Policy-based Automatic Configuration of Network Elements in Separate Segments

Tomomi Yoshioka, Tomohiro Igakura, and Toshio Tonouchi

NEC Corporation
Networking Research Laboratories
4-1-1 Miyazaki, Miyamae-ku, Kawasaki-city,
Kanagawa 216-8555, Japan
E-mail: {t-yoshioka@ab, t-igakura@bx, tonouchi@cw}.jp.nec.com

Abstract

We propose a policy-based automatic network configuration mechanism that makes the configuration of segmented networks inconsistency free. In our proposed mechanism, a policy describes what kind of events and configuration messages a network element wants to accept, and is registered on a central policy server by each network element in the initialization phase of the element. After the initialization, all events and messages are sent to the policy server, and the server forwards them to appropriate network elements considering the registered policy.

As a result of introducing the policy server, network administrators have to do nothing to the policy to adjust them for the new network configurations, which expels the inconsistency. For instance, when a network element is removed from the network, an error message is generated by the peer of the element, is forwarded to a fault management server via the policy server, and a hot stand-by element is activated automatically.

We made a proof-of-concept implementation composed of 9 Linux machines and 2 personal computers, and proved our mechanism through three scenarios, that is, an addition of a new AP Service Server, a removal of a failed AP Service Server and a takeover by a hot stand-by server, and an addition of a network segment composed of a firewall and a server.

Keywords
Policy-based Management, Automatic Configuration, End-to-End Provisioning

Contact Person
Tomomi Yoshioka(t-yoshioka@ab.jp.nec.com)
1. Introduction – Corporate Network

Large networks such as corporate networks always suffer from heterogeneity because these networks consist of many network elements and many segments as figure drawn above. This kind of networks easily causes inconsistency of configuration among multiple network segments. Moreover, since corporate networks keep on changing to adapt the rapidly evolving business situation, the configurations of network elements are also required to be updated. To manage this kind of heterogeneous, distributed, complex and dynamic networks, system administrators want to focus on policy level management instead of taking their time for node level configuration. Therefore we selected to use policy based management system as the large scale network management system.
2. Issues

In [1], Policy Server only can distribute the same policy to network elements. This causes two problems. One is the difficulty to handle heterogeneity in a network. Since a policy must be applied to all network elements, it is difficult to support different kinds of network elements simultaneously, e.g., routers, firewalls and various kinds of servers. The second problem is the difficulty to manage the policy because a change in the policy affects all network elements.

We therefore believe that we must improve a Policy Server to handle various policies simultaneously, which enables network elements to receive policies optimized for each elements, and which prevents a policy change for a network element to affect others.

In addition, traditional policy control language[2,3,4] focuses on the control of one network element, and lacks the ability to describe relationship among managed network elements. For example, in Ponder[5,6], a policy is given for a set of network elements called a target, and it does not have a mechanism for specifying collaboration among network management elements.
Proposal; Policy-based Automatic Configuration System

- Policy Server;
  - accepts policies that determine messages sent to network elements,
  - forwards the control command messages to hosts that have to receive it.
- Therefore;
  - this policy-based automatic network control system is able to handle heterogeneity because the Policy Server can receive and send messages to any kinds of network elements,
  - since the message forwarding is managed with the policy on the server, each network element receives messages that do relate the element.

---

3. Proposal; Policy-based Automatic Configuration System

Let us first consider the architecture of the control system itself. In its core lies a Policy Server[7], which manages all the various policies registered in it. In this section, we refer to the servers of our previous discussion as AP Service Servers, and to all non-client elements (including AP Service Servers) as hosts. We define the policy as a message forwarding rule for the Policy Server to forward an incoming message to appropriate network elements.

The figure above describes an example of our proposed architecture.

1. Bob (client) registers his policy in the Policy Server in the initializing phase. Other hosts such as a printer manager and a pager server also register their policies in the Policy Server.
2. A printer that has already registered its policy in the Policy Server sends a control command message to the Policy Server.
3. The Policy Server forwards the message to a host (e.g., printer manager, pager server and so forth) whose policy indicates that it offers the particular service for the control command.

Our proposed Policy Server accepts policies that indicate where a received message is forwarded, and manages these policies. In addition, when the Policy Server receives the control command message, the Policy Server forwards the message to host(s) whose policy indicates that it offers particular service for the message. This policy-based automatic network control system is therefore able to handle heterogeneity because the Policy Server can receive and send messages to any kinds of network elements. In addition, since the message forwarding is managed with the policy on the server, each network element receives messages that do relate the element.
4. Policy-based System Architecture

4.1 Overview of automatic management consistent over separated segments

Shown above is the overview of our policy-based system architecture that realizes automatic management consistent over separated segments.

