Resilient Distributed Policy Repository for Mobile Ad Hoc Networks

Maroun Touma, Seraphin Calo, Keith Grueneberg, Xiping Wang, David Wood
IBM Research
Hawthorne, NY, USA
touma,scalo,kgruen,xiping,dawood@us.ibm.com

Abstract—A Resilient Distributed Policy Repository (RDPR) supports the modification, dissemination and uniform execution of operational policies across complex network topologies spanning multiple organizations, geographies and domains. Operating in a highly dynamic environment, such as in Mobile Ad Hoc Networks (MANETs), network partitions form due to nodes disconnecting and reconnecting to the system. This paper describes a framework for a RDPR that uses self-management policies over a distributed data grid to enable localized adaptation of operational policies and resilience of on-going operations. In particular, we discuss the need for, approach to, and implementation of policy supersession to resolve potential policy conflicts caused by local policy adaptation in a partitioned network.

Keywords—distributed repository; partition; supersession; data grid; self-management;

I. INTRODUCTION

The dynamic nature of Mobile Ad Hoc Networks (MANETs), where devices may join or leave the system at any time, makes the problem of providing reliable access to shared information a very challenging one. In particular, management applications need access to control policies that capture the behaviors that have been prescribed for the system. They must be presented with a single unified view of the policy set that governs such issues as network membership, security, access control, etc. Further, in systems with multiple administrative domains, as in coalition environments, it cannot be assumed that there is only one source for changes to such data. There will be multiple sites at which policy editing can be done, requiring synchronization across the distributed system. Even within a particular domain, distributed editing may be supported for reasons of redundancy, convenience, or responsiveness to changing local conditions. As policies change, they must be distributed automatically to all nodes that require them.

In this paper, we propose the use of a data grid as the basis for a Resilient Distributed Policy Repository (RDPR) that is self-configuring and self-healing. We use policy mechanisms to provide the self-management characteristics of the repository. The nodes of the mobile data grid on which the RDPR is based are loosely coupled and are dynamically self-directed to their neighbors based on specified policies to programmatically adapt to operating conditions. Replication and synchronization of policies are managed by the grid. The number of nodes at which policies are replicated for resiliency is set by an administrative control policy that is also maintained in the repository. Policy data is maintained in memory for performance, and the RDPR appears as a single unified cache to the management applications.

In order to make the RDPR self-configuring, we formulate configuration constraints as policies, and a policy-based mechanism is used to check the actual configuration against the specified configuration rules. Reconfigurations can then be automatically triggered according to changing conditions in the operating environment. For self-healing, the RDPR can automatically react to certain error conditions based on the set of policies that were formulated to handle such error conditions.

To handle certain situations that can arise when the MANET becomes partitioned, we have also introduced the concept of policy supersession. Policies may need to change because of the lack of communications with other parts of the network, or a local commander may need to override certain policies because of changing conditions in a particular sector. Our policy specification tool allows policy authors to declare that a given policy is being superseded because of a certain event. Once a cancelling event occurs, the superseding policy is de-activated and the superseded policy is re-activated. This specifically handles the situation that occurs when the network comes back together after having been partitioned. Whatever policies have been changed because of the partitioning will be changed back to their original form.

There has been previous work on distributed policy repositories [8], but it concentrations on issues such as distributed access and data placement. Here, we specifically use grid technology and policy mechanisms to provide a distributed repository that is resilient and self-managing.

In Section 2 we describe our policy distribution framework, which follows the IETF/DMTF policy model. A coalition scenario is presented to motivate the requirements for this framework; and, a distributed policy repository is proposed for MANET environments. In Section 3, we further propose the use of an in-memory data grid as the basis for a distributed policy repository that can provide reliable access to operational policies in a distributed, dynamic system. Mechanisms for making this grid self-configuring and self-healing are presented; and, we describe the use of policy supersession as a means of dealing with changing policies as network partitions form and re-join. A reference implementation of a distributed policy repository is described in Section 4. Finally, we discuss the overall research work and present our conclusions in Section 5.
II. POLICY DISTRIBUTION FRAMEWORK IN A MANET

