Distributed Intelligence – Trends in the Management of Complex Systems

Seraphin B. Calo, Dinesh C. Verma
IBM Research
Yorktown Heights, NY, USA

Elisa Bertino
Purdue University
West Lafayette, IN, USA

ABSTRACT

The ability to incorporate intelligence in even small devices and to make use of contextual information from widely deployed sensors has already begun to change management paradigms. As edge computing and IoT become more prevalent, systems will increasingly consist of cooperating, heterogeneous, distributed, autonomous elements. Architectures for cognitive, collaborative systems are evolving to deal with such complex environments. Concepts from multi-agent systems and autonomic computing are being applied to cope with the scope and breadth of large collections of interacting devices and services. Technologies for security and access control must evolve as well. Policy-based mechanisms are widely used and have been very successful in protecting information and controlling access to systems and services. They tend to rely, however, on a centralized infrastructure and on the automated enforcement of directives. Newer paradigms are being investigated that allow policy structures to be more dynamic and contextual, while still preserving the desired levels of control. We will present trends in the evolution of architectures for distributed, federated systems, and the technologies for managing them.

INTRODUCTION

The Internet of Things (IoT) has become increasingly important due to its great potential for providing ubiquitous services based on real time contextual information. It represents not only the use of simple sensors and actuators, but also the application of sophisticated software and complex systems to data-driven processes. Thus, there is the opportunity to evolve interoperable networks of devices and systems producing large amounts of data and providing a plethora of distributed services.

One of the technologies being explored for overcoming many of the problems being faced in the development of an infrastructure for IoT environments is that of Multi-Agent Systems (MAS) [1], which are based on distributed intelligence and an architecture for supporting cooperating components.

SACMAT'17, June 21-23, 2017, Indianapolis, IN, USA

ACM ISBN 978-1-4503-4702-0/17/06...$15.00
http://dx.doi.org/10.1145/3078861.3078881

Independent software agents represent the goals of their actual counterparts (e.g., sensors, hardware devices, operating software) and take actions autonomously. These agents form collaborative units to execute complex processes. They can incorporate learning mechanisms and cognitive analytics. Such agent based approaches offer a promising alternative to more centralized or hierarchical architectures for IoT software that have difficulties coping with the management of huge numbers of devices and the volumes of data that they produce.

Another key technology for dealing with distributed, autonomous systems is that of policy based management. It has proven to be an invaluable element in simplifying the management and operations of complex distributed environments in many different domains. Due to their wide applicability, several policy management frameworks have been developed, and many policy specification languages and information models have been defined [2].

Despite the benefits and usefulness of policy based management, currently deployed technologies have followed an approach in which the management system provides relatively constrained instructions to a managed device. These instructions are typically specified by a central authority and are meant to be explicitly followed by all elements to which they apply. However, in distributed systems, the local context may change or there may be some uncertainty in the values of certain attributes of the system. The directed policies may thus no longer apply or their applicability may have become ambiguous. Further, they incorporate no cognitive mechanisms for learning from past experiences. A policy management system that allowed local modifications or adaptations of globally defined policies might thus be better able to effectively support local autonomy.

In the next section, we present some thoughts on the architecture needed for building cognitive collaborative systems. Then we describe the technologies currently being applied to policy based management, and some extensions being explored for adaptive policy inferencing. Next, we present the overall generative policy approach that further extends the functionality of such systems to include the influences of local context. This would significantly advance the attainment of the capabilities needed for the operation and management of collaborative, autonomous systems. We illustrate the use of generative policies with examples in the two subsequent sections. Finally, we state our conclusions and mention some future investigations.

COGNITIVE COLLABORATIVE SYSTEMS

In order to accommodate the huge numbers of objects that will be part of the Internet of Things (IoT) a suitable management paradigm with appropriate technological foundations must be developed. Internet-connected sensors, actuators and other types of smart devices need an infrastructure for association and control if they are to be coordinated effectively to provide situational awareness and actionable intelligence. The management functionality must overcome the dynamic, heterogeneous, distributed nature of such systems, while providing high reliability. The idea is to enable seamless and interoperable connectivity among heterogeneous devices and systems, while hiding their complexity and providing sophisticated services.

