REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
JULY 2009	Conference Paper Postprint	June 2007 – March 2009

4. TITLE AND SUBTITLE

ENABLING DISTRIBUTED MANAGEMENT FOR DYNAMIC AIRBORNE NETWORKS

5a. CONTRACT NUMBER
FA8750-07-C-0110

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62702F

6. AUTHOR(S)

Cho-Yu J. Chiang, Gary Levin, Shihwei Li, Constantin Serban, Michelle Wolberg, Ritu Chadha, Gregory Hadynski, Lee LaBarre

5d. PROJECT NUMBER
NATM

5e. TASK NUMBER
TE

5f. WORK UNIT NUMBER
LC

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Telcordia Technologies, Inc.
One Telcordia Drive
Piscataway, NJ 08854-4182

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/RIGC
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
N/A

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TP-2009-64

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release; distribution unlimited PA# 88ABW-2009-1101 Date Cleared: 20-March-2009

13. SUPPLEMENTARY NOTES

14. ABSTRACT
In this paper we describe our experience with integrating a distributed policy-based management system (DRAMA) with an open-source network management system (OpenNMS). Network operations seeking the benefits of policy-based network management often have pre-existing network monitoring systems. While these pre-existing systems are capable of monitoring the network, they are limited in their: 1) ability to provide distributed network management, 2) support for automatically reconfiguring the network in response to network events, and 3) ability to adjust management traffic bandwidth consumption based on network conditions. For dynamic networks such as those consisting of airborne platforms, there is a need to provide the above capabilities in any management solution while preserving any underlying management systems. As a result, we integrated DRAMA with OpenNMS to add distributed policy management capability to a commonly used network management system. In this paper, we describe the background for this effort, our approach for integrating OpenNMS with DRAMA, and the design of a distributed resource indirection framework that allows the use of the same policies across different distributed policy decision points managing network devices with different attribute values.

15. SUBJECT TERMS
Mobile networks, intelligent agents, policy based networking, ad-hoc networks, network management

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
					Gregory Hadynski
a. REPORT	b. ABSTRACT	c. THIS PAGE	UU	9	19b. TELEPHONE NUMBER (*Include area code*)
U	U	U			N/A

Enabling Distributed Management for Dynamic Airborne Networks

Cho-Yu J. Chiang, Gary Levin, Shihwei Li, Constantin Serban, Michelle Wolberg, Ritu Chadha
Telcordia Technologies
{chiang, gary, sli, serban, mwolberg, chadha} @research.telcordia.com
Gregory Hadynski, Air Force Rome Laboratories, Gregory.Hadynski@rl.af.mil
Lee LaBarre, MITRE, clabarre@mitre.org

Abstract

In this paper we describe our experience with integrating a distributed policy-based management system (DRAMA) with an open-source network management system (OpenNMS). Network operations seeking the benefits of policy-based network management often have pre-existing network monitoring systems. While these pre-existing systems are capable of monitoring the network, they are limited in their: 1) ability to provide distributed network management, 2) support for automatically reconfiguring the network in response to network events, and 3) ability to adjust management traffic bandwidth consumption based on network conditions. For dynamic networks such as those consisting of airborne platforms, there is a need to provide the above capabilities in any management solution while preserving any underlying management systems. As a result, we integrated DRAMA with OpenNMS to add distributed policy management capability to a commonly used network management system. In this paper, we describe the background for this effort, our approach for integrating OpenNMS with DRAMA, and the design of a distributed resource indirection framework that allows the use of the same policies across different distributed policy decision points managing network devices with different attribute values.

1. Introduction

The benefits of distributed policy-based network management for mobile ad hoc networks have been demonstrated in previous work [1][3][4][5][6]. All of our prior effort was "green field" work and assumed a clean slate where no management capabilities existed and therefore management capabilities would need to be developed from the ground up for the network elements and the network environment at hand. In reality, however, a policy-based management system often has to interface and interoperate with existing network management systems in order to augment their capabilities for many reasons, including cost effectiveness and continuity of operations.

