Generative Policy model for Autonomic Management

D. Verma, S. Calo, S. Chakraborty
IBM Research
Yorktown Heights, NY, USA

E. Bertino
Purdue University
West Lafayette, IN, USA

C. Williams, J. Tucker
UK Dstl, Porton Down,
Wiltshire SP4 0JQ, UK

B. Rivera
Army Research Labs
Adelphi, MD, USA

***Abstract*—Policy based Management has been used for autonomic management in several systems, but current approaches have used a model of policies that follow the event-condition-action paradigm or variants thereof. While proven successful in many contexts, these systems nevertheless require a manager to predict future conditions and define the applicable rules in advance. A higher level of autonomic behavior can be enabled if the managed system could be allowed more flexibility in its actions, while maintaining the requirements for consistency and compliance that have led to the successes of current policy based management paradigm. In this paper, we present a new approach for policies - which enables managed systems to take more autonomic decisions regarding their operations.**

Keywords—Policy based Management; Coalition Operations, Software Defined Architectures, Generative Policies, Autonomic Systems

I. INTRODUCTION

Policy based management has proven itself useful to simplify the management and operations of complex distributed environments in many different domains, enabling management advances in many domains such as industrial systems[1], computer network management [2], enterprise access control[3], distributed systems management [4], security and identity management[5], storage area networks[6], military sensor networks [7], self-managing cells [8], and many other areas. Due to their wide applicability, several policy management frameworks have been proposed, among which the IETF/DMTF framework [2] has been widely adopted. Several policy specification languages and information models, both general purposes and for specific domains, have been proposed such as the IETF/DMTF Policy Core Information Model - PCIM[9], Ponder [10][11], XACML[12], KaoS[13], IBM Autonomic Computing Policy Language- ACPL[14] and many others as discussed in survey papers [15] and used to create autonomic management systems[16].

Despite the tremendous benefits and usefulness of policy based management in many different domains, currently deployed policy based management approaches have effectively followed an approach in which a management system provides relatively constrained instructions to a managed device, without leaving much room for autonomic behavior for the device. While some aspects of autonomic behavior is shown when managed devices and management systems are treated as a single unit, current policy based systems do not permit significant autonomic behavior at the level of the managed device. In this paper, we present an architecture which can let the managed device obtain more autonomic behavior, and illustrate its usefulness by means of an example.

In the next section, we define the high level goals and objectives for autonomic behavior. We illustrate the autonomic behavior goal with a common security management situation. We then propose an approach which allows devices to generate their own policies. We demonstrate how the architecture can be applied for several autonomic management problems, with special focus on a problem focus on a problem for managing software defined assets (e.g. computation, storage, networking) across multi-national coalitions. In particular we wish to be able to flexibly bring together different coalition assets to achieve a combined capability that could not be achieved by a single coalition member. We refer to this as software defined coalitions [17] and describe it further in Section V.

II. GOALS FOR AUTONOMIC MANAGEMNT

A traditional management system can be modeled by the abstracted physical layout shown in Figure 1, which shows a set of managed devices connected to a management system by a communication network. The managed devices are given configuration information by the management system. Operational information (e.g. alerts or logs) is provided by the managed device to the management system. The management system would have a set of processing algorithms, e.g. policies and rules to deal with the set of alerts and logs that are processed. In order to handle the alert or log, the system may decide to send a reconfiguration command to the managed device. The syntax and semantics of alerts, logs and configurations are determined by the domain of management, such as fault management, security management, performance management etc. When the system is not able to deal with the alerts or logs by itself, the human operator intervenes to deal with the situation, diagnoses the underlying cause, and then reconfigures the system to react to the unexpected situation.

