CNT 4603: System Administration
Spring 2014

Introduction To Policy Management – Part 2

Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 4078-823-2790
http://www.cs.ucf.edu/courses/cnt4603/spr2014

Department of Electrical Engineering and Computer Science
Computer Science Division
University of Central Florida

Autonomic Systems

- Policies can be used to build systems that are autonomic, that is, exhibit the properties of self-configuration, self-protection, self-optimization, and self-healing.
  - A self-configuring system would configure itself according to its intended function.
  - A self-protecting system would identify threats to itself and take corrective actions.
  - A self-optimizing system would modify its configuration according to the current workload to maximize its performance.
  - A self-healing system would automatically repair any damage done to its components.
- The manner in which policy technology can be used to enable the development of such systems is the focus of this set of notes.

Policy-Based Self-Configuration

- One of the most time-consuming operations in the management of any system is the initial configuration of a new installation, or the reconfiguration that needs to be performed when new requirements are received.
- The basic approach to self-configuration is to offer a simplified set of abstractions that the administrator needs to manipulate, while the detailed configuration of a myriad of parameters in the system are hidden to a large extent.
- To better understand how policy-based self-configuration works, we'll focus on an example of a hosting service provider.

Policy-Based Self-Configuration

- For the sake of simplicity, let's assume that all the customers of this service provider run their Web sites only within the premises of the service provider, and each Web site is supported by one or more instances of a Web server such as Apache or IIS.
- The service provider has a pool of stand-by servers that can be deployed for any customer after a proper installation of applications.
- Some of the customers whose Web sites draw heavy traffic may need multiple servers at the site with a load balancer in front of them, whereas customers whose sites are not as popular may be sharing a single server with other customers.

Policy-Based Self-Configuration

- The service provider can set up a hosting Web site with a set of routers, virtual LAN switches, load-balancers, and server blades that can enable this service.
- All of these devices make up the *target system* for the policies.
- Since most hosting service providers will have system administrators who can write scripts to automate common processes (like you now can), we'll also assume that this service providers will have developed a series of scripts so that they can automatically allocate a server from the shared pool to a specific customer, and conversely return a server back to the pool once the busy period is over or the contract with the corresponding customer expires.
- We'll assume a similar script is prepared for automating the addition or removal of a new virtual server for smaller customers.

Policy-Based Self-Configuration

- When a new customer is added or an existing customer is removed, the configuration of the site needs to be changed according to the change in the set of customers being supported.
- If there are mechanisms available for servers to be assigned in an automated manner to different customers from a shared pool, then the number of servers or processors assigned to a specific customer may change depending on the intensity of traffic to that site.
- Sometimes the hosting site may want to enforce limits on how much bandwidth a customer's site can use in a month, and may want to reconfigure the site to restrict the throughput available to a hosted site if the traffic to that site exceeds predetermined thresholds.

Policy-Based Self-Configuration

- We'll assume that the service provider characterizes its customers into two groups: large and small.
- The service provider may instantiate policies for self-configuration of its site, which might be structured as follows:
  - If a large customer has 75% or more utilization of all its servers, and has less than its maximum allowed number of servers, then allocate an additional server from the free pool to that customer.
  - If a large customer has 30% utilization or less on all of its servers, and has more than its minimum allowed number of servers, then remove a server from that customer's allocation and return it to the free pool.
  - If a small customer has reached 125% of the monthly bandwidth allowed to its site, then disallow further access to the site for the rest of the month.
  - If the addition of a new small customer causes the number of small customers on a server to exceed the threshold, allocate a new server from the free pool and migrate half of the existing small customers to that new server.

Policy-Based Self-Configuration

- Based on our service provider scenario, we can identify several attributes: utilization rate, number of servers, number of free servers, monthly bandwidth, and so on.
- Each of the policies described provides a constraint on the new configuration of the system.
- The behavior of the system (allocation of servers between customers and the free pool) is constrained to perform to the guidelines established in the policies.
- The policies may change based on the experience of the service providers – instead of using server utilization, it might opt to use the bandwidth consumed as a trigger for reallocation of servers, or it may use a combination of both.