Our work was performed in the context of an airborne network consisting of multiple high-speed flying platforms, a handful of ground mobile nodes and a ground control station. The nodes communicate with each other using several different types of radios. On board these airborne platforms, there are network assets on multiple local area networks. If a traditional centralized network management solution were used to manage such a network, either the entire airborne network needs to be managed from the ground, or each platform needs to be managed independently by its local management systems. The former is not considered an effective approach because of lack of bandwidth efficiency and inability to support disconnected operations; the latter is not a perfect solution either as the ground control station will not have sufficient control over the platform networks. In addition, it is desirable that the management system be able to respond to network events by supporting autonomous changes to the network.

Since it is highly desirable to support any existing platform-based network management solution, we explored the concept of integrating a distributed policy-based management system with a fixed-network management solution. Such a combination makes sense because 1) the existing management system can remain in place and perform its function as usual, 2) the ground control station can have control over the entire network, 3) the combined system can respond to network events with appropriate changes by allowing policies to invoke local management scripts, and 4) bandwidth efficiency is significantly enhanced as policies are used to control bandwidth consumption for network management purposes based on dynamic network conditions.

The challenge, therefore, is to architect a solution that preserves the capabilities of the existing monitoring system while augmenting its capability to provide policy-based control of the network. The result is a hybrid management system that allows the existing system to preserve its "look and feel" for the network administrators, while adding policy management capabilities and achieving bandwidth efficiency. This paper describes the architecture of this integration, the enhancements to DRAMA [3] to allow for its integration with OpenNMS [7], an open-source centralized network management system, and the results accomplished via the integrated system. To facilitate managing multiple airborne platforms with different configuration settings using the same set of policies, we also designed a resource indirection representation framework to facilitate the distributed network management paradigm.

The remainder of this paper is organized as follows: Section 2 provides background about the target airborne network, DRAMA, and OpenNMS. Section 3 discusses the integration of OpenNMS with DRAMA. Section 4 provides an overview of the *resource indirection framework*, which allows policies to be written once, and be applicable to network elements across multiple platforms that comprise the network. A discussion of the functionality provided by the integrated system, lessons learned from this exercise and possible future work are provided in Section 5, followed by a summary in Section 6.

2. Background

The focus of this work is to design a distributed policy-based management solution for a dynamic airborne network by integrating a distributed PBNM system with a conventional network management system. Below we provide some background for the work.

The airborne network under consideration comprises an immobile ground node, multiple airborne nodes and several surface mobile nodes. All nodes contain local area networks on board, and they use multiple different types of radios to communicate with each other. The network is a special type of Mobile Ad hoc NETwork (*MANET*) communicating over wireless links. Each node can act as a router forwarding transit traffic. Major characteristics of MANETs include absence of network infrastructure, dynamic wireless link conditions, and unstable network topology. Because physical environments could be noisy and distances between nodes change over time, effective bandwidth on radio links fluctuates and link connectivity changes. These characteristics provide a strong motivation for providing self-forming, self-configuring, and self-healing capabilities in the network.

DRAMA, a distributed policy-based network management system [1][3], is a good candidate for the airborne network for the following reasons: 1) its ability to provide distributed network management even when the network is partitioned, 2) its support of automated changes to the network configurations in response to network events and 3) its ability to adjust management traffic bandwidth consumption based on network conditions. The high-level architecture of the DRAMA system is shown in Figure 1. As shown here, a collection of Policy Agents with different roles manage the airborne 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. Policy Agents react to network status changes on various levels (globally, domain-wide, or locally) by automatically reconfiguring the network as needed to deal with fault and performance problems. In this hierarchical architecture, any node can dynamically take over the role that was assumed by another node to provide resilience to network failures.