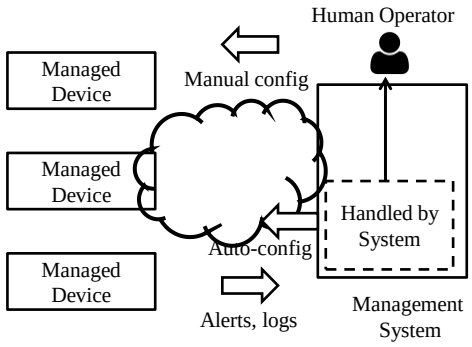


Fig. 1. Simplified model of systems management

Several technologies have been developed with the field of network and systems management to reduce the amount of

alerts that needs to be handled by the human operator. The existing state of the art for policy management provides one such approach. The typical architecture for policy management is shown in Fig 2. The human operator is provided an easy to specify objective for the managed device to use (human view of policy). The management system translates them into a machine view of policy through the process of refinement or transformations [2]. The machine view of policy is provided to entities known as policy decision points (PDP). Each managed device embeds a policy enforcement point which can consult a PDP and use the information provided by the PDP to reconfigure itself. When a situation requiring an alert arises, or any entry in a system log needs to be processed, the PEP converts it into a standard request that can go to the PDP. The PDP makes a decision on the appropriate situation and informs the PEP on how to handle the situation. The PEP can then change the system configuration to react to the environment.

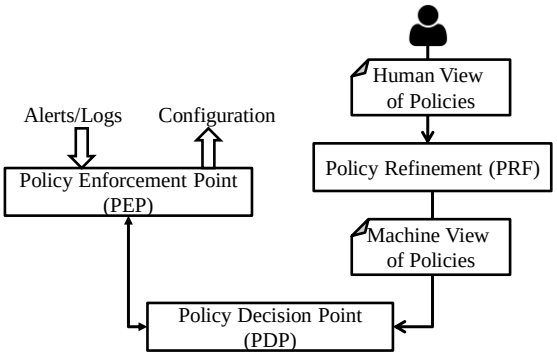


Fig. 2. Policy based Management Architecture.

The policy management system, PDP and PEP make up the logical constructs which can then be mapped into the physical layout of managed devices and management system is a couple of different ways, as shown in Figure 3. The PEP is usually embedded as a component of the managed device, while the policy refinement process is embedded as a part of the management system. The PDP could be embedded either as a part of the management system, or it could be embedded within the managed device. When the PDP is embedded within the managed device, the device exhibits limited autonomic behavior, by operating according to the policies defined by the management system. .

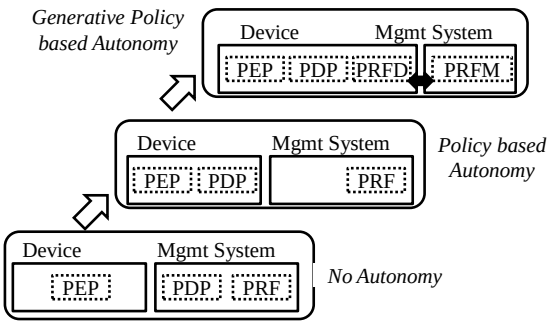


Fig. 3. Generative Policy. based management

As mentioned earlier, the current model for policy based management has the semantics where the machine view of policies are determined by the refinement process, and the managed device with the embedded PDP has no ability to define its own policies. Our goal is to enable a situation where the managed device can derive its own policies, which means embedding a part of the policy refinement process within the device itself. Embedding policy refinement within the device has the most significance when modular design principles are used, enabling the same refinement process within the device to be used across multiple management domains. Within the context of coalition operations, where a device belonging to one coalition partner may need to interact with the management systems of another partner, policy refinement within the device increases the autonomy and responsiveness of the managed device.

In Fig. 3, the refinement component embedded in the device is shown as PRFD, while the refinement component within the management system is shown as PRFM. PRFD uses refinement algorithms that can work without requiring a human input, since the PRFM in the management system can be improved significantly with interactive human input. The semantics of the interaction PRFM and PRFD, shown by the black arrow in Fig 3, is a key component that enables this stage of autonomy. Our goal for autonomic management is to enable devices to reach the Generative Policy based Autonomy stage shown at the top of Fig 3.

We call our approach a generative model for policies, since policies are generated by the device itself. Generative policies allow a flexibility and freedom of action for the managed device to determine their own behavior characteristics. The current model of policies used in the state of the art is very prescriptive in that it defines the actions to be taken under various conditions, and does not permit much freedom to the managed device.

The broad approach for generative policy based management can be illustrated in an intuitive manner using an example from a common security management situation in data centers and cloud sites. This example is discussed in the next section.