A promising direction for the organization of such complex environments is to make the constituent elements of the system self-managing, and to create an architecture where knowledge mechanisms are tightly coupled with the computing elements. We imagine a collection of autonomous operational units that collectively form a collaborative system. There are several related technologies that have been proposed, and their characteristics are being investigated. These include: Multi-Agent Systems (MAS) [1], Self-Managed Cells (SMC) [3], Autonomic Managers (AM) [4], and the infrastructure developed for Mobile Edge Capture and Analysis (MECA) [5]. Our intent is to also include more recent advances in cognitive computing and generative policy management.

At a high level, the characteristics of the elements of the system (hardware, software, datasets, etc.) would be captured in Virtual Objects (VOs) that would represent them within the software environment. The VOs would track the status of the assets they represent, and would be able to communicate with them and control them to the extent that the actual asset can be externally controlled. Composite Virtual Objects (CVOs) would represent collections of semantically interoperable Virtual Objects (VOs) that provide federated information services.

Autonomous Operational Units (AOUs) would represent collections of hardware and software assets that jointly can perform an operation or process. The AOUs include a management system with cognitive capabilities, and maintain rules, process flows and policies associated with their operational domain as in Figure 1. They can be goal driven, discovering VOs and other AOUs that can provide necessary capabilities for carrying out their responsibilities. They can predict the expected behavior of physical assets, and would incorporate learning mechanisms for improving their effectiveness and performance.

The AOUs would maintain relationships with other AOUs, so that their collection forms a (logically) distributed collaborative system. These collections would establish patterns of behavior, provide mechanisms for continuously optimizing the functioning of their associated physical systems, and would enforce complex policies and workflows associated with multiple interacting organizations, e.g., coalitions. They are self-organizing and self-configuring, coming together to accomplish cooperative tasks.

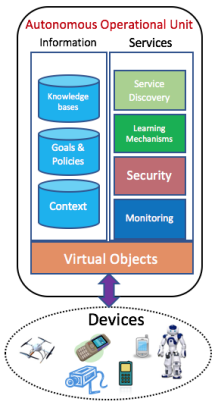

Figure 1: Autonomous Operational Unit

POLICY BASED MANAGEMENT

Policy Based Management Systems (PBMS) typically follow the policy model defined by the Internet Engineering Task Force (IETF) and the Distributed Management Task Force (DMTF) [6]. This framework consists of four basic elements: a policy manager (PM), a policy repository (PR), a policy enforcement point (PEP), and a policy decision point (PDP). Policies themselves are defined as declarative event-condition-action (ECA) rules. A Boolean condition is defined over a set of run-time data provided to the policy at evaluation time by the managed environment. When an event triggers the evaluation of the condition, if it evaluates to *true*, then the action is executed. Policy management frameworks based on this model may provide additional capabilities that make it easier for system administrators to policy enable their applications, middleware, and services.

The human operator or policy administrator specifies objectives for the functioning of the managed devices. The management system translates these higher level policies into a machine view of policy through the process of refinement or policy transformation as in Figure 2. The machine view of policy is provided to entities known as policy decision points (PDPs) either directly or through a policy repository. Entities known as policy enforcement points (PEPs) represent managed resources with respect to particular decisions. When a PEP needs to make one of these decisions, it calls out to its associated PDP, which obtains the relevant policies, interprets them and communicates appropriate actions back to the PEP. This allows decisions to be made based on the current state of the system and the active set of policies, rather than being hard coded in the application. The behavior of the system can then be changed by changing the policies rather than rewriting or extending the software code itself.

The elements of the policy infrastructure can be configured in several different ways, depending upon the desired architecture of the system under management. The PEP is associated with the managed resources and will usually be co-located with them, while the policy refinement process is usually embedded in the policy management tool. The PDP could be embedded in the policy management tool in systems that are meant to be centrally managed, or it could be co-located with its associated PEPs in systems that are meant to be distributed. In the latter configuration policies would be pushed to appropriate PDPs when they are

specified or changed, and stored locally. When the PDP is embedded within the managed environment the managed system can exhibit a greater degree of autonomic behavior, since it does not have to go back to a central point to determine which of a set of alternative actions should be taken at specified decision points.

Beyond methods for policy specification, mechanisms for learning applicable policies within the confines of higher level constraints are being investigated. There are efforts to develop techniques for policy mining that serve to: automatically learn security policies from historical examples and granted exceptions; and, infer roles that help compact and simplify security management. Earlier work used data logs to infer policies directly from system usage [7]. This allows the finding of RBAC models which reflect the observed usage of entitlements and the attributes of users. These models are thus causally associated with actual usage of entitlements and combinations of user attributes.