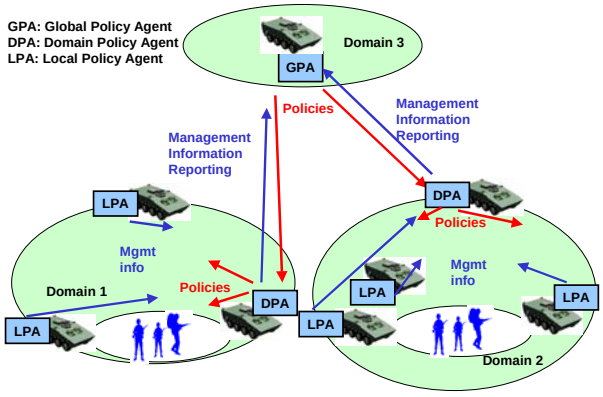


Figure 1. High-level Architecture of the DRAMA System

DRAMA policies are Event-Condition-Action (ECA) [2] obligation policies. A policy can be triggered by a single event or one of many events associated with the policy. The condition part of a policy is represented by a boolean expression containing system and/or policy variables. Whenever an event occurs, DRAMA

identifies the currently activated policies that are triggered by the event and evaluates their conditions. If the condition of a policy evaluates to true, then all the actions associated with the policy will be executed.

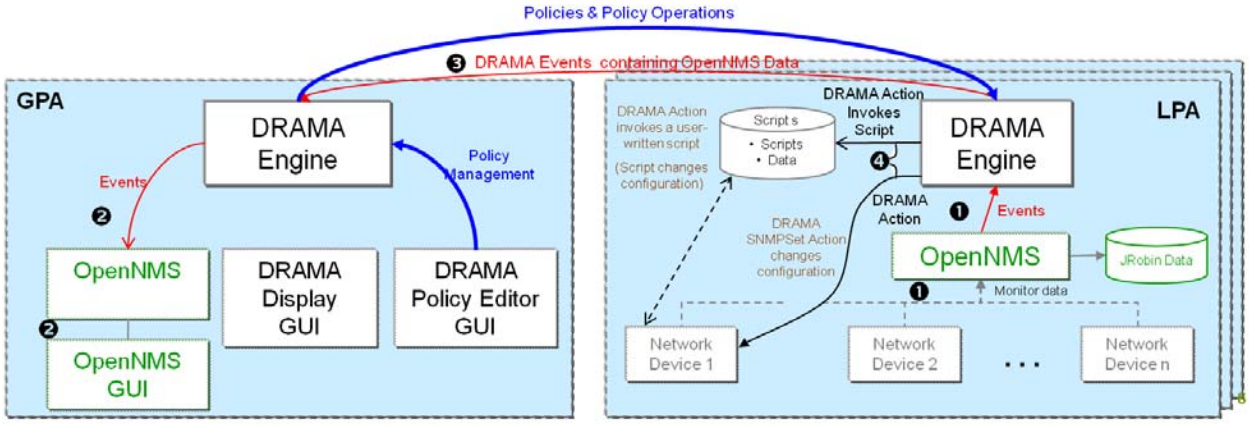


Figure 2. DRAMA and OpenNMS Integration – High-Level Approach

OpenNMS: OpenNMS [7] is an open source network management system. It provides scalable monitoring facilities for a wide range of devices and services. OpenNMS uses a centralized network management model, that is, a designated management station is responsible for monitoring the health of the network. The station can be configured to poll devices, receive traps, send notification events, generate alarms based on threshold crossings, and provides a Web Server that allows data to be presented on a web-based dashboard. It also incorporates other open source efforts including JRobin [10] to produce graphs displaying the values of designated monitored parameters as they vary over time.

3. Integration of DRAMA and OpenNMS

3.1 Motivation

OpenNMS uses a centralized management station to collect data from all the nodes in the network. The management station periodically polls the devices to retrieve their data. Therefore, a break in network connectivity would result in OpenNMS not being able to collect data and manage the network. While losing network connectivity is infrequent in a wired network, it is quite common in a wireless ad hoc network. In addition, periodic polling is an expensive and unnecessary operation for collecting data in bandwidth deficient networks. On-demand data push is considered a more appropriate model for such networks. OpenNMS is essentially a network monitoring system and does not provide the capability to automatically trigger management actions, with the exception of sending notification alerts. Finally, OpenNMS is completely unaware of its own bandwidth usage, and as a result, its management behavior is independent of network conditions.