III. Illustrative Problem Scenario

Maintaining secure access to documents is a common problem in any data center/cloud site. We can consider a situation where we have a set of documents, some of which are considered sensitive, and others that are not. A set of users have access to sensitive documents, which can be accessed either using a web-based application or via a secure shell based system. In order to prevent attacks on the systems, a packet filtering firewall is provided to safeguard access to both the web-based system and the secure shell based system. The configuration is shown in Fig 4. The scenario is a common one encountered in almost any site requiring access control to a set of documents.

The current practice in securing such systems is for a human administrator to manually configure filtering and access control policies for the firewall, web-server, secure shell server and the document server. In a typical scenario, the ports on a

firewall need to be configured to allow access to the web server. However, if we provide a higher degree of autonomy to the web-server, e.g. assume it is using a moving target defense [18], and changes its port for the web-server at some regular periods, the configuration of the packet filter firewall needs to be repeated manually every-time such a change happens. The management tool will have to create the appropriate policies for each of the devices.

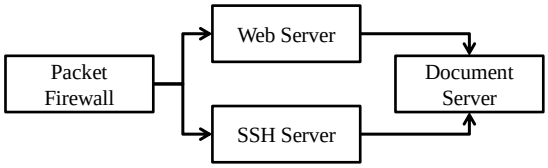


Fig. 4. Illustrative Problem Scenario

Instead of a manual reconfiguration of access controls after each change, it would be highly desirable if the human operator (the human view of policies) simply specified the access requirements on the documents. Based on those access control requirements, the packet firewall, the web server, the SSH server, and the document server would each derive their policies to comply with the desirable policies on their own. If the web-server switched its port as part of the moving target defense, the packet firewall would automatically adjust its filtering policies accordingly. In these cases, the policy refinement process happens within the devices themselves, and does not require any human intervention until access control on specific documents are changed.

Note that a similar policy refinement will be desirable if the devices were to be managed not for access control security, but also for fault management, configuration management or performance management. When a fault happens in one of the components, we would desire all the components impacted by the fault to generate new policies to deal with that fault. Similarly, for performance management, the different components should be able to determine their policies for managing performance, e.g. by increasing the number of parallel instances, etc. In all aspects of autonomic behavior, we would like the devices and components to define their policies on their own.

In the next section, we present the general solution for generative policies.

IV. Generative policy architecture

In the generative policy architecture, the PDP is embedded within the managed device, and it gets its policies from the PRFD module that is also embedded within the managed device. The PRFM is responsible for sending the overall coordination guidelines to the PRFD. The architecture overall is as shown in Fig 5.

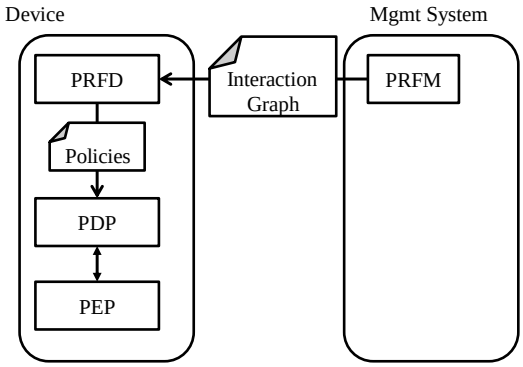


Fig. 5. Generative Policy Architecture

The PRFM provides two types of information to each PRFD. One is the representation of an interaction graph. An interaction graph is an abstract description of the various entities within the environment that the PRFD needs to interact with. The interaction graph is defined as a relationship between entities in different roles in the system, not as an exhaustive listing of all the different devices in the system. The role of each PRFD in the interaction graph is defined by the PRFM. The PRFM also associates a resource model, i.e. a set of attributes with each link in the interaction graph.

Each PRFD receives the interaction graph from the management system, and uses it to generate the policies for its local PDP. Each PDP discovers other PDPs in the system it should connect with according to the interaction graph, and finds out the attributes of the connecting links. In order to generate these policies, the PRFD uses a policy grammar that defines the syntax of the policies that can be generated locally. The grammar is used to generate expressions that are defined over an alphabet that includes the local attributes of the managed devices and the attributes of the links that the managed device has with other devices as defined in the interaction graph. As the managed device discovers other devices in the roles identified in the interaction graph, new policies conforming to the grammar and including the attributes of the newly discovered links are generated.