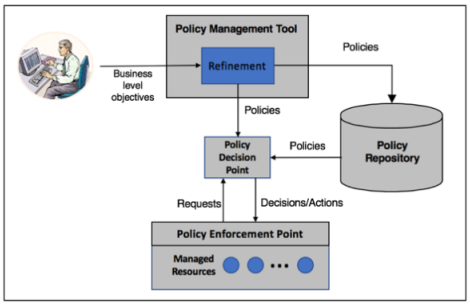

Figure 2: Policy Based Management System

Continuing research applies Behavioral Analytics for anomaly detection to derive policies for operational control [8]. Such techniques, as well as classical machine learning techniques like transfer learning are also being investigated. In transfer learning, the knowledge gained from the creation of policies in one context can be stored and applied to the learning of policies in other related contexts.

A general architecture for dynamic policy adaptation is shown in Figure 3. The policy decisions at various PDPs would be monitored and that knowledge would be used to modify the deployed policies, as well as to provide input to the refinement process so that it could be leveraged in the specification of subsequently defined policies. Only relatively simple instances of the general approach have so far been studied, mostly with respect to the incorporation of history data [9].

Generative policy architecture

A key concept in the generative approach is that local elements will generate their own operational policies within the bounds of higher level policy structures supporting collaboration and meant to assure compliance with high level constraints and the pursuit of common goals. The formal model of a generative PBMS must thus provide constructs for specifying both the higher-level policy structures, the operational policies that are locally generated, and the relationships between the two.

In the generative policy architecture, the policy refinement process (PRF) is separated into two parts, one associated with the global management system (PRFM) and one associated with the managed system (PRFD). The PDP is embedded within the managed system, and it gets its policies from the PRFD module as shown in Figure 4. The PRFM is responsible for sending the overall coordination guidelines to the PRFD. The PRFM provides two types of information to each PRFD. One is an interaction graph, that contains an abstract description of the various entities within the environment with which the PRFD needs to interact.

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 set of attributes with each link in the interaction graph. These are the attributes whose values can be obtained via the indicated link.

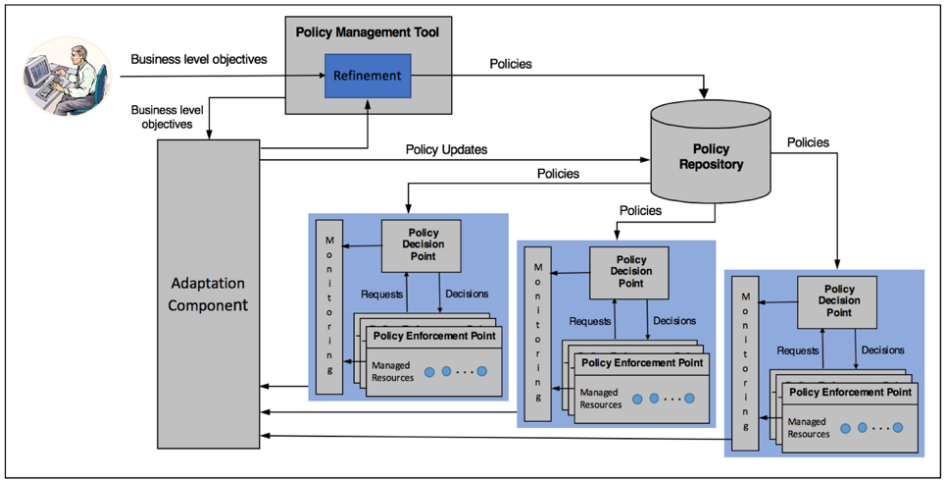

Figure 3: Dynamic Policy Adaptation

Each PRFD receives the interaction graph from the global management system and uses it to generate the policies for its local PDP. This provides each PDP with the attributes of the connecting links that can then be resolved by the policies that require them. To generate these policies, the PRFD can use different mechanisms. One approach is that of utilizing a grammar (either a Context Free Grammar, CFG, or an Attribute Grammar) for capturing the set of allowable policies, and having the distributed elements generate and employ operational policies only if they are derivable from that grammar. Another approach would be to specify a State Machine that indicates the expected operation of the managed system from the point of view of the global management system. This description would typically indicate certain goal states that the local systems should achieve. A third alternative would be for the PRFD to receive policies at a higher level in terms of more abstract concepts, and utilize a refinement hierarchy to produce operational policies pertinent to the local context.