On the other hand, DRAMA complements OpenNMS by supporting the management data push model, providing automated distributed management actions, and changing its management behavior (e.g., changing reporting intervals) according to network conditions. Nevertheless, DRAMA does not have the rich monitoring facilities that OpenNMS provides. DRAMA also does not compare to OpenNMS with respect to monitoring report generation and display, as well as the ability to alert network administrators using a variety of alerting mechanisms.

3.2 Approach

To take advantage of the respective strengths of both DRAMA and OpenNMS and keep the look and feel of OpenNMS for network administrators, we designed and implemented an approach to integrate the two management systems that exhibites the strengths of both. A high-level architecture showing our approach integrating the two systems is shown in Figure 2. In this figure we simplify matters by showing only a GPA and some LPAs. OpenNMS is co-located with DRAMA on both the GPA node and LPA nodes. On

the LPAs, OpenNMS is used to monitor the local area networks on board the nodes, and it can be configured to feed the data to J-Robin for generating monitoring graphs. We connect OpenNMS and DRAMA by having OpenNMS send certain events to DRAMA. These events are processed by the event conversion component supplied by OpenNMS to convert OpenNMS events into DRAMA events. Once DRAMA receives events, two types of policies could be triggered. The first type of policy is a thresholding-based report policy, which will determine if an event should be forwarded up the management hierarchy, and if so, what types of events should be forwarded and how they should be forwarded. The second type of policy could trigger local management actions, such as a script being invoked to change certain configuration settings on a device. When an event is forwarded by DRAMA, it will reach its parent node in the hierarchy (a DPA or a GPA); the parent node will convert the event and pass it to OpenNMS, as if a device managed by OpenNMS had been configured to push data to OpenNMS. The process of converting between OpenNMS and DRAMA events is illustrated in Figure 3.

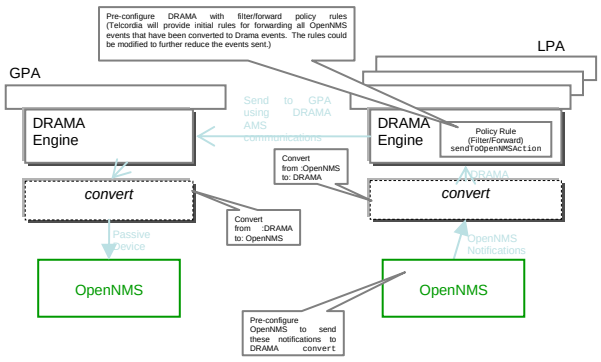


Figure 3. DRAMA/OpenNMS Event Conversion Process

The above architecture has the following salient features. First, from the OpenNMS viewpoint, it monitors the entire airborne network from the GPA, while the relay/filter function provided by DRAMA is completely transparent. Second, DRAMA is used in this management framework, and local management policies can be triggered to realize the distributed policy management paradigm. Third, it takes advantage of DRAMA's capability to reduce management traffic bandwidth usage according to network conditions. Finally, for DRAMA-aware network administrators, they can watch the network monitoring reports and decide if any on-board policies should be activated or deactivated, or any new policies should be pushed to the remote platform to adjust network communication behaviors. These policy operations are transparent to OpenNMS operations.

3.3 Implementation

The implementation of the integration comprises two independent components: the conversion of events between the two systems and the transport of the events between nodes in the system.

One guiding principle of the design was to minimize the impact on the OpenNMS implementation, avoiding the modification of any OpenNMS source code. A second principle was to keep libraries for the two systems independent, to avoid complexities in the build process and to allow support of a version of DRAMA without OpenNMS. This was accomplished by using TCP to avoid the tight binding of RMI and using introspection and dynamic loading of classes to remove compile-time dependencies.