The grammar can be provided to the PRFD by an independent management system. As an example, a U.S. operator may provide the grammar to its drones, while in the context of a coalition operation, the management system PRFM may belong to another coalition partner. For each distinct domain to which the policy architecture is applied, the set of valid roles, the attributes which define the mapping of the nodes in the interaction graph to the terminals in the policy grammar, and the specific policy grammar are specified.

Let us consider the application of this architecture to the illustrative problem scenario. In this scenario, three roles for different entities can be identified, a network protection role(N), a protocol protection role(P), and a document protection role(D). In the scenario instance shown in Fig 3, two devices are in the role of protocol protection, while only one device is in the other two roles. The global interaction graph is shown in Fig 6 with the letters N, P and D indicating the different roles, and the attributes required for each link. The specific interaction graph that will be provided to the PRFD in

each of the devices is also shown, with the role of the device marked in black circles in each device specific interaction graph.

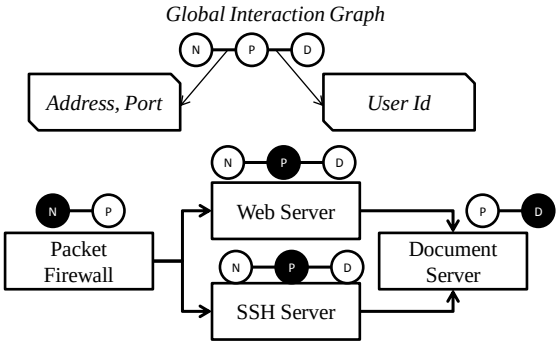


Fig. 6. Interaction graphs for the Illustrative Problem

When the PRFD for each of the devices receives the interaction graph, it searches for the other nodes that are associated with adjacent roles in the interaction graph. The discovery module finds out the other devices in those roles, and finds out the attributes identified by the devices in those roles in the interaction graph. Then, the PRFD uses the grammar available with it to generate its own set of policies to be used for its PDP.

In the illustrative scenario, once the address and port for the protocol protection role are identified, the packet firewall can generate the appropriate packet filtering policies for the firewall. Similarly, the web-server can receive the set of user-ids that are authorized to access the document server, and install its protection policies to only allow those user-ids to access the document server. Note that the document server will provide several user-ids to the protocol protection servers, while each of the web-servers and SSH server will only provide a single network address and port for itself to the packet firewall. Eventually, the document server would need to have its document protection policies. In this case, these policies come directly from the PRFM.

The grammar for policy generation for the packet filter firewall will generate the 5-tuple (source and destination addresses, source and destination ports, and the protocol) that determine whether or not a packet is allowed within the network. The address and port numbers of each protocol protection device (the web-server and SSH server in this scenario) will provide this information to the firewall which can use it to create its network packet filtering policies. When the attributes for the web-server and SSH-server change, e.g. due to a scheme such as moving target defense [18] changing those attributes, the discovery process will again be triggered and the policies in the packet filtering firewalls will be regenerated.

While the previous example was for access control, the same scheme can be generalized for other aspects for the same system setup, e.g. increasing the number of instances of each node in Fig 4 dynamically for performance management, or for a node to dynamically restart the other node for resiliency management. When nodes are dynamically instantiated using technologies like NFV or SDN, roles can also be dynamically assigned to different instances running in the system.

In the next section, we describe a common situation that arises in coalition operations, and how the generative policy architecture can be used in that situation.

V. Software Defined Coalitions

In coalition operations, e.g. when joint operations need to be undertaken by soldiers belonging to different countries such as those belonging to an alliance like NATO, it is frequently essential to establish a dynamic community of interest. A dynamic community of interest (CoI) is a group of people that come together for a brief period in order to perform a specific task, and the group dissolves after a specific time has elapsed or the mission has been completed[19]. Such dynamic communities of interest may also be formed in non-military contexts, e.g. when different civilian agencies come together to fight a fire or deal with the aftermath of a strong hurricane. In a dynamic CoI, not all members are trusted equally, e.g. in a coalition formed between U.S., UK and a third hypothetical country of Holistan [20], U.S. and UK are more likely to trust each other than the members of Holistan.