Policy-Based Self-Configuration

- Similarly, the service provider may choose not to block small customers that exceed their throughput limits, opting instead to charge them an additional amount of money.
- Based on our previous descriptions of the different types of policies, realize that all of the policies discussed here are instances of action policies.
- The example set of policies listed on page 7, allow the system administrator to manage the customers using attributes (utilization, number of servers, bandwidth rate, and so on) that are decoupled from the details of actual server configuration (their IP addresses, commands to control bandwidth, their operating system versions, and so on).

Policy-Based Self-Configuration

- Thus, the goal is to allow the system administrators to view system management in terms of the abstracted attributes that allow them to specify *what* needs to be done, leaving the details of *how* it can be done to the underlying mechanisms that support a policy-based management system.

- Policies do not describe the mechanisms for the reallocation of servers, the migration of customers to a new server, or disabling access to any site.

- However, assuming that the appropriate scripts to do these tasks exist, the ability to specify the policies and invoke the correct script for the required actions would enable the system to self-configure itself in accordance with the policies established by the service provider.

Policy-Based Self-Configuration

- Building a policy enabled management system for this type of scenario would involve three steps:

  1. Determining a way to specify the policies.
  2. Enabling support within the system to interpret and enforce the policies.
  3. Invoking a mechanism to distribute policies from the entity specifying them to the entities interpreting and enforcing them.

- To specify policies, a language that can capture the semantics of policies needs to be selected and a tool to specify the policies needs to be developed.

- We'll look at an information model and policy language to accomplish this first step a bit later.

Policy-Based Self-Configuration

- The system management software that allocates and reallocates servers would need to understand policies expressed in this language as well, so that it can enforce the policies by transferring the servers under various operating conditions.

- Finally, it is important that the policy specification from a system administrator be distributed to a system that can enforce the policies. In our service provider examples, if there is only one instance of the system management software, the third problem of distribution is trivial because there is no need for synchronizing among multiple copies of the policy.

Policy-Based Self-Protection

- Policy-based management in network administration has been practiced for more than a decade and has been used successfully in many application areas.
- A common usage of policies in network management is to regulate the traffic in a network, especially dealing with the security, access control, and quality of service (QoS) of different traffic streams.
- Some examples of network traffic security policies are illustrated on the next page:

Policy-Based Self-Protection

- Allow telnet connections out of the local network.
- Block telnet connections into the sites on the local network.
- Allow only secure HTTP traffic, and block any other traffic into the local network.
- If a UDP packet on an illegal port is received from an external computer, disallow any communication to that computer.

- The enforcement of these policies within the network, which in this case is the *target system*, needs to be managed in the context of the configuration of a specific network.

Policy-Based Self-Protection

- If we consider a fairly simple model of network access protection, in which access to the local network is secured and protected by means of one or more perimeter firewalls, then the policies can viewed as configuration constraint policies similar to those we've just described.
- The enforcement of these policies requires that the configuration of the firewall be done in a manner that is consistent with these policies.
- In this case, the *attributes* are source and destination IP address and port numbers.

Policy-Based Self-Protection

- As was the case with policy-based self-configuration, we need to have a mechanism to specify the policies, interpret and enforce the policies, and to distribute the policies from the entity specifying them to the entities interpreting and enforcing them.
- For the purpose of defining policies, a machine readable language needs to be developed for their specification.
- For access and control policies, the language could be a standard access control policy language such as the "eXtensible Access Control Markup Language" (XACML) defined by the Organization for the Advancement of Structured Information Standards (OASIS) or some other policy language with similar capabilities.

Policy-Based Self-Protection

- The firewalls in the system need to implement and enforce the access control policies.
- If they are able to interpret the language selected for specifying policies, they can take the policy as it is specified for enforcement. Otherwise, a translation of the policies to a format that the firewall can interpret needs to be done.
- In some cases, the policies might not be able to be translated easily into a firewall configuration. An example would be the presence of a policy requiring that a notification should be sent out to an administrator whenever an access attempt to a forbidden site is made. Because the firewall is not capable of sending notifications – it merely records passively the sites that users are trying to access – an additional mechanism is needed.

Policy-Based Self-Protection

- If there is more than one firewall in the system, a mechanism will be required to keep the policies specified in different firewalls consistent.
- While this could be done manually, in a fully autonomic system the process to distribute the policies would also be automated. There are various mechanisms that could be designed to handle this task, but for now, we'll defer any discussion on the various alternatives that are available.