The policy framework defined by the Internet Engineering Task Force (IETF) and the Distributed Management Task Force (DMTF) [10] consists of four basic elements: a policy manager (PM), a policy repository (PR), a policy enforcement point (PEP), and a policy decision point (PDP). Over time, this model has been applied to many application domains [1] where policies are used for: provisioning of virtual private networks; deployment of distributed applications; and, management of all aspects of computing systems, including configuration, storage services, and security. The primary advantage of such an approach is that it simplifies the complex task of administering large, distributed systems by allowing management operations to be specified in terms of the objectives or goals that need to be met, rather than the detailed instructions that need to be executed. In this model, the role of the PM is to provide a means for users to input the different policies that are to be active in the system. These are stored in the PR. The PEP is the element of the system that can apply and execute the different policies; and, when it needs to make a decision, it calls out to a PDP which is responsible for obtaining the relevant policies from the PR, interpreting them and communicating them back to the PEP.

Let us consider a scenario where two or more organizations come together to form a coalition and agree on a common set of operational policies that will govern all joint operations. In this scenario, each organization will deploy its own set of assets in a MANET in such a manner that: all the assets deployed by the various organizations use the same operational policies; and, during the course of the mission, an organization can modify the operational policies as necessary to deal with an imminent threat or opportunity.

These requirements do in fact present difficult challenges in a MANET environment. Most notably, the dynamic nature of the network, where assets may join or leave at any time, prohibits the use of centralized mechanisms for the PR. A fully decentralized policy management and distribution solution must meet stringent requirements for:

1) Scalability: Using replication and cooperative caching techniques in a MANET, the PR must address the needs of the coalition in conducting multiple missions simultaneously using a large number of deployed assets spread over a large geography.

2) Self-configuring: Using constraints captured as management policies, each instance of the PR must be able to establish intermittent links for policy replication with other PR instances.

3) Self-healing: Using constraints captured as management policies, the PR must provide continuity of services and uptime despite partial and intermittent node failures. In this case, node failure refers to either a process or network connectivity failure. The latter being a frequent occurrence in a MANET.

4) Policy Supersession: Using policy supersession, the PR maintains the integrity of the operational policies when two partitions merge again after having been separated. It replaces traditional conflict resolution methods that are based on log records as the latter fail to account for the intent of the policy author when propagating the changes to other network partitions.

Fig. 1 below outlines the three layers of a policy distribution framework for MANETs:

- The network communication layer consists of the assets deployed by the coalition. It includes thousands of mobile devices spread across a wide geography communicating over wireless links.
- The Resilient Distributed Policy Repository (RDPR) is a logical representation of the mobile assets that collectively act as the PR and host the operational policies for the coalition.
- The policy-based asset fabric is a logical representation of the mobile assets used for conducting the coalition mission.

Figure 1. Highly decentralized framework for policy management

III. RESILIENT DISTRIBUTED POLICY REPOSITORY

The Resilient Distributed Policy Repository (RDPR) is designed for a dynamic network of nodes with embedded Policy Enforcement Points (PEP). We expect the network to encounter changes in topology and membership. To support the target environment, we expect there need to be multiple Policy Decision Points (PDPs) distributed across the network, each requiring access to the policies that will be used to make decisions. The PDPs may be co-located on the same nodes as the PEPs or on nodes that can be reliably reached by the PEPs. Distributing PDPs throughout the network similarly requires policies to be distributed, albeit not necessarily following the same topology. The RDPR is a solution to the problem of providing reliable, redundant, efficient and up-to-date access to relevant policies and must support the following:

- PDP topology and membership changes
- Administrative policy modifications at one or more points in the network
- Consistency of policies across a connected network
• Automatic resolution of policy editing conflicts
• Uninterrupted service across PDP restarts
• Adaptability to node resources and capabilities

A MANET is the exemplar case for the RDPR. A MANET presents an environment of ad hoc linkage between nodes that can join and leave the network at any time. Network membership and connectivity are all highly dynamic and the member nodes in the network are expected to have differing levels of capabilities including CPU, memory, storage, battery life and bandwidth for communications. We would like the RDPR to provide seamless operations in the context of MANETs or any other environment that is highly dynamic in nature.