When dynamic CoIs are formed, they require supporting IT infrastructure to conduct their operations more effectively. The assets from the CoI can come from all the different coalition members, and they need to interoperate together in order to provide the most effective IT support. Software Defined Coalitions (SDC) is an approach to provide such support, and it combines/extends concepts from software defined networking [21] to operate in the context of coalition operations. A more detailed description of the software defined coalitions can be found in [17]. The scope of a software defined coalition may vary from a simple sharing of assets to creating a seamless virtualized infrastructure across all assets, but it is the former that we will be exploring further in this paper for its autonomic management needs.

The assets used in such a dynamic CoI would be capable of a significant processing power, as well as freedom of actions. These assets include drones, self-driving cars, and autonomous mobile robots of different sorts, as well as static cameras and sensors which are capable of substantial processing power. Each of these assets is capable of running the PEP, PDP and the PRFD components of the general architecture, and take decisions on their own.

During the operations of the CoI, regardless of the specific actions or missions being conducted by the coalition, several management issues can arise. Assets may be subject to breakdown, subject to security attacks, and need to deal with unexpected situations.

Let us consider the situation where some drones and robotic mules from U.S. and UK are part of a coalition mission where the mission commander is from Holistan. In order to conduct the mission efficiently, the drones have to comply with the commander's instructions. At the same time, the two countries may not want to provide complete control of operations to the commander who is only partially trusted, and may be concerned about the safety of their assets during operations. In addition to ensuring the safety of their drones,

some routine aspects of management need to be handled. The drones involved in a mission need to be provided the policies so that they can securely communicate to other assets in the formation. Faults arising during the operation need to be handled autonomously, and different security threats have to be handled appropriately.

In the next section, we look at how the general architecture described in Fig 5 can be used to handle some of the issues in autonomic management in the context of a coalition SDC.

VI. Autonomic Management in a SDC

We examine three management issues associated with SDC, namely the task of providing the right authorization credentials to the assets, having the assets control their movement during the operations, and how the assets can handle abrupt failures during the conduct of a mission.

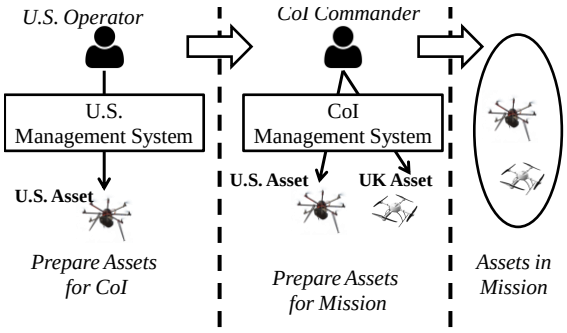


Fig. 7. Different Management Systems in a SDC supporting a dynamic CoI

The assets in the SDC are subject to dual management, and have to be able to deal in an autonomous manner with the instructions and commands from both types of managers. Furthermore, for a significant part of the actual operation, assets may have to work independently in a mode where they may be disconnected from their respective managers. A partial setup for these operations is shown in Fig 7. The U.S. asset (drone) that has to participate in the SDC for the CoI initially needs to be prepared for participation by the U.S. operator using a U.S. management system. Analogously, the UK operator would be preparing the UK drone, and any other coalition partner would prepare the corresponding assets. Once the assets are with the CoI commander, the CoI commander needs to prepare them for the mission. At this stage, the asset may only have connectivity to the commander, and have no connectivity to the U.S. management system. Once the commander has prepared the assets, which may belong to more than one nation, the assets perform the actual mission, e.g. surveillance of some geographical area. During the period of the mission, the assets may be operating independently without necessarily having a connection to any management system.

A. Autonomic Authorization

When the dynamic CoI is being formed, the country operator has to make sure that the asset being given for the CoI can accept configuration and management commands from the CoI commander. At the same time, the operator may want to put in some constraints on how the asset may be used. When the asset is being provided to the CoI, the identity of the commander may not even be known. Nevertheless, the ability to enable the CoI commander to provide instructions to the asset needs to be provided.