3.3.1 Conversion of events

The conversion of events between OpenNMS and DRAMA comprises the following components:

- Capture of OpenNMS Events
- Creation of DRAMA Events from OpenNMS Events
- Creation of OpenNMS Events from DRAMA Events

3.3.1.1 Capture of OpenNMS Events

OpenNMS provides a highly configurable system and support for the development of plug-in components. We wrote an `EventListener` and installed it as a service in OpenNMS. This component receives a copy of every generated OpenNMS Event, filters it according to a configurable list of patterns (which reduces the number of events reported to DRAMA), and sends the OpenNMS Event over TCP as a Serialized String of the XML representation of the event to the component that creates DRAMA Events. The XML String is marked with "`opennms:`" so that the receiver can recognize the source of the message.

3.3.1.2 Creation of DRAMA Events from OpenNMS Events

The component that creates DRAMA Events listens on a TCP port. If a String arrives, prefixed by

"`opennms:`", it is treated as the XML for an OpenNMS Event and a DRAMA `openNms` Event is created, with the XML as a component field.

3.3.1.3 Creation of OpenNMS Events from DRAMA Events

A DRAMA Action is provided to transmit DRAMA `openNms` Events to the GPA. It uses the `PersistentCommsSender` to reliably and persistently transmit the event to the GPA.

A DRAMA application implements this component, listening with a `PersistentCommsReceiver`. When a String is received, it is the XML representation of an OpenNMS Event.

OpenNMS provides a port to which the XML representation of an OpenNMS Event may be sent to create the event. The DRAMA Event `openNms` contains the original XML for the OpenNMS Event, so we need only pass it to the port and the event is recreated.

3.4 Transport of Events

Connectivity in an *ad hoc* wireless network may change over time and occasionally be lost. DRAMA provides reliable delivery of messages, but reliability only guarantees that the sender will know whether the message has been received. The classes `PersistentCommsSender` and `PersistentCommsReceiver` provide a layer above reliable communications, queuing messages and resending periodically until the message has been received.

3.5 User Scripts for Configuration

The need to reconfigure devices on remote platforms when specific network conditions are detected is a challenge for distributed ad hoc networks. How do you control actions on remote platforms when connectivity to that platform may not be possible from the central platform? DRAMA's capability to have policy rules that enforce actions when events are detected provides the infrastructure necessary to solve this challenge.

Another important issue is that while monitoring of network devices utilizes standard SNMP MIBS, configuration (or re-configuration) of network devices often requires software that is not standards-based (e.g., CLI interface, device-specific loading, etc.).

The solution for the OpenNMS integration is to allow DRAMA to support a generic policy action that can invoke a user-written script. The user-written script can then provide the code needed to interface to the device and re-configure it as appropriate. With this capability policy rules can be written that can be triggered by events in the network to invoke re-configuration actions on the remote platforms without full connectivity to that platform.

4. Device Resource Indirection

4.1 Motivation

One of the challenges for policy-based network management is ensuring that the policy language effectively supports the operational requirements of the administrators. It must not burden the administrators with additional work to manage all the nodes/devices in the network. In order to allow DRAMA to define policy rules that are applicable to many platforms that have similar device resources, there is a need to provide *device resource indirection* (or *device resource meta-data*) in a policy rule action for attributes such as the management interface address and the management port of a network device (device resource). This meta-data will allow a generic policy rule to be written and that policy rule can be enforced for the local device resources on each platform. Each platform keeps its own platform-dependent device resources.

By applying meta-data to policy rules, a DRAMA system will:

- Require fewer rules. Having fewer rules to write means easier management.
- Distribute fewer rules. Having fewer rules to distribute results in lower network bandwidth usage. This is critically important in a wireless network environment.

