considered the critical elements of management of such an ad hoc team to provide: a) the leader with the ability to steer its subordinates and b) monitor relevant operations of its subordinates; and finally c) to provide robustness of the

agent-community under certain unexpected adverse conditions, such as unpredictable failure of the leader itself. We have shown that the overhead added due to LGI would not adversely impact performance. We plan to extend this work by implementing our mechanism on a TPM and extending it to support hierarchy of laws for ad hoc scenarios.

REFERENCES

[1] N. H. Minsky and V. Ungureanu, "Law-governed interaction: a coordination and control mechanism for heterogeneous distributed systems," *ACM Trans. Softw. Eng. Methodol.*, vol. 9, no. 3, pp. 273–305, 2000.

[2] M. Ott, I. Seskar, R. Siraccusa, and M. Singh, "Orbit testbed software architecture: supporting experiments as a service," in *Testbeds and Research Infrastructures for the Development of Networks and Communities, 2005. Tridentcom 2005. First International Conference on*, 23-25 2005, pp. 136 – 145.

[3] R. Chadha, H. Cheng, Y.-H. Cheng, J. Chiang, A. Ghetie, G. Levin, and H. Tanna, "Policy-based Mobile Ad Hoc Network Management," in *Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on Policy for Distributed Systems and Networks*, Jun. 2004, pp. 35–44.

[4] G. Xu, C. Borcea, and L. Iftode, "Satem: Trusted service code execution across transactions," in *Reliable Distributed Systems, 2006. SRDS '06. 25th IEEE Symposium on*, 2-4 2006, pp. 321 –336.

[5] ——, "Trusted application-centric ad-hoc networks," in *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on*, 8-11 2007, pp. 1 –10.

[6] F. Viterbo, M. Endler, and J.-P. Briot, "Ubiquitous service regulation based on dynamic rules," in *Engineering of Complex Computer Systems, 2008. ICECCS 2008. 13th IEEE International Conference on*, march 2008, pp. 175 –182.

[7] L. Kagal, T. Finin, and A. Joshi, "A policy language for a pervasive computing environment," in *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*, 4-6 2003, pp. 63 – 74.

[8] C. Serban and N. H. Minsky, "The LGI website (includes the implementation of LGI, and its manual)," Rutgers University, Tech. Rep., Jun. 2005. [Online]. Available: http://www.moses.rutgers.edu_07.19.2010

[9] B. Schneier, *Applied Cryptography*. New York, NY, USA: John Wiley and Sons, 1996.

[10] R. Rivest, "The (MD5) message-digest algorithm," RFC 1321, MIT, Tech. Rep., Apr. 1992. [Online]. Available: http://www.ietf.org/rfc/rfc1321.txt_07.19.2010

[11] N. H. Minsky, "Law governed interaction (LGI): A distributed coordination and control mechanism (an introduction and a reference manual)," Rutgers University, Tech. Rep., Jun. 2005.

[12] C. M. Ellison, "The nature of a useable PKI," *Comput. Netw.*, vol. 31, no. 9, pp. 823–830, 1999.

[13] N. Minsky, "On conditions for self-healing in distributed software systems," in *Autonomic Computing Workshop, 2003*, 25 2003, pp. 86 – 92.

[14] A. Tripathi and R. Miller, "Exception handling in agent-oriented systems," in *Advances in exception handling techniques*. Springer-Verlag New York, Inc., 2001, pp. 128–146.

[15] X. Ao and N. H. Minsky, "Flexible regulation of distributed coalitions," in *LNCS 2808:the Proc. of the 8th European Symposium on Research in Computer Security (ESORICS) 2003*, Oct. 2003, pp. 39–60.

[16] T. Clausen and P. Jacquet, "Optimized link state routing (OLSR) protocol," RFC 3626, Oct. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3626.txt_07.19.2010

[17] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (AODV) routing," RFC 3561, Jul. 2003. [Online]. Available: http://tools.ietf.org/html/rfc3561_07.19.2010

[18] *TPM Specification*, Trusted Computing Group (TCG) Std. 1.2, Rev. 103, Jul. 2007. [Online]. Available: http://www.trustedcomputinggroup.org/resources/tpm_main_specification_07.19.2010

[19] F. Hong, L. Hong, and C. Fu, "Secure OLSR," in *Advanced Information Networking and Applications, 2005. AINA 2005. 19th International Conference on*, vol. 1, 28-30 2005, pp. 713 – 718 vol.1.

[20] Q. Li, Y.-C. Hu, M. Zhao, A. Perrig, J. Walker, and W. Trappe, "SEAR: a secure efficient ad hoc on demand routing protocol for wireless networks," in *ASIACCS '08: Proceedings of the 2008 ACM symposium on Information, computer and communications security*. ACM, 2008, pp. 201–204.

[21] ORBIT Radio Grid Testbed. WINLAB, Rutgers University. [Online]. Available: http://www.orbit-lab.org/wiki/Tutorial/Testbed_07.19.2010

[22] A. Tonnensen. OLSR daemon 0.4.10. [Online]. Available: http://www.olsr.org/_07.19.2010

[23] C. Serban, Y. Chen, W. Zhang, and N. H. Minsky, "The concept of decentralized and secure electronic marketplace," *Electronic Commerce Research*, vol. 8, no. 1-2, pp. 79–101, 2008.

[24] N. H. Minsky and T. Murata, "On manageability and robustness of open multi-agent systems," in *SELMAS*, 2003, pp. 189–206.