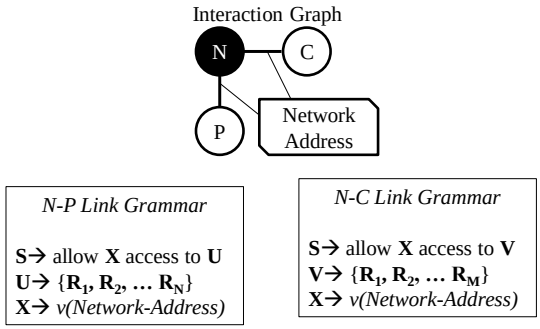


Fig. 8. Interaction Graph for Authonomic Authorization. N indicates the role of a node (self), P the role of a peer U.S. node and C the role of a coalition commander.

During the operation of a SDC, several peers may connect or try to connect to the asset. The authorization of any peer has to be done in two stages, a connection stage and an authorization stage. For the connection stage of the asset, let us assume that the typical shared secret approach is used. Anyone connecting to the peer has to present the proof that they know a shared secret, which can be implemented via a variety of mechanisms, e.g. using a zero-knowledge proof [22]. The shared secret is predefined and provided to all of the assets involved in the CoI, and only those assets are able to connect to each other. Note that different secrets may also be used to authenticate a peer asset from the same country versus a peer asset from another coalition partner.

The next stage of authorization for services can use the generative policy approach. In order to prepare the asset to be used in the CoI, the U.S. operator will provide an interaction graph containing three roles to the U.S. asset, the role of the mission commander and the role of a U.S. peer asset, in addition to the role of the asset itself. Assuming that network addresses are used for access control and authorization, each of the links is associated with a resource model that allows them to identify the network address of the peer on the link. Suppose the mission commander is allowed access to some M resources (a resource could be a service or a capability on the drone), and any peer U.S. node is allowed access to some N resources, a possible semi-formal grammar to generate access control for peers and mission commander will be as shown in Fig 8. In the Figure, the notation v(attribute) is used as the value of the attribute in the link that is dynamically discovered.

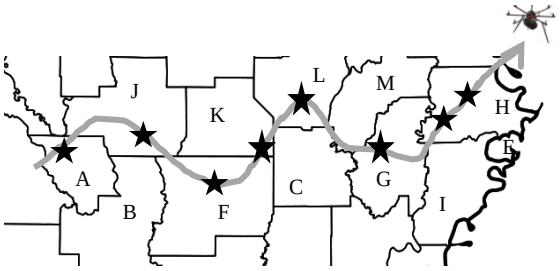
In this model, as different assets connect with the subject peers by providing the right shared secret, the access control policies for each new connected peer can be generated by the asset automatically for its use. In some of the missions, the asset may need to opportunistically connect to peers in other roles, and can derive policies for such interactions and sharing. As the mission evolves, participants in the mission change, and higher level policies change, the policies for the operation of the assets can be derived automatically.

B. Movement Control

One of the constraints the U.S. forces may have on their drones is that they only be operated in areas considered safe for them. UK forces may have their own constraints on the operation of their drones. When the drones are given over to a commander from a separate country, the two countries may still want the drones to conform to their policies.

The interaction graph used will still be the same as that for autonomic authorization, which will allow the mission commander to set paths for the asset. However, in addition to the access control, a path validation constraint is also provided to the asset. The path validation constraint is used by the drone to check whether the path provided by the commander, using a set of waypoints, is acceptable to the drone.

The waypoint approach is a common method to control the flight of autonomous drones. In this approach, the path of a drone is determined by means of a set of predefined waypoints which the drone follows to complete a path. In the scenario we are discussing, the CoI commander will define a set of waypoints which can be used to instruct the drones how they should fly for any specific mission.



Path = AJFKLMGHH

Fig. 9. A sample path for drones specified using waypoints

In the architecture we are proposing, a grammar can be used by the U.S. (or UK) commander to provide a constraint on what are acceptable paths for the drones for a coalition mission. Each of the locations on the map can be mapped into one or more geographical areas with a name, turning the waypoint specification into a string formed by the names of the areas. The grammar would then define which of such strings are acceptable for the drone. Fig 9 shows a typical waypoint defined path, and the string which corresponds to that path.

As an example of the control of drone path, the UK may have a policy which requires them to avoid any of the areas that are considered unsafe. The U.S. may be more permissive, but require that no more than two consecutive waypoints be in unsafe areas. Assuming that there are six regions A-F of which A,B,C are safe and three regions D,E,F are unsafe, the U.S. and UK grammars could be as shown in Fig 10. By specifying the grammar, the country operators can provide an approach for a coalition drone to validate the paths provided to it by the mission commander, and negotiate a path which allows them to conform to their country policies with the mission commander.