4.2 Approach

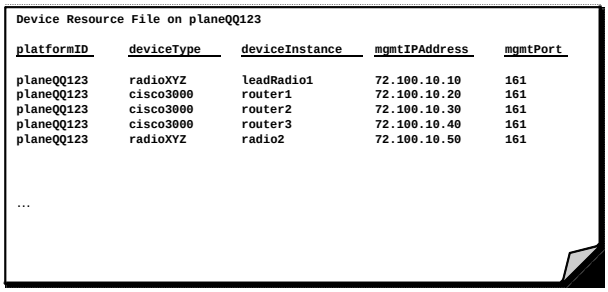
The concept of meta-data is not new. Applying device resource meta-data to DRAMA is similar to using variables in programming. The concerns to be addressed are as follows:

- How to statically specify device resource data on each platform?
- How to statically represent device resources in a rule?

- How to dynamically bind generic device resource meta-data in a rule to the local data on each platform during rule enforcement?

4.2.1 Platform-dependent Device Resource Data

Each platform is preconfigured with a device resource file, which contains device resource data relevant to the platform. Each network device under DRAMA's management is represented as a device resource in the file. A device resource contains information including the platform id, the device type, the device instance name on the platform, and the IP address and a port number through which DRAMA talks to the device. Figure 4 provides an example.



Device Resource File on planeQQ123

platformID	deviceType	deviceInstance	mgmtIPAddress	mgmtPort
planeQQ123	radioXYZ	leadRadio1	72.100.10.10	161
planeQQ123	cisco3000	router1	72.100.10.20	161
planeQQ123	cisco3000	router2	72.100.10.30	161
planeQQ123	cisco3000	router3	72.100.10.40	161
planeQQ123	radioXYZ	radio2	72.100.10.50	161

…

Figure 4. A Device Resource File Example

When DRAMA boots up on a platform, it verifies the device resource file and loads data from the file into a device resource repository. The device resource repository, a singleton in the system, provides device resource inquiry services to the system. The repository takes a criterion and returns a list of device resources that match the criterion. The results are cached to speed up future searches based on the same criterion.

4.2.2 Device Resource References in a Policy Rule

A policy rule contains a list of actions. To enforce a rule, each action is enforced in the sequence order it appears on the list. An action might be to monitor certain devices; the next action might be to configure other devices. We want to apply an action to each of device resources we care about. For this reason, in a rule, the place to associate the searching criterion and device resource references is in each individual action: an optional device resource filtering criterion is associated with an action; device resource references are used in the parameters to the action.

4.2.2.1 Device Resource Filtering for Action

Each rule action can have an optional device resource filtering criterion. If a criterion appears, the action will be applied to each of device resources matching the criterion during the enforcement.

A filtering criterion has three parts: *<Platform id>.<Device type>.<Device instance name>*. Each part allows a string or a wildcard character, '*'. Using a wildcard is to match all device resources for the corresponding part. In addition, <Platform id> allows a special keyword, LOCAL. Using LOCAL is to match all device resources that belong to the local platform where DRAMA is running.

A device resource matches a criterion if and only if the device resource matches each part of the criterion.

4.2.2.2 Device Resource References in Action Parameters

Device resource references can be used only in the parameters to an action. The five fields in the matched device resource are *$deviceResource->platformId, $deviceResource->deviceType, $deviceResource->deviceInstance, $deviceResource->mgmtIpAddress* and *$deviceResource->mgmtPort*.

During the enforcement, DRAMA will substitute each reference with its true local environment value. To escape the value substitution to a reference, prefix a backslash, '\', to the reference. A pair of curly braces can surround a reference to clearly mark the start and the end of the reference (for example, *${deviceResource->mgmtIpAddress}*). This is useful when we try to do string concatenations in the parameters.

4.2.3 Dynamic Binding in Action Enforcement

4.2.3.1 Action Enforcement

A device resource criterion in an action is optional, and we use the following algorithm to enforce the action:

- *If NO criterion is provided, then enforce the action only once.*
  - *If parameters refer to device resource, throw an exception (rule enforcement fails).*
- *If a criterion is provided,*
  - *If it matches NO device resources, then skip this action enforcement, continue to the next action,*
  - *Else enforce the action for each of the matched device resources.*

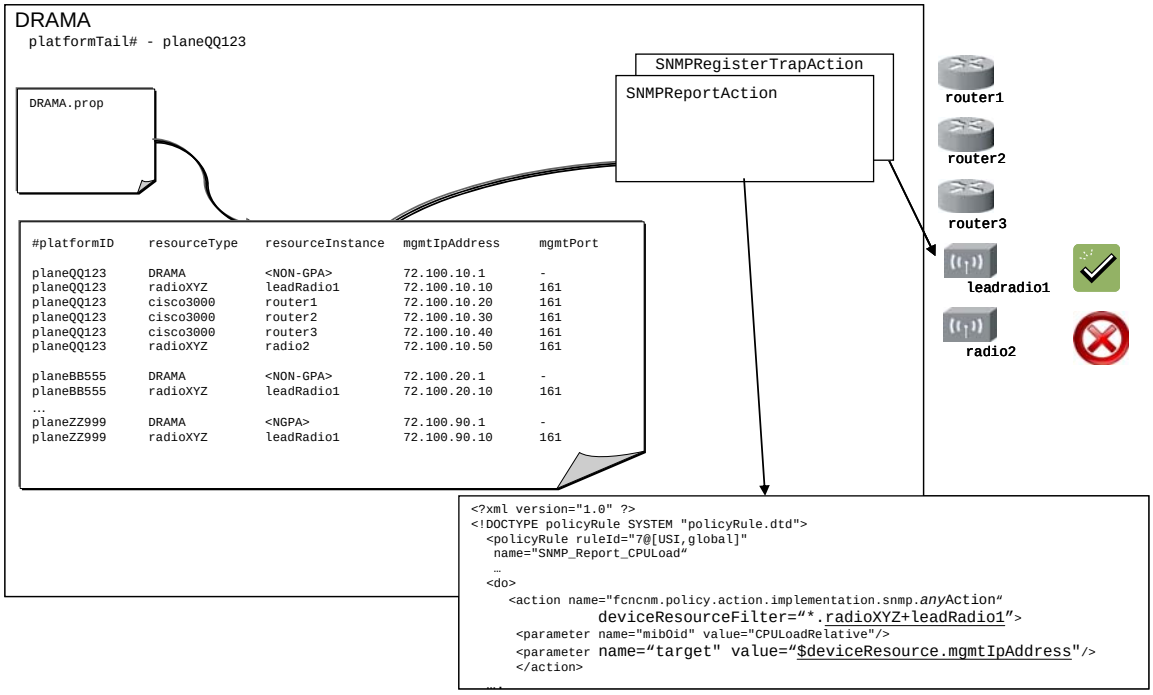


Figure 5. A Resource Indirection Example

4.2.3.2 Dynamic Device Resource Binding

For each matched device resource, we perform value substitutions for action parameters against the device resource, and use the new parameters after substitutions for the actual work in the action enforcement. The work of parsing parameters and of performing substitutions is delegated to the interpreter generated from ANTLR [9] a popular Java parser generator.

To conclude this section, we use Figure 5 to illustrate the concept of resource indirection. In this figure, a resource description file that contains the resource descriptions of all resources on all nodes is shown. A DRAMA policy excerpt is also shown to illustrate the concept. Assume in this example that *platformID* is *planeQQ123*. The policy specifies that if it is triggered, its action will be applied to *radioXYZ* and *leadRadio1* only, and the *mgmtIPAddress* attribute of the applicable devices will be passed to the action as a parameter so that the action can retrieve the CPU load information via SNMP. Therefore, this policy is generically applicable to all nodes where *radioXYZ* and *leadRadio1* are on board, as long as the resource description file contains the correct attribute information.

5. Discussion

This work to extend DRAMA is currently being integrated into the customer's operational environment. The benefits of this work are currently being evaluated. However, we believe that this will provide valuable insights into specific areas where further work may be needed to derive additional benefits from the inter-operations of DRAMA (policy-based network management) and OpenNMS (open source network monitoring software). Specifically we are interested in the following:

- What capabilities are needed to allow network management staff to effectively troubleshoot network problems?