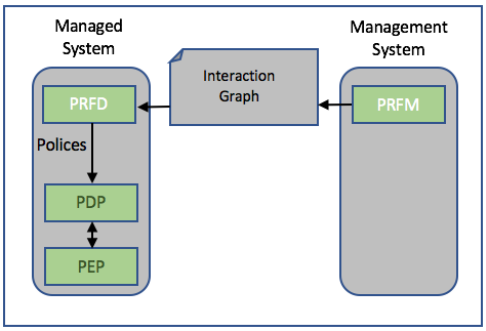

Figure 4: Generative Policy Architecture

Such policy mechanisms can also be provided to the PRFD by an independent management system. As an example, in a military coalition, a U.S. operator may provide the mechanism agreed to for a particular mission to the drones under its control, while in the context of the 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 values referenced in the policies, and the specific policy structures are specified.

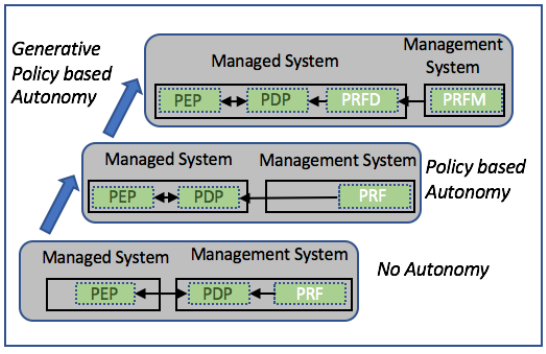

Figure 5: Evolution of Policy Based Management

As mentioned earlier, the current model for policy based management has the semantics that the machine view of policies is determined by the refinement process, and the managed system with the embedded PDP has no ability to define its own policies. Our goal is to enable a situation where the managed system 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. Our view of the evolution of policy based management is depicted in Figure 5.

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 as discussed in the next section.

ILLUSTRATIVE PROBLEM SCENARIO

Maintaining secure access to documents is a common problem in any data center/cloud site. We consider a situation where we have a set of documents, some of which are treated as sensitive, and others that are not. A set of users have access to sensitive documents, which can be obtained either using a web-based application or via a secure shell based system. A packet filtering firewall is provided to safeguard access to both systems. The configuration is shown in Figure 6. 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 more autonomy to the web-server, e.g., assume it is using a moving target defense [10], 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.

Instead of a manual reconfiguration of access control policies after each change, it would be highly desirable if the human operator 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 own policies to comply with those specifications. 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.

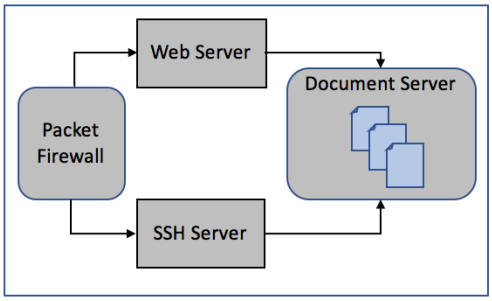

Figure 6: Illustrative Problem Scenario

Note that a similar policy refinement process would be desirable if the devices were to be managed not for access control, but for fault management, configuration management or performance management. When a fault happens in one of the components, we would want all the components impacted by the fault to generate new policies to deal with the fault situation. 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. In all aspects of autonomic behavior, we would like the devices and components to define their policies on their own.

Let us consider the application of the generative policy architecture to the access control scenario. In this case, 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 Figure 6, the web server and the SSH server both have the protocol protection role. The firewall has the network protection role and the document server has the document protection role. The global interaction graph is shown in Figure 7 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.

When the PRFD for each of the devices receives the interaction graph, it searches for the other nodes that are associated with adjacent roles. The discovery module finds 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.

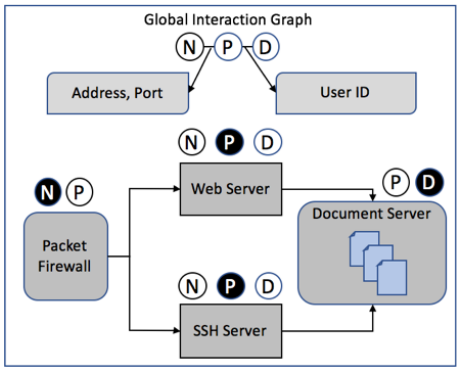

Figure 7: Roles and Interaction Graph

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 it can use 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 [10] 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 Figure 6 dynamically for performance management.