In the above scenario, multiple drones may be in operations concurrently for a mission. When multiple drones belonging to the U.S. are provided the paths, they can validate each other's paths, and even mutually exchange information with each other of collected surveillance information using the generative policies for authorization defined in the previous section.

U.S. Movement Grammar

S→ X | SY
A→ {A, B, C}
U→ {D, E, F}
X→ XA
Y→ UUA | UA

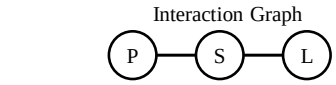
UK Movement Grammar

S→ SA
A→ {A, B, C}

Fig. 10. Grammar for Movement Control

C. Autonomic Fault Management

The architecture proposed in Fig 5 can be used in turn by the mission commander to enable coalition operations in an autonomous manner by a fleet of drones. Suppose the commander has a set of drones that it wants to fly over a region and look for suspicious elements, e.g. a camouflaged weapon or an insurgent base. It could assign a swarm of drones to fly over the region with the responsibilities split among different drones. A set of scout drones would fly quickly over a region to assess that it is safe, followed by a set of surveillance drones which will make a slow but more thorough pass over the area. The coordination of duties between different scouts and surveillance drones will be done by means of a leader drone which has plans for various evasive manners. A drone is assigned the task of being the backup leader in case the normal leader encounters a problem.



P-S Link Grammar

S→ allow **X** access to **U**
U→ {R}
X→ *v(Identity)*

S-L Link Grammar

S→ allow **X** access to **U**
U→ {Q, T, V}
X→ *v(Identity)*

P-L Link Grammar

S→ allow **X** access to **U**
U→ {Q, V}
X→ *v(Identity)*

V – Role assignment service
R – Route Status service
Q – Route Control service
T – Surveillance Status service

Fig. 11. Interaction Graph for Authonomic Authorization

This interaction graph provided by the mission commander to the different drones to enable this scenario is shown in Fig. 11. The system assumes that each drone has four types of services, the role assignment service enables the assignment of roles to drones for a specific service, the route status service allows a peer to check on the status of an area checked via a quick scouting pass, the surveillance status service allows a drone to check for results of a more thorough surveillance, while the route control service allows a peer to dictate the drone how to route itself.

A scout will have the role assignment service, route control service and the route status service, with the leader having access to the role assignment service and the route control service; while the surveillance drones will have access to the

route status service. A surveillance drone will have the role assignment service, route control service and the surveillance status service, all accessible by the current leader. The leader would have an algorithm deciding on the roles that each drone will have in the surveillance mission.

Assuming that the mission commander has provided the right shared secrets for the drones in the mission to communicate with each other, the drones can conduct the mission with the leader planning out the routes for the other drones. In the case of a failure of a node, e.g. the leader develops a fault and the backup takes over, the discovery process and policy generation process in each drone will create appropriate policies for the new role assignment among the remaining drones. The drones can enable appropriate access control on their own even if they are disconnected from the mission commander for part or even the whole of the operation.

The three aspects described above are not the only aspects of management required in a system. However, the general architecture embedding part of policy refinement within the management device, the concept of the interaction graph, and the use of grammars to generate policies results in a general approach which can be extended to other domains.

VII. CONCLUSIONS

In this paper, we have examined the problem of enabling autonomic behavior in managed devices, and enabling them to generate policies for their operations on their own. The policies are generated for each domain according to an interaction graph which is provided by a management system, and which defines the scope of activities for the devices. We have given several examples of how these architecture can be used to generate policies and achieve self-management in many different contexts. This architecture provides a significant advancement over the current state of the art in policy based management.

While the approach and the architecture provide a promising approach for autonomic behavior, several new directions need to be explored to make the architecture truly useful. Algorithms for checking the validity and effectiveness of generated policies need to be developed to the framework. We need to apply the framework to several other system management scenarios, and validate that this will address autonomy challenges in those scenarios.

VIII. ACKNOWELDGEMENTS

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, 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 copy-right notation hereon.