Policy-Based Self-Protection

- In another variation of self-protection, policies may be defined that indicate how the set of applicable policies should be changed.
- As an example, if a system detects that the system is under attack by a newly identified infected host, the applicable policy rules may be augmented to prevent traffic from that infected host to reach the rest of the network.
- There are several other uses of policies in managing computer networks. The application of policy-based fault tolerance is one such example.

Policy-Based Self-Optimization

- A computer system can be called self-optimizing if it can reconfigure itself so as best to satisfy the needs and requirements imposed on it.

- In the case of enterprise systems, the optimization requirements imposed on the system are usually driven by the needs of the business. A business that owns the computer system could have some contractual obligations that it may have signed on for. For example, the hosting services provider that we used as an earlier example, may have signed a service level agreement (SLA) promising a target system response time to a customer. It would need to provision, configure, and operate its systems so that it is in the best situation to meet its obligations.

Policy-Based Self-Optimization

- Self-optimization in computer systems may be specified by means of metric constraint policies or goal policies.
- These policies would require a bound on a metric that the system may or may not directly control.
- When the metric constraint policies are specified, the system is expected to try to do its best to meet them.
- To be able to match these policies, the system must translate them into a set of action policies – that is, a set of actions that can be invoked when some conditions about the system behavior are satisfied.
- The translation of metric constraint policies into action policies is the process of policy transformation that we'll examine later.

Policy-Based Self-Optimization

- As an example of this type of policy – consider the following:
  - The service provider must be able to process at least 1000 transactions per second under the condition that the system is operating under normal load 70% of the time. Otherwise, the provider is not obligated to fulfill the requirement.
- The policy must also define normal and overload conditions (they are just not shown here).
- An implementation will provide a client of the service provider with access to the system attributes so that client or perhaps a third party can verify that the agreement has been fulfilled.
- If policies are violated, the policies themselves may include penalties or compensations as actions to encourage the system to conform its behavior to the contract.

Policy-Based Self-Optimization

- In a fully autonomic system, the system may try to predict when it is expected to fail the requirements and take corrective actions before any constraint is violated.

Policy-Based Self-Healing

- Sometimes the primary role of policies is to make sure that the operational state of the system satisfies the policies that are defined within the system.
- If the system is not satisfying those constraints, it should take corrective actions or create an alert.
- Thus, both action and alert policies can be used to implement self-healing systems, although one may take the position that the alert policy simply allows the system to call for assistance when it sees an issue rather than healing itself.
- An example of policy-based self-healing would be:
  - `The temperature in the blade center must be maintained at less than 65F.`

Policy-Based Self-Healing

- Given the policy on the previous page, if one of the cooling units in the blade center were to fail, the policy can trigger an action to put some of the blades to sleep in an effort to reduce the temperature in the center. If the action does not sufficiently reduce the temperature, an alert policy can be triggered to request the attention of the system administrator.
- Another good example can be found in storage area network (SAN) configurations management. SAN configuration is a complex problem because of the interaction between many different devices and software systems.
- Configurations need to make sure proper device drivers are installed, incompatible devices are not connected or configured in the same zone, redundancy requirements are satisfied, and so on.

Policy-Based Self-Healing

- To cope with the complexity of situations such as this, generally various sets of policies that represent best practices for interoperability and reliability are employed.
- An example of such a policy might be:
  - `The same host bus adapter cannot be used to access both tape and disk devices.`
- This policy could be verified automatically if there is an appropriate software module installed at different computers and devices in the storage area network. Data about the system configuration is collected and policies are evaluated against that data. If a configuration is not compliant, the policy management service would report the violation and may isolate parts of the system to avoid errors or failures.

Policy-Based Self-Healing

- To cope with the complexity of situations such as this, generally various sets of policies that represent best practices for interoperability and reliability are employed.
- An example of such a policy might be:
  - `The same host bus adapter cannot be used to access both tape and disk devices.`
- This policy could be verified automatically if there is an appropriate software module installed at different computers and devices in the storage area network. Data about the system configuration is collected and policies are evaluated against that data. If a configuration is not compliant, the policy management service would report the violation and may isolate parts of the system to avoid errors or failures.