The RDPR should support the ability to modify policies through a management interface, perhaps in response to changes in the distributed environment. This requirement makes the design of the RDPR significantly more complex. If policies were not expected to change over the life of the PDP deployment, it would be sufficient to simply copy all the policies to the nearest node with storage. However, with the ability to update policies comes the need to distribute those updates across the network in a timely and consistent manner. Furthermore, in response to a local condition, policy updates may be made locally within a disconnected partition of the network. If and when that partition rejoins other partitions, an automated decision must be made about how to process the update. To support this use case, we provide the ability to override or supersede an existing policy, such that the superseding policies are deactivated (or removed) once the common network is reformed.

In general, we would like PDPs to be resilient to disconnection and restart and continue to render decision requests for any of their associated PEPs across such events. If the PDP does not have storage capabilities, it must reconnect with some portion of the RDPR. If, on the other hand, it has storage capabilities, then it may persist policies so that in the event the PR cannot be contacted on restart, the PDP can continue to service decision requests, albeit with perhaps out of date policies.

The RDPR must anticipate and provide support for PEPs and PDPs implemented on mobile devices having different sets of resources available to them. Resources considered at the PDP nodes include CPU performance, available physical memory, and the ability to provide persistent storage. In the case of persistent storage, nodes may have different types of storage, for example file systems or databases, or a node may have no ability for persistence at all. The latter are referred to as leaf nodes and will typically use a memory cache of policies. Leaf nodes rely on other nodes in the network, referred to as root nodes, to provide policies to their PDPs. Because leaf nodes have no persistent storage, they can not survive a restart in isolation and must instead retrieve policies from a root node in the RDPR.

Root nodes are fully functional nodes and have good CPU, memory and persistence support. As such, they play a primary role in synchronizing policy updates across the network and providing persistent storage on behalf of all the nodes in the distributed repository.

A. Self-Configuring Mobile Data Grid

A mobile data grid defines a highly scalable environment for the dissemination of operational policies in support of distributed applications requiring reliable access to the PR even if a node in the grid fails. Using redundancy, data replication techniques and in-memory caching, it provides high availability, resilience and improved data access performance in a decentralized data management configuration where various devices, designated as nodes of the grid, cooperate on load balancing, access control and self-management of the grid ([2],[3],[4],[5]).

Figure 2. Mobile data grid topology for RDPR

In defining a data grid solution for a RDPR, we further propose a multi-master implementation that combines the resilience of an in-memory data grid with the fault-tolerance capabilities required for MANETs. In a multi-master implementation, the data grid is organized into individual administrative domains that are loosely coupled to provide a unified grid in multiple locations connected through policy defined links. Each domain consists of one or more nodes using asynchronous replication services to synchronize changes between any two domains. Replication services consist of: the bi-directional replication links established between peer domains where changes to the operational policies made on either side of the link are propagated to the other side; and, the management policies used to control the establishment of replication links based on local events detected by each domain such as application load, available capacity and bandwidth limitations.

Fig. 3 below highlights some of the key features of an in-memory data grid policy repository for MANETs. It defines a highly distributed architecture where each domain in the grid manages a subset of the operational policies and coordinates with other domains in the grid to provide the end application with a single unified view of the policy set.

The principles of self-organizing networks ([6],[7],[8]) are widely applied in MANETs for network connectivity and the overall management of the underlying protocols for message routing. A key aspect of a self-configuring RDPR is the ability of each domain in the policy repository network to use local information to achieve resilience and scalability. The absence of a global entity to guide the decisions of each of the constituents in the RDPR network often extends the time required for the whole network to converge. This is due...
in large part to the very nature of the grid where each member communicates through its neighbors and therefore must heavily rely on local information to establish replication channels with its neighbors.

![Diagram of Cooperative Caching of Policies in RDPR](image)

**Figure 3.** Cooperative caching of policies in RDPR

Constraints on the RPDR configuration are captured as management policies such that a policy-based configuration mechanism is used to specify and evaluate configuration rules. The system configuration is checked against these rules and reconfigurations are performed based on the changing conditions in the run-time environment.

Management policies express conditions that, when met, trigger a set of actions. Each domain in the mobile data grid is aware of local conditions and bases all its decisions on these conditions. They include knowledge of who is the nearest active domain in the grid, and how many active replication channels it has. There will be no global state maintained for the grid. By having each domain in the grid react to the local conditions it sees, we can guarantee that the grid as a whole will converge to the desired topology.