  The capabilities provided by this work address the need to notify the network administrators of network problems across the distributed network. Further operational use cases are needed to define the data flows that will be required in this integrated environment to troubleshoot problems on remote platforms.

- Can network administrators work effectively with the OpenNMS display (dashboard) and DRAMA's policy-based controls (policy editor)?

  Both the OpenNMS and the DRAMA network management consoles will be used in this solution.

A better understanding of the needs of network managers to access both consoles is required to understand any display integration that may be required.

- Does this integration solution work effectively for mobile ad hoc networks?

  The bandwidth-limited mobile ad hoc network provides many operational challenges for network management. In particular, the overhead to perform network monitoring and control should not over-burden the network. As this solution is made operational, it will be important to understand whether this architecture meets the requirement that network management traffic use the available network capacity in an efficient manner.

Finally, we plan to perform a detailed performance study of the integrated software using the Virtual Ad hoc Network testbed [8], which allows unmodified software instances to communicate with each other over a high-fidelity simulated network that can simulate various airborne network scenarios. Not only will this approach help us gauge the system performance and fine-tune the design, but it will also test the software robustness in a highly dynamic wireless network environment.

6. Conclusion

This paper describes the integration of DRAMA with OpenNMS to provide a network management solution for airborne networks. The rich OpenNMS monitoring facilities are used to manage local area networks on the airborne platforms, and via event relay facilities provided by DRAMA, OpenNMS can be used to monitor the entire airborne network without incurring excessive bandwidth overhead. In this integration, DRAMA provides a generic capability to invoke user-written scripts as policy actions allowing for automated remote re-configuration. In addition, DRAMA also allows network administrators to activate and deactivate policies to change the airborne network behavior according to the collected network monitoring status on the ground. Lastly, the DRAMA policy language was extended to provide a generic resource indirection framework, which makes the creation and maintenance of policy rules more operationally viable.

7. Acknowledgments

The authors would like to acknowledge the operational input provided by the airborne network managers who reviewed the integrated software.

8. References

[1] R. Chadha et al., “Policy-Based Mobile Ad Hoc Network Management”, Proceedings of the IEEE 5th International Workshop on Policies for Distributed Systems and Networks, Yorktown Heights, New York, June 7-9 2004.

[2] R. Chadha, “Beyond the Hype: Policies for Military Network Operations”, ICSNC 2006, French Polynesia, October-November 2006.

[3] R. Chadha, Y.-H. Cheng, C.-Y. J. Chiang, S. Li, G. Levin, and A. Poylisher, “DRAMA: A Distributed Policy-Based Mobile Ad Hoc Network Management System”, Proc. of the 2005 Military Communications Conference (MILCOM 2005), Atlantic City, NJ.

[4] C.-Y. J. Chiang, R. Chadha, G. Levin, S. Li, and Y.-H. Cheng, “AMS: An Adaptive Middleware System for Ad hoc Networks”, Proc. of the 2005 Military Communications Conference (MILCOM 2005), Atlantic City, NJ.

[5] C.-Y. J. Chiang, R. Chadha, Y.-H. Cheng, S. Li, G. Levin, and A. Poylisher, “A Novel Software Agent Framework with Embedded Policy Control”, Proc. of the 2005 Military Communications Conference (MILCOM 2005), Atlantic City, NJ.

[6] C.-Y. J. Chiang, Y.-H. Cheng, S. Demers, P. Gopalakrishnan, L. Kant, R. Chadha, S. Li, G. Levin, A. Poylisher, Y.Ling, S. Newman, and R. Lo, “Performance analysis of DRAMA: A distributed policy-based system for MANET management”, Proc. of the 2006 Military Communications Conference (MILCOM 2006), DC.

[7] OpenNMS, *http://www.opennms.org/*

[8] P. Biswas et al, “An Integrated Testbed for Virtual Ad hoc Networks“, Proceedings of TRIDENTCOM 2009, April 6-8, 2009, DC, USA.

[9] ANTLR, *http://www.antlr.org/*

[10] JRobin, *http://www.jrobin.org/*