It models a company’s intranet as a network composed of three components, that is, a Personnel Department which offers a desired service, a Research Department to which belongs a Client who wishes to use the Personnel Department’s service, and a System Administration Department which supervises the servers that control all the elements of the network system.

Each of these three segments is protected with a firewall and they are connected one another with a Virtual Private Network (VPN).

For the Research Department Client to use the service offered by the Personnel Department, three steps are required:

1. Authentication of the Client
2. Reconfiguration of all relevant firewalls
3. Access to the AP Service Server

Let us next consider how these three steps are actually performed.

We adopt an original protocol for communication among the servers and the firewalls. The original protocol is called Compact Data Representation (cDR), which is dedicated to communication among elements with poor CPU powers, such as a hub. The cDR is a light-weight protocol while the SOAP is heavy. This is why we adopt the cDR. The cDR is designed so that it can express data that the SOAP can express. A stub in a client and a skeleton in a server talk each other through the cDR. The skeleton plays another important role: it absorbs heterogeneity of hosts. For the sake of this, we do not mind the heterogeneity.
4.2 Configuration Sequence

1. For a Client to utilize a service, it is first necessary for him/her/it to log onto the system, and to do this he/she/it first asks Authentication Server.

2. After the authentication has been made, the Authentication Server sends a request to the Policy Server to register the Client.

3. The Authentication Server reconfigures the firewall that protects the Client’s department.
   
   More specifically, it reconfigures the Network Address Translation (NAT) function contained in the firewall so that the “sender address” in the Client’s message will be changed.

4. The Authentication Server sends a message to the Policy Server to reconfigure the firewall that protects Client in the Research Department and the firewall that protects the AP Service Servers in the Personnel Department.

   An example of a control command message that makes such a request is shown in the figure above.

5. The Policy Server compares header data in the control command message with policies registered in it and forwards the message to appropriate firewalls to request reconfiguration (an example policy is also shown in the figure).

When the reconfiguration of all firewalls completes, the Client can access the desired AP Service Server.

6. The Client sends an execution request message to the firewall that protects the Personnel Department.

   This message contains the port number of the desired service provided by the Personnel Department’s AP Service Server. As the message passes through the Personnel Department’s own firewall, the sender IP address is changed.

7. The Personnel Department’s firewall forwards the execution request message to the AP Service Server.

8. The AP Service Server executes the requested service and replies all required information resulting from that execution, including notification of execution, to the firewall of the Research Department.

   This message passes through the Personnel Department’s firewall without modification in sender address.

9. The Research Department’s firewall forwards the message to the Client.

   As shown, this design enables automatic reconfiguration of firewalls, making it easy for the Client to use a service provided by another segment in the intranet.
4.3 Control Command Message and Policy

The sample control command message in the above figure represents the message mentioned in Step 4 in the previous page. Any control command message consists of a header and a payload. We assume that a header consists of sequences of pairs, each pair being the combination of an attribute name (e.g., “receivers”) and an attribute value (e.g., “/net/fw/”).

The payload includes command data written in cDR to be executed on the destination Host(s). In the above figure, the data in the payload indicates that Method “thruClient” in Class “ClientNatAndFw” in the destination Host(s) is executed with Parameter Values of “192.168.0.50” and “10.56.33.54”.

The header plays the key role in forwarding the message. The Policy Server compares the message with policies stored in it and determines the destination of the message. The policies in the server describe patterns, any of which may match with an attribute-name/attribute-value pair in a message header. If the condition in the policy rule set matches with the header, the matched message is forwarded to the corresponding destination Host.

In the Step 4, the control command message is sent to the Policy Server, which compares the message with registered policies. In the above figure, there are two policy matches: the first one is the condition of “receivers=/net/fw/research/xxx01” and another is the condition of “receivers=/net/fw/personnel/xxx01”. The message is therefore forwarded both to the firewall of the Research Department and to the firewall of the Personnel Department.

The firewall of the Research Department has a NAT function, which translates a private IP address of a client to a global IP address. The firewall protecting the Personnel Department thus permits access from the Client distinguished with the global IP address.

This example shows the advantage of our proposed policy mechanism. The Authentication Server, which issues the control command message, does not care the destination where the message is sent; our policies describe which host will receive what kind of message. Based on those policies, the Policy Server forwards incoming messages to appropriate destinations. Consequently, even in the case that a new server is connected to the network, the existing policy set needs no modification since the policies depend only on the server, but independent from another network elements.
5. Plug and Play of a New AP Service Server

In order to confirm the effectiveness of proposed architecture, we made a prototype system which supports the following three features. The system consists of 2 personal computers and 9 Linux machines (RedHat 7.2) including the Policy Server. We describe the first feature in this section.