Using management policies for configuring the RDPR replication channels in MANETs, we expect that:

- No human intervention will be required for the setup and on-going operations of the policy replication channels within the grid. Once the management policies have been defined, we expect the RDPR to behave in a predictable fashion where each domain enforces the stated policies to provide resiliency when a connected peer domain fails or becomes unreachable.

- The topology of the RDPR will change as it constantly adapts as the underlying network topology changes. Collectively, all the domains in the grid cooperate to deliver the required services for policy management. Individually, a domain represents a self-contained policy repository and maintains a loose binding to its local neighbors in order to sustain the global operation of the RDPR.

- The number of replication channels established will need to be minimal in order to preserve bandwidth. As such, each domain follows specific constraints in order to provision data to its closest peer having a maximum of two replication paths to two other neighboring domains.

![Diagram of Framework for Self-configuring RDPR](image)

**Figure 4.** Framework for self-configuring RDPR

Fig. 4 shows the self-configuration logic for a RDPR domain. The system specifies a set of configuration policies. When the operating conditions change, the Config. Change Detector triggers the Policy Event Generator which invokes the Policy Evaluator to determine a set of new configuration parameters to adapt to the operating environment change.

### B. Self-Healing Mobile Data Grid

A RDPR must be tolerant of failures in nodes and connectivity, hence automated error-handling is desired to assure robustness, resiliency and continuous operation of each domain. This requires that each domain in the RDPR be able to dynamically detect error conditions and programmatically correct for failures.

An RPDR domain is made up of one or more nodes dedicated to the management of the operational policies supporting one or more PDPs and it consists of the catalog services, the container services and the grid performance management services.

The catalog services layer consists of a distributed data directory identifying the repository location of various collections of policies stored within the domain. The container services layer is responsible for the storage, retrieval and indexing of these collections of policies. Within each container, operational policies are organized into policy collections. Each policy collection is assigned a primary store location. Each primary store may have zero or more replicas depending on available resources. The grid performance management services layer is responsible for monitoring, error detection and recovery procedures to ensure the on-going operations of the grid.

![Diagram of RDPR Domain Data Grid Services](image)

**Figure 5.** RDPR domain data grid services
RDPR error detection and correction in a MANET focuses on the performance management services layer of the data grid, more specifically the number of policy collections and number of replicas for each collection. Organizing the operational policies into policy collections increases both scalability and availability of the policy repository. Each policy collection is made up of a primary copy and the configured number of replicas. The following formula Containers = (Policy_Collections * (1 + Replicas)) is used in our model, and enforced by the management policy in order to determine the number of containers required to scale out a single domain.

By collecting and analyzing errors from system logs we develop a knowledge base capturing error conditions and expressing them as management policies. Through a set of policies that tell how to handle each error condition, the RDPR domain automatically reacts to the errors according to a policy evaluation process.

![Mobile Data Grid Infrastructure](image)

**Figure 6.** Error detection and correction in RDPR

In Fig. 6, the sensor detects the malfunction of the data grid, and passes the information detected to the policy engine. The policy engine, based on the specified policies and the captured knowledge base, directs the effector to correct the errors. This self-healing mechanism can be implemented at different levels, for example, at the node level to correct the internal failure of a node, or at the system level to eliminate permanently failed nodes and add new nodes to maintain the performance and resilience of the data grid.

C. **Policy Supersession**

The dynamic nature of MANETs often causes nodes to disconnect and re-connect to the network for a variety of reasons. A network partition occurs when a cluster of mobile devices moves out of range of the radio access network making these nodes inaccessible to other devices. In this case, all replication links to the operational policy repository within the cluster are severed due to loss of network connectivity. While the RDPR provides protection against the intermittent loss of a replication link, it does not completely eliminate the risk of a network partition should all links to a policy repository domain get severed for an extended period of time.

In some scenarios, network partition implies that the active operational policies may no longer apply and need to be modified. These modifications are usually temporary in nature and only apply when the network is in an abnormal state. When the network reconnects, these temporary policies often conflict with the original policies. We propose an automatic resolution to these conflicts, where the original policy is superseded by a temporary policy and reverts back to the original policy when the network goes back to normal.