REFERENCES

[1] S. Illner, A. Pohl, H. Krumm, I. Luck, D. Manka, and F. Stewing. "Policy-based self-management of industrial service systems." Proc. of 4th IEEE International Conference on Industrial Informatics, pp. 492-497, 2006.

[2] D. Verma,"Simplifying network administration using policy-based management." IEEE network vol. 16, no. 2, pp. 20-26, 2002.

[3] R. Bhatti, A. Ghafoor, E. Bertino and J. Joshi, "X-GTRBAC: an XML-based policy specification framework and architecture for enterprise-wide access control", ACM Transactions on Information and System Security (TISSEC), vol 8, no. 2, pp.187-227, 2005.

[4] M. Maullo and S. Calo, "Policy management: An architecture and approach", Proceedings of the IEEE First International Workshop on Systems Management, pp. 13-26), 1993.

[5] A. Ganek, A. Nadalin, N. Nagaratnam, and D. Verma. "An autonomic approach for managing security and identity management policies in enterprises." Journal of High Speed Networks vo. 15, no. 3, pp. 291-300, 2006.

[6] D. Agrawal, J. Giles, K. Lee, K. Voruganti, and K. Filali-Adib. "Policy-based validation of SAN configuration." In Policies for Distributed Systems and Networks, Proceeding of Fifth IEEE International Workshop on Policies, pp. 77-86, 2004.

[7] D. Verma, G. Cirincione, and T. Pham. "Policy enabled interconnection of sensor networks using a message queue infrastructure." Proc. of SPIE Defense and Security Symposium, 2008.

[8] S. Keoh, N. Dulay, E. Lupu,K. Twidle, A. Schaeffer-Filho, M. Sloman, S. Heeps, S. Strowes, and J. Sventek, "Self-managed cell: A middleware for managing body-sensor networks. In Mobile and Ubiquitous Systems: Networking & Services", Fourth IEEE Annual International Conference on Mobile and Ubiquitous Systems, August 2007.

[9] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen. Policy Core Information Model--Version 1, IETF RFC 3060. 2001.

[10] N. Damianou, N. Dulay, E. Lupu and M. Sloman. "The ponder policy specification language." In Policies for Distributed Systems and Networks, pp. 18-38, 2001.

[11] K. Twidle, N. Dulay, E. Lupu, and M. Sloman. "Ponder2: A policy system for autonomous pervasive environments." Proc. of Fifth IEEE International Conference on Autonomic and Autonomous Systems, pp. 330-335, 2009.

[12] S. Godik, A. Anderson, B. Parducci, P. Humenn and S. Vajjhala, OASIS eXtensible access control 2 markup language (XACML), OASIS Technical Report, 2002.

[13] A.Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni and J. Lott, "KAoS policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement". Proc. of IEEE Policies for Distributed Systems and Networks, pp. 93-96, 2003.

[14] D. Agrawal, K. Lee and J. Lobo, Policy-based Management of Networked Computing Systems, IBM Research Report TC23685, August 2005.

[15] W. Han, and C. Le, A survey on policy languages in network and security management. Computer Networks, 56(1), pp.477-489., 2012

[16] M. Huebscher, and J. McCann, A survey of autonomic computing—degrees, models, and applications, ACM Computing Surveys, vol 40, no. 3, August 2008.

[17] V. Mishra, D. Verma, C. Williams and K. Marcus, Comparing Software Defined Architectures for Coalition Operations, submitted to International Conference on Military Communications and Information Systems, 2017.

[18] S. Jajodia et. al., eds.: Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats, Advances in Information Security, Springer, 2011

[19] E. Asmare et. al., "Secure Dynamic Community Establishment in Coalitions," IEEE Military Communications Conference, IEEE MILCOM 2007, Orlando FL, Oct 2007.

[20] D. Roberts, G. Lock, and D. Verma. Holistan: A futuristic scenario for international coalition operations. In Proceedings of Knowledge Systems for Coalition Operations, 2007

[21] Nunes, Bruno Astuto A., et al. "A survey of software-defined networking: Past, present, and future of programmable networks." IEEE Communications Surveys & Tutorials 16.3 (2014): 1617-1634.

[22] O. Goldreich and Y. Oren, Definitions and properties of zero-knowledge proof systems. Journal of Cryptology. vo 7, no 1, pp 1-32, 1994.