When an AP Service Server is newly connected to the network system, the network elements need new configurations to adapt the current situation. This configuration is conducted automatically as follows:

1. A DHCP (Dynamic Host Configuration Protocol) request is sent to a DHCP server in the System Administration Department. This DHCP server assigns an IP address to the new AP Service Server and sends it a DHCP reply.
2. The AP Service Server registers its policy into the Policy Server with the assigned IP address.
3. The AP Service Server registers itself to the firewall of the Personnel Department so that the firewall can pass the service request.
4. The AP Service Server sends a request to the Policy Server to reconfigure the firewall for potential clients.
5. The Policy Server also reconfigures the client side firewall if necessary.

As described above, any time a new AP Service Server is added to the system, our Policy Server reconfigures the related firewalls automatically.
6. Automatic Failure Recovery

Shown above is the second example of our proposed architecture, that realizes automatic failure recovery. The Personnel Department contains AP Service Servers 1 and 2. The AP Service Server 1 is current working server and Server 2 is the hot stand-by. And the System Administration Department operates a Fault Management Server. When a failure occurs in AP Service Server 1, it is automatically disconnected and AP Service Server 2 takes it over.

6.1 Failed AP Service Server deregistration process

When a Client sends an execution request message to a failed AP Service Server 1:

1. The AP Service Server 1 replies an error response to the Client.
2. The Client then automatically sends an error report to the Policy Server.
3. The Policy Server compares header field in the error report to policies registered in it beforehand and forwards the error report to the Fault Management Server.
4. The Fault Management Server sends a request message to the Policy Server to deregister AP Service Server 1 policy.

All the Network elements have to do is to send request messages to the Policy Server. The Policy Server takes all the responsibility for forwarding these requests to appropriate servers for failure recovery.
Feature 2 ~Automatic Failure Recovery (2)~

- AP Service Server 2 take-over process

6.2 AP Service Server 2 back-up execution
After AP Service Server 1 has been disconnected, AP Service Server 2 takes over the Server 1 as follows:

5. The Fault Management Server sends a take-over request message to the Policy Server.
6. The Policy Server forwards the take-over request message to AP Service Server 2.
7. AP Service Server 2 takes the Server 1 over, serves the original requested service and sends the result of the service to the Policy Server.
8. The Policy Server sends the result to the Fault Management Server.
9. The Fault Management Server sends the result to the Policy Server.
10. The Policy Server sends the result to the Client.

With this automatic utilization of AP Service Server redundancy, Clients need not be concerned about AP Service Server failures.

We assume a hot stand-by server always synchronizes its internal status with a master server, therefore the hot stand-by server do not have to pay any attention to which server the request is sent, or when the take-over action is processed. This issue is a matter of server architecture and beyond our scope.
Feature 3 ~Reconfiguration of Network Segments~

Even if a new network segment is added, existing policies is required no modifications.

7. Reconfiguration of Network Segments

Shown above is the third example, in which a new network segment has been added. This reconfiguration is conducted automatically as follows:

1. Registration of new Department’s firewall
   The firewall protecting Department $\beta$ notifies the Policy Server of its existence.

2. Registration of new AP Service Server $\beta$ in Department $\beta$
   AP Service Server $\beta$ registers itself into the Policy Server.

3. Log-on of a Client

4. A command message to reconfigure the server side firewalls

5. Reconfiguring the server side firewalls
   The Policy Server compares header data in the received command message to policies registered in it in step 2 and forwards the command messages to appropriate firewalls.
Conclusion & Future Work

• Conclusion
  – We proposed a network control system utilizing our Policy Server and made a proof-of-concept implementation.
  – Important features of this network control system are:
    • Enabling to apply policy control even in heterogeneous network environment
    • Adapting policy controls respecting the relationship among network elements
  – The example applications of our system are:
    • Plug and play of a new AP Service Server
    • Automatic failure recovery
    • Reconfiguration of network segments

• Future Work
  • A translator which makes stubs changing protocols from WSDL
  • An application tool which helps administrators to write policy
  • Quantitative evaluation
  • A study of next-generation autonomous system

8. Conclusion & Future Work
We proposed a network control system utilizing our Policy Server and made a proof-of-concept implementation.

Important features of this network control system are
- enabling to apply policy control even in heterogeneous network environment, and
- adapting policy controls respecting the relationship among network elements.

The example applications of our system are
- plug and play of a new AP Service Server,
- automatic failure recovery, and
- reconfiguration of network segments.

In the future, we are going to develop
- a translator which makes stubs changing protocols from WSDL, and
- an application tool which helps administrators to write policy.

The rests of work but development are:
- quantitative evaluation (e.g., number of policy which the Policy Server is able to manage, number of clients whom system administrator is able to supervise, and so forth), and
- a study of next-generation autonomous system.

References