This section will describe a specific MANET scenario and the need for policy supersession.

In a network with mobile devices, some devices may have sensitive information and belong to different parties. Authorization policies need to be deployed to allow or disallow sharing of information among the different parties. In other cases, one party may need to temporarily “borrow” a device from another party while their own device is not working. For example, in a military coalition, the United States and the United Kingdom have mobile networks of sensors that are deployed for a particular mission. As seen in Fig. 7, in the initial network, the UK has two partitions (Node 2 and Node 3), three acoustic sensors and one camera.

![Scenario for policy supersession](image)

**Figure 7.** Scenario for policy supersession

The correlator requires three acoustic sensors in order to determine the bearings of the sound detected. When a sound is detected, such as a gun shot, the correlator signals the camera to take an image that is then sent to the UK Monitor for viewing. Initially, the US and UK RDPR nodes, which contain the policies authored with the policy editor, are connected. If Node 3 goes down, or moves out of range the UK no longer has access to one of the acoustic sensors, so a camera image will not be sent to the monitor. The UK asks the US for permission to use their acoustic sensor and the US will then deploy a policy that gives the UK the appropriate access. When this new policy is deployed, it will create a conflict with the original policy that disallows access to the camera. When Node 3 is restored, the UK should no longer have access to the US camera. In the next section, we describe a model that allows policy administrators to supersede policies temporarily and the original policies to be automatically restored when the network is restored.

When two domains of the same grid are separated, changes to the policies in any one domain are recorded as
supersession policies. Both the original policy and the superseding policy are deployed to the control policy repository in order to maintain traceability and overall integrity of the control policies. The three steps of the Policy Supersession model include:

Step 1- Deploy: During this step the operational policies are deployed to the RDPR along with any templates used for editing the policies.

Step 2- Supersede: During this step, a new copy of an existing operational policy is created and modified to deal with a new situation that has emerged while the network is partitioned.

Step 3- Re-instate: During this step of the process, the policy framework deactivates the superseding policy and reactivates the superseded policy as the two network partitions rejoin.

It is common practice to have both the original policy and the policy that supersedes it deployed to the RDPR. When a superseding policy is activated, the corresponding policy it supersedes is automatically de-activated by the policy framework. Vice-versa, when the two partitions rejoin and the reason for supersession is no longer valid, all superseding policies are de-activated and the corresponding policies they superseded are automatically re-activated.

A superseding policy can have the same name as the policy it supersedes. In fact, in our model, a superseding policy is often created by copying an existing policy and modifying the condition or decision clause of the original policy. As such, the superseding policy retains all the meta-data from the operational policy it supersedes with the addition of two attributes that are unique to the superseding policy:

- `System.supersede=true`
- `System.supersede.id=<unique ID of the superseded policy>`

The first attribute designates that this policy is a superseding policy and as such it has a different life-cycle from other control policies in the RDPR. The second attribute uniquely identifies the original policy that this policy supersedes. As such, a superseding policy can supersede one and only one control policy.

An event is triggered when any two partitions in the grid re-connect to each other. This event will signal the policy framework to restore the original state, which will deactivate all superseding policies and re-activate all superseded policies. In the next section, we will describe how these superseding attributes can be easily added through the use of a web-based policy management tool.

IV. REFERENCE IMPLEMENTATION

A. IBM Websphere eXtreme Scale

IBM WebSphere eXtreme Scale (WXS) is an in-memory data grid that provides transactional integrity, high availability and transparent fail-over. When running a single domain, WXS consists of a cluster of catalog servers and one or more container servers. The catalog servers provide a directory service for all container servers that store and replicate application data. WXS can also run a unified grid in multiple domains connected through user defined links. Each grid is fully independent and runs its own catalog service. The links defined between two domains are used by WXS to make both domains identical. These replication links are bi-directional, so changes made on either side of the link will be propagated to the other side. By defining inter-links between domains, various topologies can be constructed, such as line topology, tree topology, ring topology, etc., to meet different applications’ needs. When running WXS in a MANET, the interconnections between domains can frequently fail as nodes can dynamically leave and join the network. In our implementation of the RDPR, we address these limitations using management policies.