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

SOFTWARE DEFINED COALITIONS

In coalition operations, e.g., when joint missions need to be undertaken by soldiers belonging to different countries, it is frequently essential to establish a dynamic community of interest. A dynamic community of interest (CoI) is a group of individuals that come together for a specified period to perform a particular mission, and the group dissolves after a specific time has elapsed or the mission has been completed [11]. 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 hurricane. In a dynamic CoI, not all members are necessarily trusted equally, so policies related to information sharing may differ between groups of coalition members.

When dynamic CoIs are formed, they require supporting IT infrastructure to conduct their operations more effectively. The assets can come from all the different coalition members, and they need to interoperate. Software Defined Coalition (SDC) technologies provide such support. They combine and extend concepts from software defined networking to operate in the context of coalition operations. A more detailed description of software defined coalitions can be found in [12].

Let us consider the situation where some drones from the U.S. and the U.K. are part of a coalition mission where the mission commander is from a third coalition nation. 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.

The assets in the SDC are subject to dual management, and must 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. The U.S. asset (drone) needs to be prepared for participation in the CoI by the U.S. operator using a U.S. management system. Analogously, the UK operator would be preparing the UK drone. Once the assets are assigned, 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 other management systems. The assets belong to more than one nation, and together they perform the actual mission, e.g., surveillance of some geographical area.

Movement Control

One of the constraints the U.K. forces may have on their drones is that they can only be operated in areas considered safe for them. US 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 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. Figure 8 shows a typical waypoint defined path.

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. As a simple example, let us assume that there are six regions A-F of which three regions, A, B, C are safe and three regions, D, E, F are unsafe. Further, 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 requires that no two consecutive waypoints be in unsafe areas.

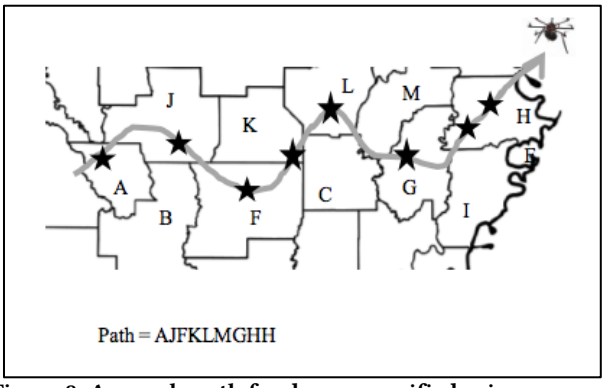

Figure 8: A sample path for drones specified using waypoints

In this example, we will consider an alternative way to send the high-level policy information. Instead of a grammar, the coalition commander will send the drone the region map and a corresponding transition diagram as in Figure 9, along with the characteristics of each region, in this case whether the region is considered "safe" = {a, b, c} or "unsafe" = {d, e, f}. It also indicates that the goal is to move the drone from an initial state s_0=a to an end state c∈F, following the conventions of a formal state machine.

A state machine is a quintuple (Σ, S, s_0, δ, F) where: Σ is the input alphabet (a finite, non-empty set of symbols); S is a non-empty set of states; s_0 is an initial state, an element of S; δ is the state-transition function: δ: S × Σ ⟶ S; and F is the set of final states. In this example, the states will be determined by the regions covering the mission theatre {a, b, c, d, e, f}. With $\lambda_x \in \Sigma$, x ∈ {a, b, c, d, e, f}, we assume that an event that issues λ_x will cause a transition to state x.

Each drone will have local context variables that were either set initially by their owning coalition member or reflect encountered operating conditions, **w** = {w_1, w_2, … w_R}. We assume that w_1 = identity of owning coalition member. The drone will calculate the set of possible trajectories without cycles: T = {abc, aec, abec, aebc, aefc, adfc, adec, aedfc, abefc, adfec, adebc, adefc, abedfc, adfebc}.

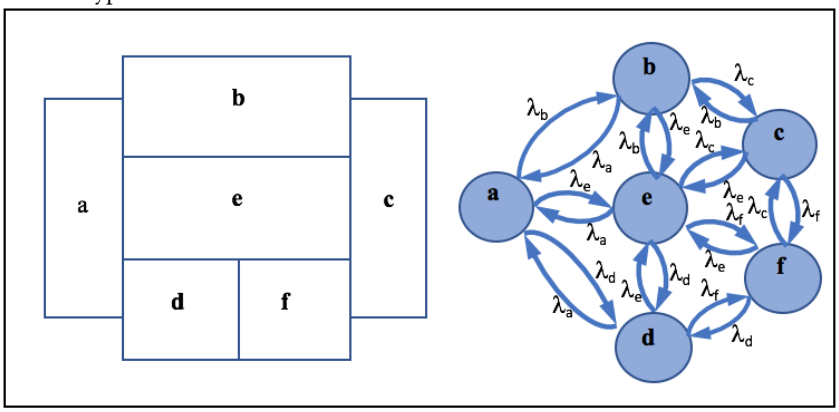

Figure 9: Regions and State Transitions

If w_1 = UK, only the subset of trajectories where all the states are safe will be considered: U = {abc}. If w_1 = US, then only the subset of trajectories in which any unsafe state is followed by a safe state will be considered: U = {abc, aec, abec, aebc}.

These constitute the locally generated policies for choosing an appropriate path.Note that if a suggested path is sent by the coalition commander, it will be checked against these allowable paths. It will be followed if it is in the acceptable set, or a reconciliation must be pursued. Otherwise, the drone can calculate its own path from the information provided and its local context.

CONCLUSIONS

In this paper we have considered how the management of computing systems is evolving due to the increasing ability to include intelligence in the devices comprising them. In particular, the influence of edge computing and IoT devices is leading to more distributed architectures for more autonomous elements. The mechanisms for operating complex systems, protecting information, and controlling access to them must also evolve.

For policy based technologies, we describe a number of efforts at making them more adaptable, contextual and cognitive. The new area of generative policies seems quite promising. A key concept in the generative approach is that local elements generate their own operational policies based upon their local context, but within the bounds of higher level policy structures supporting collaboration and meant to assure compliance with high level constraints and the pursuit of common goals.

Examples of the use and advantages of the generative policy approach have been given, and we will continue to explore its application to different domains. Some basic mechanisms that can be used to develop such generative policy systems have also been presented. There are several alternatives that could be pursued, and algorithms must be developed for checking the validity and effectiveness of generative policies to ensure that they are following the overall intent of the collaborative system.

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] Y Shoham, K Leyton-Brown, *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*, Cambridge University Press, 2008.

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

[3] Dulay N., Lupu E., Sloman M., Sventek J., Badr N., Heeps S., *Self-managed Cells for Ubiquitous Systems*. In: Gorodetsky V., Kotenko I., Skormin V. (eds) Computer Network Security. MMM-ACNS 2005. Lecture Notes in Computer Science, vol 3685. Springer, Berlin, Heidelberg.

[4] J. O. Kephart, D. M. Chess, *The vision of autonomic computing*, Computer, Volume 36, Issue 1, 2003.

[5] F. Ye, R. Ganti, S. Calo, R. Dimaghani, K. Grueneberg, *Meca: mobile edge capture and analysis middleware for social sensing applications*, Proceedings of the 21st International Conference on World Wide Web, Pages 699-702, Lyon, France, April 16 - 20, 2012.

[6] B. Moore, Ed., RFC 3460, Policy Core Information Model (PCIM) Extensions, http://www.rfc-editor.org/rfc/rfc3460.txt

[7] Ian Molloy, Youngja Park, Suresh Chari, *Generative models for access control policies: applications to role mining over logs with attribution*, SACMAT '12 Proceedings of the 17th ACM symposium on Access Control Models and Technologies, Pages 45-56.

[8] Maroun Touma, Elisa Bertino, Seraphin Calo, Brian Rivera, Dinesh Verma, *Framework for behavioral analytics in anomaly identification*, SPIE DS 2017 (to appear).

[9] Jorge Lobo, Jiefei Ma, Alessandra Russo, Emil Lupu, Seraphin B. Calo, Morris Sloman, *Refinement of History-Based Policies*, In Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning, *Springer-Verlag, Berlin, Heidelberg 280-299 (2011).*

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

[11] E. Asmare, N. Dulay, E. Lupu, M. Sloman, S. Calo, J. Lobo, *Secure Dynamic Community Establishment in Coalitions*, IEEE Military Communications Conference, (MILCOM 2007), Orlando FL, Oct 2007, Pages 1-7.

[12] 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.