B. Policy Management Framework

The Policy Management Library (PML) [14] is a Java-based toolkit that provides a policy-language independent framework, and includes a policy repository, policy decision points, and policy enforcement points. The framework allows for the analysis of sets of policies for simultaneous applicability, coverage and conflicts. PML examines
evaluation requests and selects appropriate policies for a given request, taking into account the instance data provided to the request, attributes and type (authorization or obligation) of the PEP and the policies. PML is designed for small footprints in MANETs.

PML uses the Apache Imperius [17] incubator project’s Java binding for evaluation of CIM-SPL policies [16]. For the purpose of our research activities, we extended the PML framework to use WXS for policy management.

Figure 11. PML framework using WXS data grid

C. Configuration Policies

The proposed management policies for a self-configuring, self-healing RDPR consist of a small number of policies that govern how each domain establishes replication links with other domains and manages available nodes and replicas. The policies are evaluated independently by each domain whenever a particular event occurs that changes that particular domain’s context. The evaluation of each policy is done over a set of instance data provided by the domain managed environment. The relevant instance data for each domain include:

- **DomainsList**: a Java object that contains a list of domains accessible to this particular domain.
- **DomainLinksCount**: a Java object that represents the number of active replication links that this domain has.
- **MaxSyncReplicas**: an integer value that specifies the maximum number of synchronous replicas for each policy collection within a given domain.
- **MinSyncReplicas**: an integer value that specifies the minimum number of synchronous replicas for each policy collection within a given domain.
- **AvailableNodes**: a Java object with the list of available containers within a given domain.

For illustration purposes, Fig. 12 below shows the SPL policy used to establish a policy replication link between two domains and trigger the onMerge event for re-instating any superseded policies.

![Sample policy for re-instating superseded policies](image)

D. Authoring Tool

The Policy Authoring, Management and Negotiation Tool (PAMNT) [11, 12] is a web-based application that allows an operator to author policies in a structured policy language using templates. From the user perspective, PAMNT provides drop-down choices to allow the author to create policies with human readable phrases (e.g., “If the image monitor is controlled by the US, then downgrade the resolution by 50%.”). The template enables the policies to be saved in a structured policy format that is executable in the runtime system, such as CIM-SPL [15].

In the management component of PAMNT, authored policies are deployed to a policy repository for evaluation in a runtime system. The policy repository, in this case, is configured to use a WXS grid server for deployment. In our configuration of WXS, a policy deployed to one node of the grid server will automatically replicate to the other grid nodes. If one of the grid nodes goes down or moves out of range, a policy may need to be modified temporarily by the commander. As in the policy supersession scenario described earlier, an authorization policy needs to be modified to temporarily “allow” a US network to share information with a UK network (Fig. 13).

![Superseding policy editor](image)

PAMNT enables deployed policies to be modified by creating a copy of the active policy and adding the supersede attributes (Fig. 14). The attributes contain a unique identifier of the original policy, so that the policy can be automatically restored when the network reconnects.
The WXS server detects an onMerge even t and evaluated (Fig. 15). When the connection to the no de is re-established, the original policy is restored. Both the original and the superseding policy are stored in the repository of deployed policies, but only active policies are looked for policies that contain the System.supersede=true attribute. The superseding policy is inactivated and the original policy becomes inactive. The superseding policy is inactivated and the original policy is restored.

When the modifications are finished, the superseding policy is deployed and the original policy becomes inactive. Both the original and the superseding policy are stored in the repository of deployed policies, but only active policies are evaluated (Fig. 15). When the connection to the node is re-established, the WXS server detects an onMerge event and looks for policies that contain the System.supersede=true attribute. The superseding policy is inactivated and the original policy is restored.

V. CONCLUSION

We have developed a Resilient Distributed Policy Repository based on data grid technology and policy mechanisms that can provide reliable access to control policies in a MANET environment. It has self-configuration and self-healing capabilities, and presents a unified view of system control policies to management applications. We have also incorporated support for policy supersession so that policies that are changed while the network is partitioned can be changed back to their original form. A functioning prototype has been implemented that demonstrates a coalition scenario.

Our future research will focus on resolving editing conflicts more broadly by using meta-policies.

ACKNOWLEDGMENT

This research was sponsored by the U.S. Army Research Laboratory (ARL) and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. ARL, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES