Active Bandwidth Brokers:
A New Framework for Policy-Based Management

By

Kason Wong

A thesis submitted in conformity with requirements
for the degree of Masters of Applied Science and Engineering
Graduate Department of Electrical and Computer Engineering
University of Toronto

© Copyright by Kason Wong (2001)
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author’s permission.

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L’auteur conserve la propriété du droit d’auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-63038-2
Abstract

We discuss a new approach in Policy-Based Management (PBM), which can realize end-to-end Quality of Service (QoS) for all connections across the Internet. The design is based on a reference PBM standard and its supporting communications protocols. These protocols are Common Open Policy Service (COPS) and Lightweight Directory Access Protocol (LDAP). Our proposed model is designed to work with active networks technology. In contrast to the traditional network, an active network is able to manipulate data in the packets, and change the network functionalities on a per-user and application basis. Our proposed framework, the Active Bandwidth Broker (ABB), delivers an easily maintainable, mobile and load sharing architecture for policy control. Our goals are to achieve a distributed PBM framework to improve scalability, and to introduce a mobile mechanism to enhance the availability of the service. This results an intelligent network that is incorporated with QoS control to generate performance protection for voice, video and Internet business applications.
Acknowledgements

I would like to thank Professor Eddie Law for his invaluable guidance, and advice throughout my research.

I would also like to express my gratitude to my peers in the Network Architecture Labs at the University of Toronto, especially Achint Saxena and Roy Leung for their helps and supports.

Thanks to my parents and friends for their consideration, especially Wheelma Lee, Amy Wang, Karen Wang, and Michele Yeung for their kindness and encouragement.

Finally thanks to OGS and OGSST and Communication Group at the University of Toronto for financial support.
# Table of Contents

**LIST OF TABLES** ......................................................................................................................... VI
**LIST OF FIGURES** ......................................................................................................................... VII
**LIST OF TERMS AND ACRONYMS** .............................................................................................. IX

**CHAPTER 1** .................................................................................................................................. 1

1.1 **NEW POLICY-BASED MANAGEMENT FRAMEWORK** .......................................................... 5
1.2 **PROVIDING QUALITY OF SERVICE WITH POLICY-BASED MANAGEMENT** ................. 7
   1.2.1 Integrated Services .................................................................................................................. 8
   1.2.2 Differential Services ............................................................................................................... 8
   1.2.3 PBM and QoS control .......................................................................................................... 9
1.3 **SUMMARY** ............................................................................................................................... 11
1.4 **CONTRIBUTION** ..................................................................................................................... 11
1.5 **ORGANIZATION OF THIS THESIS** ....................................................................................... 12

**CHAPTER 2** .................................................................................................................................. 13

2.1.1 **Negotiating QoS with Service Level Agreement** ................................................................. 16
2.1.2 **Allocating Resource with Bandwidth Broker** .................................................................... 17
2.1.3 **Executing Policies in Policy Enforcement Point** ............................................................... 18
2.2 **BENEFITS OF POLICY-BASED MANAGEMENT** ................................................................. 20
2.3 **ISSUES AND CONCERNS** ..................................................................................................... 20
2.4 **SUMMARY** ............................................................................................................................... 22

**CHAPTER 3** .................................................................................................................................. 23

3.1.1 **Two Approaches in Active Networking** ............................................................................. 24
3.2 **ACTIVE SIGNALLING PROTOCOL (ASP)** .............................................................................. 25
   3.2.1 Example of ASP Active Applications .................................................................................. 28
3.3 **ACTIVE BENEFITS IN NETWORK MANAGEMENT** ................................................................. 29
3.4 **OTHER ACTIVE APPLICATIONS** ........................................................................................... 31
3.5 **SUMMARY** ............................................................................................................................... 32

**CHAPTER 4** .................................................................................................................................. 33

4.1 **ABB IMPLEMENTATION** .......................................................................................................... 36
4.2 **COMPONENTS OF ABB** ......................................................................................................... 39
   4.2.1 **Edge Router/PEP** ............................................................................................................ 40
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.2</td>
<td>P-ABB/PDP</td>
<td>41</td>
</tr>
<tr>
<td>4.2.3</td>
<td>S-ABB</td>
<td>41</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Policy Repository</td>
<td>43</td>
</tr>
<tr>
<td>4.3</td>
<td>PROTOKOLS</td>
<td>45</td>
</tr>
<tr>
<td>4.3.1</td>
<td>SLA Message</td>
<td>47</td>
</tr>
<tr>
<td>4.3.2</td>
<td>COPS Message</td>
<td>47</td>
</tr>
<tr>
<td>4.3.3</td>
<td>LDAP Message</td>
<td>49</td>
</tr>
<tr>
<td>4.3.4</td>
<td>Location Protocol Message</td>
<td>50</td>
</tr>
<tr>
<td>4.4</td>
<td>ABB OPERATIONS</td>
<td>52</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Distributed Policy Processing</td>
<td>52</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Server Migration</td>
<td>54</td>
</tr>
<tr>
<td>4.5</td>
<td>SUMMARY</td>
<td>55</td>
</tr>
<tr>
<td>5.1</td>
<td>MEASUREMENT</td>
<td>60</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Measurement Metrics</td>
<td>60</td>
</tr>
<tr>
<td>5.2</td>
<td>PERFORMANCE RESULTS</td>
<td>62</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Evaluation of Congestion Avoidance Mechanism</td>
<td>62</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Evaluation of Distributed Policy Management</td>
<td>68</td>
</tr>
<tr>
<td>5.3</td>
<td>SUMMARY</td>
<td>71</td>
</tr>
<tr>
<td>6.1</td>
<td>FUTURE WORK</td>
<td>74</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>APPENDIX A</td>
<td>81</td>
<td></td>
</tr>
<tr>
<td>APPENDIX B</td>
<td>84</td>
<td></td>
</tr>
<tr>
<td>APPENDIX C</td>
<td>87</td>
<td></td>
</tr>
</tbody>
</table>
List of Tables

Table 4.1 Client-type for ABB Protocol Message ................................................................. 46
Table 5.1: Active Node Specification ..................................................................................... 58
Table 5.2: Node Configuration in Test Bed ........................................................................... 59
Table 5.3: Response Times for Static and Mobile Server of ABB Model ......................... 64
Table 5.4: P-ABB Process Time for Decision-Making ......................................................... 69
Table 5.5: Observation of Traffic related to Policy Rule Searches and Results .............. 71
List of Figures

Figure 1.1: Policy-based Management ................................................................. 2
Figure 1.2 Basic Policy-based Management Model ............................................. 4
Figure 1.3 Overview of Policy-Based Management to Enable QoS ................... 9
Figure 2.1: The Basic Structure of Policy-based Network Management System 14
Figure 2.2: Structure of Policy Storage in the Repository ................................. 15
Figure 2.3: The Basic Functionalities of a Bandwidth Broker ......................... 18
Figure 3.1: Packets Processing within Active Network ..................................... 24
Figure 3.2: Architecture of an Active Node ..................................................... 26
Figure 3.3: Traditional traceroute signalling .................................................... 28
Figure 3.4 Operation of Traceroute in Active Network ..................................... 29
Figure 4.1: Centralized Policy-Based Management Model .............................. 34
Figure 4.2: ABB Framework for PBM ............................................................ 34
Figure 4.3: Structure of ABB Implementation ............................................... 36
Figure 4.4: Architecture of Active Node .......................................................... 38
Figure 4.5: Components of Active Bandwidth Broker (ABB) ......................... 40
Figure 4.6: Policy Conflicts Solution ............................................................... 42
Figure 4.7: Policy Scheme in LDAP Directory ............................................... 44
Figure 4.8: LDAP Transceiver ....................................................................... 44
Figure 4.9: Common header for ABB message, number shows the byte order .... 46
Figure 4.10: COPS Message Exchange in ABB Framework ......................... 48
Figure 4.11: the LDAP Message in ABB Model ............................................. 49
Figure 4.12: Policy Request and Decision handling in ABB Framework ........... 53
Figure 4.13: P-ABB Migration ..................................................................... 55
Figure 5.1: Topology of Active Network ......................................................... 58
Figure 5.2: P-ABB Migration in the Congested Network Configuration .......... 63
Figure 5.3: Histogram of P-ABB Response Time in a non-congested network .... 65
Figure 5.4: Histogram of the stationary P-ABB Response Time in a Congested Network .................................................................................................................. 65
Figure 5.5: Histogram of the Mobile P-ABB Response Time in Congested Network ..... 65
Figure 5.6: Histogram of Request Loss Rate at Each S-ABB with stationary P-ABB. ....67
Figure 5.7: Histogram of Request Drop Rate at Each S-ABB with Mobile P-ABB........67
Figure 5.8: Comparison of the Effect of Limited Bandwidth to Request Loss Rate between
Stationary Server and Mobile Server. ..............................................................68
Figure 5.9: Process Time Distribution of P-ABB with no network computation power...70
Figure 5.10: P-ABB Process Time Distribution with “on-the-fly” Network Computing .70
# List of Terms and Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td>Active Application</td>
</tr>
<tr>
<td>AN</td>
<td>Active Network</td>
</tr>
<tr>
<td>ASP</td>
<td>Active Signalling Protocol</td>
</tr>
<tr>
<td>ABB</td>
<td>Active Bandwidth Broker</td>
</tr>
<tr>
<td>BB</td>
<td>Bandwidth Broker</td>
</tr>
<tr>
<td>CBQ</td>
<td>Class-Based Queuing</td>
</tr>
<tr>
<td>COPS</td>
<td>Common Open Policy Service</td>
</tr>
<tr>
<td>CLI</td>
<td>Command Line Interface</td>
</tr>
<tr>
<td>DiffServ</td>
<td>Differential Services</td>
</tr>
<tr>
<td>EE</td>
<td>Execution Environment</td>
</tr>
<tr>
<td>LDAP</td>
<td>Lightweight Directory Access Protocol</td>
</tr>
<tr>
<td>P-ABB</td>
<td>Primary Active Bandwidth Broker</td>
</tr>
<tr>
<td>PBM</td>
<td>Policy-Based Management</td>
</tr>
<tr>
<td>PDP</td>
<td>Policy Decision Point</td>
</tr>
<tr>
<td>PEP</td>
<td>Policy Enforcement Point</td>
</tr>
<tr>
<td>PHB</td>
<td>Per-Hop Behaviour</td>
</tr>
<tr>
<td>PIB</td>
<td>Policy Information Base</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>IntServ</td>
<td>Integrated Services</td>
</tr>
<tr>
<td>RSVP</td>
<td>Resource Reservation Protocol</td>
</tr>
<tr>
<td>S-ABB</td>
<td>Secondary Active Bandwidth Broker</td>
</tr>
<tr>
<td>SLA</td>
<td>Service Level Agreement</td>
</tr>
<tr>
<td>SLO</td>
<td>Service Level Objective</td>
</tr>
<tr>
<td>SNMP</td>
<td>Simple Network Management Protocol</td>
</tr>
<tr>
<td>TCP</td>
<td>Transport Control Protocol</td>
</tr>
<tr>
<td>ToS</td>
<td>Type of Service</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
</tbody>
</table>
Chapter 1

INTRODUCTION

Internet provides connectivity among people around the world. Many emerging applications, such as the Internet phone and video conferencing services, require the Internet to move from a single service class model into multiple services model. The traditional single service class model considers all network connections as equal in accessing network resources. It is usually called the “best effort” traffic model because it does not guarantee any Quality of Service\(^1\) (QoS) for the user applications. On the other hand, with multiple service classes, networks can provide predictable and controllable services for different applications with different QoS requirements. The main idea of multiple service classes is to prioritize the usage of the network resources. In other words, real-time application messages in the high priority service class can be allocated with more network resources to achieve the QoS requirements, while other messages such as email messages can be assigned to the lower priority class.

\(^1\) Quality of Service refers to the ability to deliver network services according to the parameters specified by service availability, delay, jitter, throughput, and packet loss ratio.
In order to provide QoS in the networks, network administrators are required to learn to set up various QoS mechanisms on different vendors' equipment accordingly. Traditionally network administrators must manually configure and manipulate network elements through command line interfaces (CLI) or Simple Network Management Protocol (SNMP) [7]. This is a difficult task because configuring networks tends to be a long and laborious process with this device-by-device approach. In response to this situation, the Internet Engineering Task Force (IETF) has defined a standard framework that automates and simplifies QoS control and management in the Internet. This framework is called Policy-Based Management (PBM) [2].

In general, PBM is an administrative approach that is used to simplify the management by establishing policies to deal with situations that are likely to occur. Hence, a network administrator can create network policies [46] to define how resources or services allocated and used in networks. As shown in Figure 1.1, PBM provides the infrastructure to convert these policies into configurations and to enforce them in networks.

![Figure 1.1: Policy-based Management](image-url)
The definitions of policies are described in reference [5]: "Policies are plans of an organization to achieve its objectives. A policy is a persistent specification of an objective to be achieved or a set of actions to be performed in the future or as an on-going regular activity." For QoS network management, policy indicates the relationship between network objects, such as particular groups of network elements, network resources and services, and user groups. PBM is a framework to organize these policies. It is concerned with the implementation of organizational objectives as automated operations, management and control system. While a policy is a set of guidelines and rules to streamline the network configuration for providing QoS network, PBM is an excellent tool to express and control these policies. Ideally, PBM can be used as an administrative tool throughout the network to control the access of users and applications, prioritize the usage of bandwidth and allocate network resources according to the network policies and the network states at any given time.

Compared to the conventional network management using device-by-device configuration, PBM simplifies the management interfaces by extracting commonality from different network devices. By using the same configuration control, the network administrator can achieve consistent QoS control in the managed environment and eliminate duplication of efforts. In other words, network control and QoS decision criteria across multiple devices are standardized and simplified by establishing predefined network policies, which give the network parameters over the managed domain. Network configurations for QoS control can be derived from these policies, and policies can be customized and different from network to network. The centralized data management and simplified interfaces in this PBM system are the keys to achieve better consistency and simplicity of QoS management.
As shown in Figure 1.2, the architecture of PBM consists of two kinds of entities: *Policy Decision Points* (PDP) and *Policy Enforcement Points* (PEP). PBM operates in a client/server fashion. PEPs, the clients located in network devices, construct policy requests and send them to the PDP. PDP, the policy server, responds with policy decisions according to the policies stored in the policy repository. The policy decisions are then enforced by the PEPs in the network. In a PBM network, a user access to the network can trigger a policy request. This policy request contains information of the access including application type, time of the day, source and destination address. According to the network policy, PDP replies a policy decision which controls the QoS configuration for such access.

![Figure 1.2: Basic Policy-based Management Model](image)

In terms of scalability, there are more network routers (PEPs) in a larger network, and it requires more PDPs to share the control workload. Coordination and intercommunication among multiple PDPs are necessary. Moreover, when a connection request goes through a number of domains for the end-to-end QoS, the negotiation of the SLA becomes complicated among multiple PDPs across the
networks. A simple way to streamline the policy decision process is to maintain one unique PDP in a domain, where coordination between PDPs can be eliminated. Similarly, cross-domain bandwidth allocation can be operated more effectively, as the negotiation between the two adjacent domains remains in one-to-one PDP communication. This simple solution results in a highly centralized model. However, a centralized model has a fundamental scalability problem in its structure. When there is a sudden burst in QoS requests, PDP becomes the point of failure as its local resources are depleted. Furthermore, PBM introduces additional network control traffic due to network status probing, network policy requests and replies and QoS provisioning [4]. With more network elements managed in a large service provider network, the scalability problem and network control overhead can only be exacerbated.

1.1 New Policy-Based Management Framework

In this thesis, we apply active network [9] technology to achieve a more scalable and robust PBM. Active network is a completely different network approach compared to the traditional network. Active network offers a programmable network environment that allows network nodes to dynamically customize network operations and functionalities.

The active network is “active” in two senses:

- Instead of limited operation in the packet header, routers can modify and operate on packet data flowing through them.
- User can customize the network operations and computations with programs dynamically dispatched to the remote routers.
In Chapter 3, we will discuss the active network structure and its applications in details.

Our novel PBM framework, called the Active Bandwidth Broker (ABB) is the integration of PBM and active networking technology. The objective of this work is to achieve dynamic automation of QoS network configuration, and to overcome the standard PBM architectural limitation on scalability. Applying active networking on PBM, ABB brings up three important characteristics of this project.

- Distributed policy decision process in the network: Active network enables the policy decision process within the network. In particular, the network can perform functions such as filtering and resolving conflicts on the traversing policy data. Refining of policy rules in the network, can leave simpler policy decision to be done in the policy server. This distributed system has an advantage of better scalability in term of handling more incoming policy requests compared to the traditional model.

- Mobile policy server: In an active network, installation and configuration of policy server and network elements can be done automatically with minimum administrative intervention. This dynamic installation mechanism also enables the migration of policy server in the network to avoid poor performance during localized congestion or network malfunction.

- Intelligence in the network: A traditional network requires the policy server to probe for network status periodically. In ABB framework, active nodes can be programmed to self-monitor and to report urgent messages due to device failure or buffer overflow.
The ABB framework allows distributed computing of policy decisions in the network. Instead of locating more policy servers in a larger managed domain, introducing more active nodes to the network can reduce the workload at the decision point. Hence, our ABB framework has better scalability in terms of handling larger number of policy requests. On the other hand, ABB with code mobility has better flexibility [31] in the service installation. Policy server can detect the incoming congestion, and avoid the poor performance by migrating to a low traffic area or relocating a new server as a backup service.

1.2 Providing Quality of Service with Policy-Based Management

The premise of this project is to develop a new PBM framework for QoS mechanisms. In this chapter, we present the QoS mechanisms in the IP network and PBM as the framework for these services. The detailed structure of PBM will be discussed in Chapter 2.

Quality of Service (QoS) refers to the “classification of packets for purpose of treating certain classes or flows of packets in a particular way compared to other packet.” [52] The idea is to make the data delivery service of IP network predictable in terms of some criteria such as delay, jitter, and loss rate. To enable QoS requires the use of protocols such as RSVP and DiffServ to provide integrated services and differential services.
1.2.1 Integrated Services

Integrated Services (IntServ) [38][39] architecture provides QoS at the granularity of a data flow with its signalling protocol, Resource Reservation Protocol (RSVP) [40]. The main purpose of IntServ is to ensure QoS available to the application along the whole path throughout the network infrastructure when it is required. IntServ provides the ability for applications to choose among multiple controlled levels of delivery service for their data packets. The signalling protocol, RSVP, is used to ensure bandwidth is reserved along the connection.

Although it gives very precise control, it scales poorly and brings on additional costs in delay, network traffic and devices configuration. That makes only applications such as voice and video applications reasonable in using the IntServ model. In Microsoft’s Windows 2000, QoS features already employ a heavy use of RSVP signalling.

1.2.2 Differential Services

Differentiated Services (DiffServ) [37] architecture was designed as a scalable complement to Integrated Services/RSVP. Instead of dealing with the data on a per flow basis, a small number of service classes are provided. Data packets are marked in the edge routers with a label that identifies which class it belongs to. DiffServ marked packets carry their own state with them. In the core of a network, packets are forwarded according to the per-hop behaviour associated with the ToS (Type of Service) field. By pushing the process to the edges of the network, forwarding in the core can take place more quickly and efficiently.
IntServ service can be mapped [41] to DiffServ flows for transport across the WAN or service provider networks; thus, IntServ and DiffServ can and should coexist in enterprise networks.

1.2.3 PBM and QoS control

In order to provide the service models (IntServ and DiffServ) for QoS control, a network requires an infrastructure to authenticate traffic that requests the better service levels and to verify the identity of traffic. PBM satisfies these requirements by providing a complete model describing the automatic procedure of traffic classification, storage of policies, signalling mechanism and dynamic network configuration for QoS establishment [33][34].

![Diagram of Policy-Based Management](image)

Figure 1.3 Overview of Policy-Based Management to Enable QoS

The overview of PBM is shown in Figure 1.3. The PBM architecture is designed in such a way that high-level management knowledge is captured by policy rules and stored in a policy repository. Policy rules are predetermined by the network
administrator. Once these policies are defined and deployed at the policy server (PDP), they are further translated and downloaded to corresponding network devices (PEP) to place the policies in effect. COPS protocol is used for communication between PDP and PEP.

COPS protocol is an outsourcing protocol which can be extended for specific customization. For instance, an extension COPS-PR[2] is adapted for bulk DiffServ enforcement. Another QoS control protocol, Resource reSerVation Protocol (RSVP) [40] can also be well adapted to the COPS model. Policy can be used to control the set of configuration parameters and routing for each class in Differentiated Service, and the admission conditions for reservations in Integrated Services.

**Provisioning QoS with COPS:**
In this case, user application contacts the PDP for a specific service request. If service is admitted, PDP contacts corresponding network devices (PEP) in its domain to transmit policy decisions [44] to enforce. Those policy decisions define the QoS control and network configuration predetermined for such server request. This architecture, where user (network administrator) directly contacts PDP, is called QoS provisioning model and service levels are pre-configured for Differential Services (DiffServ).

**Signalling QoS with COPS:**
In signalling QoS, user approaches an edge router (PEP) of the network. This edge route in its turn contacts the PDP and specifies the information of the user access. If service is admitted, data packets will be granted with the required QoS. The Integrated Services/RSVP architecture uses this model.
1.3 Summary

Policy-Based Management (PBM) is an efficient and effective approach to deliver Quality of Service in the network, while delineated policies are used to control access to and priorities for the use of resources. It should be stressed that the premise of this thesis is to introduce active networking technology into PBM to achieve a more dynamic framework to surmount the scalability limitation in its standard model.

1.4 Contribution

The contribution of this work is to provide a new scalable policy-based management framework: Active Bandwidth Broker, which combines active networking technology into the policy-based management model. Through performance analysis, this novel framework is proven to achieve a superior architectural advantage of scalability and flexibility to avoid the bottlenecks associated with traditional centralized model. Additionally, active network provides a heuristic infrastructure to support dynamic network element configuration and update, which release the arduous process in the QoS network management. Specially, a new function of congestion avoidance with server backup and migration mechanism is introduced and dramatically enhances the availability of management system.
1.5 Organization of This Thesis

This thesis is organized into six chapters covering the following major topics:

Chapter 1. **Introduction of Active Bandwidth Broker (ABB)**
Present a new solution of envisioning QoS guaranteed network integrating policy-based management and active network technology.

Chapter 2. **Overview of the standard PBM structure**
Depict each component of the standard PBM infrastructure and the limitation of PBM.

Chapter 3. **Introducing Active Networks**
Provide the concept of Active Networking and its advantage to improve the performance of PBM.

Chapter 4. **Design and Implementation of the Active Bandwidth Broker (ABB)**
Reveal our novel design: Active Bandwidth Broker (ABB) to generate smarter framework for QoS envisioning.

Chapter 5. **Performance Evaluation**
Illustrate ABB test performances to verify the advantages of ABB framework over the standard IETF model

Chapter 6. **Conclusion and Future Work**
Summarise the highlights of ABB model, and possible improvements and modifications on the current prototype.
Chapter 2

STRUCTURE OF POLICY-BASED MANAGEMENT

In Figure 2.1, a basic overview of the PBM architecture is illustrated. The IETF has defined a system model called the Common Open Policy Service (COPS) [1], which describes a standard structure of PBM. There are two main entities in the COPS model: the Policy Decision Points (PDP) and the Policy Enforcement Points (PEP). The PEP, located in the network devices, begins with receiving request messages from neighbour domains or users, for instance a Service Level Agreement (SLA) [36] that describes a certain level of QoS. PEP converts these request messages to COPS request messages and then forwards them to the PDP. According to the policy rules stored in the policy rule repository, PDP returns policy decisions to PEP, where the policy actions are enforced. In order to employ PDP and PEP from different vendors in the same network, COPS protocol is defined for information exchange. This protocol uses TCP (Transmission Control Protocol) as its transport protocol for reliable communication between policy clients (PEP) and their server (PDP).
Network policies can be built and stored in a directory located throughout the network rather than in one centralized location. A directory is a physically distributed repository for infrequently changing information storage. With a directory, policies can be proliferated within and across networks, so that access rights and privileges can be tracked and managed easily. LDAP (Lightweight Directory Access Protocol) [42] is a standard and the most widely-accepted protocol to access a directory repository. The structure of policies stored in the repository is defined by a policy schema. A standard proposed by IETF [6] defines a basic structure of policy storage which contains three main types of entries: policies, conditions [45], and actions [44] with the general semantics of “Policy: if Condition, then Action”. For example, one policy can be “policy A: if user is Mary && time is Monday && application is net-meeting, then allow delay < 100 ms and bandwidth = 256 kbps”.

Figure 2.1: The Basic Structure of Policy-based Network Management System
As illustrated in Figure 2.2, a policy contains one or more policy rules that are evaluated in a specific order, i.e. policy = \{policy rules | policy rules, policy rules\}. A policy rule defines conditions that trigger actions. A condition is the criteria such as who (users), what (systems, applications), or when (time, day of week, date). An action contains operations governed by the conditions in a policy rule. For QoS and security policies, a typical action is to provide a networking service (e.g. provision bandwidth, configure a class of service, allow access or usage, etc). For instance, we can implement the following setting as policy:

*Provide the VoIP service for authorized users between authorized points, but only at approval time.*

Policy rule can translated as:

*IF user IN Approved Users
AND service IN VideoServices
AND source IN VideoSources
AND destination IN VideoDestinations
AND time IN ApprovedTimePeriods
THEN provide VoIP*
The Policy condition can be:

*If the user is a member of an approved group to access services AND the VoIP is one of the supported services group AND the source of the request is approved (in the VideoSource group) AND the destination is approved (in the VideoDestinations group) AND the time requested is allowed (in ApprovedTimePeriods)*

The Policy action is:

*If the conditions are satisfied THEN provide the user with VoIP having a QoS defined by VoIP service*

### 2.1.1 Negotiating QoS with Service Level Agreement

While policy is used to establish QoS configuration within the network, Service Level Agreement describes the QoS requirement of user applications in high level metrics, for instance, VoIP and video-conferencing. SLA is used as a media for end-user to negotiate (grant/reject) QoS with Internet Service Provider (ISP). SLA defines operational characteristics in terms of Service Level Objectives (SLOs) which are the metrics to enforce, police, and monitor the service level. The SLA usually is written in high-level terminology. SLO offers a more specific metrics in support of the SLA, i.e. the priority of queuing, and bandwidth allocation.

PBM sits at the current gap between SLA establishment and its SLO enforcement, translating high-level QoS requirement into specific network configuration. For example, if a network administrator desires to ensure bandwidth for VoIP traffic,
SLA can indicate the service requirement is *VoIP QoS* and a policy can be defined action with SLO as 'set Priority: = high and bandwidth = 1 Mbps'.

### 2.1.2 Allocating Resource with Bandwidth Broker

A Bandwidth Broker (BB) [32] is a type of policy server that determines the allocation of network resources, especially on demand. BB also operates as a resource monitor of the network within its policy domain, updating information of current state of the network. It records how much of the resources in network devices and links has been used, and how much is left. Hence, BB knows the current state of network bandwidth utilization and maintains this state information in order to calculate the future usage of the network resources. According to the current bandwidth state information and policies in the static repository BB responds to the router (PEP) requests with a decision to enforce network configuration. Those requests from the edge routers will also be recorded in the data storage. The ultimate goal of a bandwidth broker is to provide end-to-end services across the Internet. This means an inter-domain negotiation between BBs in neighbouring domains are required to establish a connection through multiple policy domains.
The basic functions of a Bandwidth Broker are depicted in Figure 2.3 and they are listed in the following.

**Policy-Decision:** This involves retrieving policies, processing policy rules, receiving interface descriptions and policy requests, determining which policies are relevant, and returning policy decisions.

**Monitoring:** This is a real-time monitoring of the network and its constituent devices. This involves checking network health, whether policies have been satisfied and tracks any abnormal activities by users. Dynamic network state is needed for decision-making in PDP. Also PEP is required to report to PDP and verify that the network policies are properly accomplished.

### 2.1.3 Executing Policies in Policy Enforcement Point

Policy enforcement involves PEP applying policy actions according to PDP. The policy actions can be executed in one of the following schemes.
**Packet Marking:** Use a priority value (IP Precedence, DiffServ, and/or 802.1p fields) to define a class of service for an application. The conception of service classes for various application flows is a key principle in policy-based QoS management. Whether using IETF DiffServ standards, or using the classical IP Precedence levels, application traffic can be marked to receive specific treatment as it flows across a service provider network, and enterprise network.

- **Class of Service (CoS):** Establish a set of classes for different network flows. Each class is defined by specific bandwidth and delay characteristics.

- **Differentiated Services (DiffServ):** Indicates per-hop behaviour (PHB) of packet treatment. Some PHBs are being standardized (e.g. Expedited Forwarding); and others can be defined by service providers.

**Queuing:** Use queuing and scheduling to differentiate the packet treatment and provide QoS control. Different queues can be assigned with a range of bandwidth shares and priorities to provide required service levels.

- **Class-based Queuing (CBO):** A per-aggregate packet scheduling method to provide differential treatment to traffic classes. Packets are divided into a hierarchy of classes and each class is assigned to a set of bandwidth priorities.

- **Weighted Fair Queuing (WFQ):** A per-flow packet scheduling method that divides traffic into high and low priorities based on the volume of packets.
2.2 Benefits of Policy-Based Management

Other than envisioning QoS-enabled network, policy-based management can also provide a better control of network and service management.

*Provide services control for network security*

PBM can be used as an admission control throughout the network. It can use the predefined policies to control the access of users, groups, and applications and to prioritise their usage at any given time.

*Improve Response Times for Important Applications*

Network administrators can allocate network resource to improve performance of the critical applications. For instance, financial departments can be authorized to higher priority in access to the network during the last 15 minutes before market close for stock activities and cash transactions.

*Optimize use of the current network infrastructure and reduce cost*

A bandwidth allocation and utility plan can reduce the need to add expensive WAN bandwidth, and improve efficiency of the personnel who manage it. A combination of selective bandwidth addition plus the use of policy management can reduce the cost for both the short- and long-term network growth.

2.3 Issues and Concerns

As mention in Chapter 1, PBM in a real network requires control of a large number of network entities (PEP). In order to handle the large amount of policy
requests and monitoring tasks, more PDPs are required to be deployed into the network to share the increased amount of workload. However, information that relevant to one PEP may affect the decision making for other PEPs. Coordination and inter-communication among the multiple PDPs is required to provide a consistent policy management in the network. This introduces bigger problems in intra-server coordination and also dramatically increases the complexity of the service model. The situation can only be deteriorated by introducing cascading negotiation of Service Level Agreement between domains for cross-domain QoS requirement. Hence, the traditional PBM framework has a fundamental structural limitation of scalability.

One of the major obstacles in providing end-to-end QoS is the heterogeneous environment in the Internet. In most of the networks, Network Interface Cards (NICs), routers, switches, and traffic shapers come from a variety of vendors. PBM provides a standard infrastructural framework for various QoS mechanisms. However, for controlling multi-vender network devices, the policy server is still required to install various control interfaces. For instance, vendors may develop their own extensions or customized versions of COPS protocol for their network devices. These customized protocols may support different feature sets or limited QoS controls on the network. Therefore network administrator is still required to install these customized management interfaces on the policy servers and on the numerous managed network entities (PEPs).
2.4 Summary

Policy-Based Management simplifies and automates the network configurations, allowing dynamic allocation of network traffic resources for QoS with predefined network policies. We studied two major concerns of PBM, which are respectively the poor scalability of traditional framework and the maintenance problem in the heterogeneous network environment.
Chapter 3

ACTIVE NETWORKS

Active network (AN) [8][9] is a novel approach to network architecture, in which the routers in network perform customized computation on the message flowing through them. These networks are active in the sense that nodes can perform computations on, and modify the packet contents [10][11]. In addition, this processing can be customized on a per-user or per-application bases. In contrast, the role of computation within traditional packet networks is extremely limited. The legacy routers can access and modify only the header of a packet.

Compared to traditional network, the objectives of active network is to offer programmable paradigm which allows network nodes to perform dynamic network customization. In particular, active network provides the ability to have a network adapted to the application needs and can greatly increase the potential attributes and services of network application. This program-based approach provides a foundation for interpreting networking systems as the composition of many
smaller components with specific properties: services can be distributed and configured to meet the needs of the applications.

Figure 3.1: Packets Processing within Active Network

In an active network [19], network messages can customize service or configure network elements. As shown in Figure 3.1, active node can perform specific computation on the traversing messages in their execution environment. If those messages travel through legacy network node, they are treated in the ordinary way without processing their contents.

3.1.1 Two Approaches in Active Networking

In this section, we introduce two approaches [9] to realize active networks. These two approaches can be distinguished by whether programs and data are carried in separated messages.
Capsule approach uses miniature programs that are encapsulated inside a packet and executed at each node along their path. User data are embedded within these capsules, in much the way as the contents of a page are embedded within a fragment of PostScript code. The capsule contents are extracted to an execution environment where they can access node resources based on the behaviour of the capsule and its security level. The capsule can leave a state before it leaves the node, and hence can change the configuration of an active network element.

Discrete approach provides an alternative way to support program loading. In this case, the message processing is architecturally separated from the program injection into the node [8]. This preserves the distinction between in-band data transfers and out-of-band management channels found in the traditional network. Separating the injection of programs from the processing of messages can be particularly attractive to network administrators because this mechanism is valuable for precise control over critical program loading.

3.2 Active Signalling Protocol (ASP)

Active Signalling Protocol (ASP) [29] is an active network architecture models among many other approaches [21-26]. ASP is part of the Active Reservation Protocol (ARP) project funded by DARPA (Defence Advanced Research Project Agency) research community. The premise of the ASP project is to explore the use of active network technology for network control protocols. That is also the major reason for us to use ASP as the building blocks of the active network environment to develop a new policy-based management framework for QoS control.
The architecture of ASP is illustrated in Figure 3.2. The lowest level of an active node is Node OS, which allows concurrent execution of multiple execution environments. The Node OS protects the node and isolates execution environments by enforcing boundary and resource limitation. The layer that provides the programmable framework is the ASP Execution Environment which supports the dynamic installation and execution of active application. Each active application (AA) is programmed and executed on execution environment (EE). Execution environment is capable in supporting multiple simultaneous active application executions.

The ASP execution environment (ASP EE) is implemented on top of Java Virtual Machine (JVM) providing a platform-neutral coding environment and reliable runtime system. Different applications executing within the EE are isolated, and the standard Java `sand-boxing` severely restricts the mischief that an AA can cause. Two most important functionalities supported by the ASP EE are dynamic program loading and caching of active code. Each active packet in ASP architecture contains data structure that indicates the primordial Java class name and search paths from which application classes can be loaded. Between the
discrete and capsule approaches of active network technology, ASP EE only supports the out-of-band (discrete) loading of program code. This is because the implementation of real network control algorithms is typically too large for an individual capsule. Since ASP EE is specifically designed for activating complex network control protocols, it supports additional features which are not found in other active network execution environments.

- Sharing program code: applications for real signalling protocols are usually large and complex. However much common codes can be found among the same type of service applications. Supporting common Java byte code sharing among different applications, ASP EE can significantly reduces the memory footprint and eliminates redundant loading of common program codes.

- Dynamic class binding: ASP supports Name Mapping and Version Resolution [29] for compatibility and version specification. During dynamic class loading, the primordial Java class name indicated in the active packet is mapped to a versioned name. This mapping is a selection among different functionality codes and each of them may identify different extensions or feature sets in the protocol implementations. After the compatible extension code is selected, the versioned name is further resolved to find the "latest" version of the code.

- Soft-state storages: active applications are allowed to maintain multiple soft-state tuple spaces (name-to-object mapping) in the state repository of the ASP EE. The soft-state requires periodically refreshed otherwise the contained state is discarded. This approach of soft-state storage can avoid the dangling state after the hardware and software failures. Moreover, periodic refresh makes the state storage more adapt to network changes such as new network routes.
3.2.1 Example of ASP Active Applications

Some sample active applications are included in the ASP EE release 1.3 as demonstrations of its protocol programming interface and basic functionalities. Among those, Traceroute is a simple and excellent example to illustrate the operations of an active application in ASP EE.

The traditional implementation of traceroute is to send out either UDP or ICMP ECHO packets with a TTL (Time To Live) of one, and increments the TTL until the destination has been reached. By printing the gateways that generate ICMP time exceeded messages along the way, it is able to determine the path packets are taking to reach the destination. Figure 3.3 illustrates the signalling of traceroute in traditional network.

![Figure 3.3: Traditional Traceroute Signalling](image)

In ASP, instead of probing the network with an incremental TTL, the function of traceroute is implemented and distributed inside the network. Figure 3.4 shows the operation of traceroute in active network. A traceroute packet containing a field of number of hops (nop) is sent out to the destination. The intermediate
nodes along the path will increment the number of hop field and send back a message with the number of hop to the end host.

![Figure 3.4 Operation of Traceroute in Active Network](image)

### 3.3 Active Benefits in Network Management

**Heuristic Network**

Active networking (AN) solutions attain the outlook that the entire network is programmable and constitute a virtual computational network. AN-based protocols introduce network-based control of protocol function which leads to simple and efficient solutions to network problems. That is because network management is no longer limited on deploying protocols on the edge of the network, where probing is necessary to determine the current state of the network. Active routers can be better positioned to act dynamically, monitors the network and adapts to the protocol when the network state changes. A small module of customized monitoring and diagnostic programs can be injected locally or nearby the observing network devices. Use of active network results in significant reduction of signalling overhead as network elements can self-monitor and trigger
heuristic configuration [10]. Moreover, AN-based solutions react faster to the changing dynamics of network [17]. This shows active network elements are superior in scalability than the end-hosts model.

**Distributed Management**

Other issues that drive the research of network management system on active network are proliferation of data and heterogeneous environments [18]. Managing numerous vendor and function in network device requires a large quantity of data from the network; data that must then be analyzed before management activities can be initiated. Instead of one centralized and large server that encapsulates the complete intelligence of the system, a number of relatively small systems can be deployed in a cooperative effort to resolve a problem. For instance, provisioning services (i.e. DiffServ) is a complex process involving several parties to establish QoS. Active network can help dynamically configure the device attributes and install/update the required control software components in the network.

**Code Migration**

Active network introduces the dynamic program deployment in the network by a network class loader mechanism. The significant benefit is that no pre-calculation is required to determine the strategic point in the network in which programs should be injected. The location of the network server should dynamically depend on the current state of the network. For example, network manager may be migrated to another location, if the failure rate in accessing its services or network latency is unacceptable.
3.4 Other Active Applications

Congestion Control
Most congestion control mechanisms detect congestion when packet loss occurs, and upon timeout or receipt of duplicated acknowledgement. Faster feedback reduces the number of packets transmitted into the network during congestion. In AN, active routers can calculate the window size and selectively discard travelling packet during congestion [12][13]. These schemes reduce the degradation of the service and also allow for a faster response to congestion problems within a network.

Mobile/Wireless Networks
Active network provides an environment for dynamic program deployment, which is very suitable for dynamic topology network such as mobile network. This is crucial in mobile network due to the host movement. In AN, services can be dynamically deployed and shut down according to the need of network.

Multicasting
Research [14] has been focused on how to deliver differential QoS to the multicast subscribers according to their local available resource permission by filtering [15]. For instance, in net conferencing, data is adapted based on receiver capabilities, while reducing retransmission traffic in multicast message by caching [16].
3.5 Summary

The active network approach allows the distribution of management tasks in the network. It enables more efficient network monitoring, shorter control loops, deletes long haul dissemination of redundant and unimportant information, and facilitates new network applications. Specially, active network introduces code mobility to support dynamic configuration and migration of service applications.
Chapter 4

THE ACTIVE BANDWIDTH BROKER FRAMEWORK

As discussed in Chapter one, the traditional policy-based management (PBM) model has fundamental limitation on scalability. In this chapter, we propose a new architectural framework for the PBM, called the Active Bandwidth Broker (ABB). The implementation of ABB framework is built upon an active network. The ABB framework replaces the standard (shown in Figure 4.1) of a poor-scalable client/server PBM model with a dynamic distributed framework in an active network. With active network technology, ABB introduces the advantage of code mobility, on-the-fly computation, and heuristic network monitoring to overcome the scalability barrier of PBM.
Compared to the COPS model described in Section 2.1, the ABB framework evaluates a policy request in a more dynamic fashion, where the whole network functions as a cooperative system of bandwidth brokers. As shown in Figure 4.2, the location of the policy decision point (PDP) is referred to as the Primary Active Bandwidth Broker (P-ABB), which is responsible for making final decisions on policy requests and for storing decision records. Other active nodes in the network are referred to as the Secondary Active Bandwidth Brokers (S-ABBs), which help to share the workload of the P-ABB in making policy decision.
ABB is a distributed PBM framework, in which some functionalities of the policy server can be carried by the S-ABBs. Besides policy decision making, a policy server has three major functionalities: 1) retrieving policy rules, 2) resolving policy conflicts, and 3) observing current network state. In the ABB framework, these three functions can each form a separate program that flows to and executes in the S-ABBs. The S-ABB at the edge of the network can retrieve policy rules on behalf of the policy server. When policies are returning from the repository to the P-ABB, S-ABB along the path can resolve the policy conflicts. S-ABB can also monitor the managed network devices to report local resource allocation, and network device failure; hence eliminating periodic network probing from the policy server. This decentralized framework also provides better scalability in PBM.

Active network technology introduces server mobility [30] in the ABB framework. A mobile P-ABB can avoid the tragedy breakdown of a centralized management service, and relocate a new server as a backup service. P-ABB can also detect the incoming congestion throughout the network monitor, and avoid the poor performance by migrating to a low traffic location. In other words, S-ABB in the ABB framework can convert to take on the role of P-ABB. The mobility of P-ABB and dynamic service backup can greatly improve availability and robustness of the PBM framework.

ASP (Active Signalling Protocol) also supports dynamic loading mechanism and version resolution, which can find the latest version of the program [47]. This mechanism provides a solution to the major obstacle of QoS management, which is the laborious configuration in a heterogeneous network environment. A network usually consists of network devices from different vendors, which support different QoS control and provides different management interfaces. Multiple
versions can also result from customization of features as well as from the normal software upgrade cycles. In ABB framework, administrator can upload the service packages or new version of management interfaces to a code server. Active network allows dynamic loading of these programs wherever they are needed. This property in ABB system can tremendously simplify the difficult administrative work in providing QoS and provide flexibility in server installation.

4.1 Implementation of ABB

The Active Bandwidth Broker is implemented in Java and constructed on the defined protocol programming interface (PPI) from ASP (Active Signalling Protocol). The structure of the source code is illustrated in Figure 4.3.

![Figure 4.3: Structure of ABB Implementation](image)
As shown in Figure 4.3, the implementation of ABB framework includes four types of protocols: 1) Service Level Agreement (SLA) message used by the end users to specify the QoS requirements, 2) Common Open Policy Service (COPS) protocol used to request/reply policy decision, 3) Lightweight Directory Access Protocol (LDAP) used to search/retrieve policy rules from policy repository, and 4) a customized location protocol (discussed in Section 4.3.4) to indicate the location of P-ABB and to control its migration. According to their functionalities, the implementation of the ABB model is also separated into four Java packages. These java packages define the handling and format of those protocol messages in the ABB framework. SLA package defines the format of SLA messages and contains client UI (User Interface) to construct and launch SLA requests. LDAP package employs the Netscape Directory SDK 4.0 (Software Developer Kit) to develop an interface for retrieving the policy information stored in the LDAP directory. Net Monitor package provides network information format for network monitoring report. It should also include a network monitor program to observe current network state. However, at the current stage only a traffic information generator is used to simulate the network state information. ABB package is the core of our project, which defines the COPS protocol messages and includes a daemon program running in each ABB node. This daemon is called Active Bandwidth Broker Daemon, ABBd. The task of ABBd is to identify the message type of the incoming packet and to create a corresponding handler to process and respond to the event. In Figure 4.4, the position of ABBd in the active node structure is shown.
Built upon the Active Signalling Protocol (ASP) execution environment, ABBd can be installed dynamically across the network to install a new ABB node. The implementation of ABBd contains separate message handlers to handle different types of messages. The program in the ABB package for parsing messages and constructing handlers is shown on the following. There are 11 handlers for different message types and each of these handlers can be loaded independently in the desired location of the network. This dynamic and distributed design is the key element of ABB framework to achieve better scalability of the management model.

```java
class ABBUtil implements ABBConf {
    ...
    TaskHandler handler = null;
    switch (msgType(buffer, offset)) {
        //SLA handler
```
case SLA_REQ;    handler = new ClientHandler (...);  break;
// COPS handlers
case COPS_REQ;   handler = new COPSReqHandler (...);  break;
case COPS_DEC;   handler = new COPSDecHandler (...);  break;
// LDAP handlers
case LDAP_REQ;   handler = new LDAPReqHandler (...);  break;
case LDAP_RESULT; handler = new LDAPResultHandler (...);  break;
// Monitor and location message handlers
case PDP_BKUP;   handler = PDPDBkupHandler (...);  break;
case PDP_ADDR_REQ; handler = PDPAddrReqHandler (...);  break;
case PDP_LOCATION; handler = PDPLocaitonHandler (...);  break;
case PDP_KEEPALIVE; handler = PDPKeepAliveHandler ();  break;
case PDP_RELOCATE; handler = PDPRelocateHandler (...);  break;
case PDP_MONITOR; handler = PDPMonitorHandler (...);  break;
...
}  

4.2 Components of ABB

The ABB framework requires a cooperative network to evaluate a policy request. The main entities found in the ABB model are P-ABB, S-ABB, Edge Router (PEP), and a LDAP-enabled Policy Repository. The details of each entity of ABB framework is described in the subsections.
4.2.1 Edge Router/PEP

Edge Routers (ER) of the network will be the interface between end users and network management. As shown in Figure 4.5, Resource Allocation Request (RAR) comes from an end user in a format of SLA. This request is translated and converted into COPS messages by the Edge Router. The approval or denial of the request comes back in the format of a COPS decision message from the P-ABB. Edge router then operates as a Policy Enforcement Point (PEP) to enforce the policy action on the network entity, for instance changing the priority of certain packet in the queue or allocating more bandwidth to a data stream.
4.2.2 P-ABB/PDP

P-ABB is a unique and centralized policy decision point in a policy domain. Its management repository maintains the network status and resource allocation of the current network. With associated policies retrieved from the policy repository and the information of network status, a COPS decision response containing the policy actions is sent to the corresponding network elements (PEP) to establish the configuration. In our current development stage, this management repository is implemented as a hash table in the P-ABB.

Other aspects could be found in the P-ABB component including an administrative interface for manually configuration, an interface for bilateral inter-domain SLA negotiation and a module for intra-domain P-ABB coordination.

4.2.3 S-ABB

All active nodes in the ABB framework can operate as S-ABBs. As illustrated in Figure 4.5, S-ABB works as a helper for P-ABB in the decision-making processes, such as search filtering and conflict resolution of policy actions. Moreover, S-ABB can monitor the current resource allocation and the network device status.

Policy conflicts can happen when multiple policies exist for a QoS request. In Figure 4.6, it shows two rules for solving the policy conflicts in our implementation.
1. If two policy rules are relevant to the QoS request and their validation times cover the whole requested period, the policy with higher bandwidth allocation for the policy decision is used for the policy decision.

2. If both policy rules only cover part of the requested period, the policy with lower bandwidth allocation applies.

When S-ABB is located on the edge of the network, the incoming SLA request is converted into COPS messages by the edge router. At the same time, S-ABB extracts LDAP policy queries from the request and fetches the policy rules on behalf of the P-ABB. S-ABB can also perform as a backup policy decision point. Dynamic program loading enables S-ABB to be upgraded to a P-ABB once the policy decision function is loaded into the S-ABB.
4.2.4 Policy Repository

Policy repository is used to manage and store policy rules applied in the local domain. ABB uses a LDAP-enabled network directory described in Section 2.1 as the policy repository. LDAP enabled directory is well-suited for storing relatively static information such as policy rules. Policy rules stored in the directory can be retrieved by P-ABB via LDAP queries.

The implementation of LDAP repository makes use of the open source software -- OpenLDAP [48], which provides a standalone LDAP server and an LDBM backend database system. The LDAP Data Interchange Format (LDIF) is used to represent LDAP entries in a simple text format with technical specification in [49]. This format is used to construct the basic directory structure and policy information in the LDAP database. The LDIF file used in the configuration of LDAP server database is shown in appendix A.

In the ABB framework, each policy condition is defined as a quadruplet value indicating applicable policy actions. As shown in figure 4.7, the quadruplet condition includes the source and destination addresses with start and end times of the validation period in the format of (src/dest-addr – start/end-time). In order to simplify the database structure, policy actions are merged into the directory of policy rules, which contains two values: policy name and bandwidth. The policy rules are divided into four classes with associated bandwidth of 10M, 1M, 100k, and 56k respectively.
The Netscape Directory SDK 4.0 (Software Developer Kit) is employed to develop the interface for access, update, and management of the information stored in the LDAP directory. The SDK fully supports critical functions such as user authentication, wildcard search, and exceptions handling. An interface for LDAP repository, called *LDAP transceiver* is developed to receive LDAP queries from the ABB framework and to reply policy rules in the format of ABB messages. Figure 4.8 shows the basic operations of the LDAP transceiver.
4.3 Protocols

This section describes the implementation of protocols in the ABB framework and the coordination and communication among the major ABB components: P-ABB, S-ABB, PEP, and Policy Repository. Each ABB message consists of a common header followed by a number of typed objects. This message structure is adapted from the COPS standard message format. The common message header is shown in Figure 4.9, and all the object types in the protocol messages are listed in Appendix C.

The protocols associated with the ABB frameworks are SLA, COPS Protocol, LDAP and Location Protocol which indicates the P-ABB location. The Messages can be distinguished by the client-type field specified in the common header. The definition of each client-type is shown in Table 4.1. In general, Service Level Agreement (SLA) is used to negotiate QoS level between the end users and the Internet Service Providers (ISP). It is a protocol between the end user and the ABB model, specifying user QoS requirement. Common Open Policy Service (COPS) protocol is used to communicate between P-ABB and the policy enforcement points for establishing policy and admission control. Lightweight Access Directory Protocol (LDAP) is used to search and retrieve policies stored in the network directory. Finally, the Location Protocol is a customized protocol used to trace the P-ABB location and to control its migration.
Figure 4.9: Common header for ABB message, number shows the byte order

The fields in the header are shown in the following:

**Version**: 4 bits
Current version is 0x1.

**Flags**: 4 bits
This flag is set to 0x0 for all the messages in the prototype.

**Op Code**: 8 bits
It is the operation code of the message, and it defines the response to the message.

**Client-type**: 16 bits
The Client-type identifies the type of message. Interpretation of all encapsulated objects is relative to the client type.

**Message Length**: 32 bits
It is the size of the message in octets, which includes the common header and all the encapsulated objects. Messages must be on a 4 octet alignment.

<table>
<thead>
<tr>
<th>Client type</th>
<th>Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SLA</td>
</tr>
<tr>
<td>1</td>
<td>COPS</td>
</tr>
<tr>
<td>2</td>
<td>LDAP</td>
</tr>
<tr>
<td>3</td>
<td>Location Protocol</td>
</tr>
</tbody>
</table>

Table 4.1 Client-type for ABB Protocol Message
4.3.1 SLA Message

SLA describes the contract of the QoS level between end users and ISPs. End users construct the SLA specifying the required level of QoS and send it to the edge routers (PEP) in ABB framework for processing. SLA is defined as a high-level of QoS. These high-level QoS specifications will be interpreted by the ABB and corresponding policies will be enforced in the network.

In order to simplify the model, QoS is defined only in terms of the bandwidth (kbps) requirement in our implementation. SLA includes a request identity object that consists of the source and destination addresses of the requested path, validation time of the agreement and the bandwidth requirement.

4.3.2 COPS Message

In the ABB framework, COPS protocol is built upon the UDP transport layer instead of the standard implementation on the TCP. One of the main reasons for the UDP implementation of COPS is that ASP does not have a complete support for the TCP transport layer in their released version 1.3. It is an experimental approach for the COPS implementation. This UDP approach eliminates most of the control messages between the enforcement points and the policy servers. Since P-ABB is no longer responsible for maintaining multiple (potentially hundreds or thousands) TCP connections, a policy server is able to handle more PEPs in the network. This UDP approach not only simplifies the implementation of the COPS protocol but also improves its scalability and reducing cost of employing COPS in high-power servers. The disadvantage of COPS protocol on UDP is that an additional control mechanism is required to provide reliable transmission for large
amount of information transfer [35][43]. In an active network, the out-of-band loading mechanism can be used to support the reliable data transfer.

Request (REQ):
The COPS request message is formed in the PEP and used to report the required policy condition (source and destination addresses, validation time and bandwidth requirement). It also gives the P-ABB ability to send back decisions to establish policies in the network. The COPS request used in the ABB development framework consists of three objects: handle object, context object, and client specific information object. The handle object defines a unique identity to refer to a request state and the context object is used to specify the type of event that triggered the request. In our implementation, handle object is used to indicate the end user of the request with a text message <user>@<address>. The type of event in context object is set to incoming admission request. The client specific information object contains the information extracted from SLA message.
This information includes source/destination addresses of requested path, validation time, and requested bandwidth.

**Decision (DEC):**

COPS decision message is formed by the P-ABB and used to reply to the policy request from a PEP. The message consists of a *decision object* which specifies the policy decision and a *client specific information object* extracted from the COPS request message to identify the request.

### 4.3.3 LDAP Message

LDAP messages are used to search and retrieve policies from the LDAP enabled directory. The conditions for policy search are the source/destination addresses of the designed path and the policy validation time. As shown in Figure 4.11, policy retrieval is handled by the S-ABBs located at the edge of the network in the ABB framework. The result of the policy search is returned to the P-ABB for decision making.

![Figure 4.11: the LDAP Message in ABB Model](image_url)

49
**LDAP Search message:**
The LDAP search message is used to search policies matching the request condition. The LDAP search message contains one object, which is the search condition object with the information of source/destination addresses and validation time.

**LDAP Result message:**
The LDAP result message contains the policies retrieved from the policy repository. These policies are defined in the Policy Object. The ABB LDAP result message can contain one or more Policy Objects, depending on the number of policies retrieved. Each Policy Object varies in length.

### 4.3.4 Location Protocol Message

ABB framework supports dynamic loading of the P-ABB points in the network. This provides a high level of flexibility and scalability in system design. At the same time, this distributed model needs a service discovery protocol to coordinate the system. In our implementation, the Location Protocol is used to indicate the current site of P-ABB and to manage its migration. This simple protocol is adapted from the Service Location Protocol (SLP) [50].

A Directory Agent (DA) is introduced in the ABB framework to register and keep track of the P-ABB location. The DA is responsible to advertise the location of the P-ABB to all S-ABB points and ensure the availability of P-ABB service. In general, when a new S-ABB joins the ABB framework, it sends a location request message to DA to discover the location of the P-ABB. When a new P-ABB is invoked in the domain, it first registers itself to the DA and DA in turn notifies all S-ABB points with the new location of the P-ABB.
**PDP Address Request:**

PDP Address Request message is used to acquire the current location of the P-ABB. S-ABB advertises these request messages to discover the PDP and DA location. It contains a TTL (Time to Live) field, which indicates the max number of hop the packet can travel before it is discarded.

**PDP Location:**

PDP Location message contains the location of the P-ABB. DA sends this message in two situations. First situation is to reply to a *PDP Address Request* message from an S-ABB. Second situation is when a new P-ABB is registered to a DA; the DA will use this message to broadcast the new P-ABB address to all S-ABB points. S-ABB can also discover the DA address from this message header.

**PDP Keep Alive:**

*PDP Keep Alive* message is used to keep contact between P-ABB and DA. If DA fails to access P-ABB for a period of time, a backup procedure is used to dynamic load a new P-ABB point into the network and to continue the policy decision service.

**PDP Relocation:**

This message is used to register the new location of P-ABB. P-ABB sends this message to DA and initiates P-ABB relocation. This message is used to perform the P-ABB migration in the network. This relocation mechanism is used to avoid localized congested and malfunctioned network.
**PDP Backup:**

This message contains dynamic information in P-ABB, which is the current policy states established in the network. This information is send to the new site of the P-ABB for relocation. The format of backup data is not defined yet. A text message is used at this stage.

**PDP Monitor:**

PDP Monitor message is used to feed P-ABB with the current network status. In our implementation, network status includes bandwidth allocation, delay and jitter information for each link. This information can be used in policy decision making and congestion detecting. If a localized congested area occurs near the P-ABB point, P-ABB can invoke a migration to a new location according to this network monitor information. Since a real time network monitor is not completed at this stage, a network traffic information simulator is used in the evaluation.

### 4.4 Operations of ABB

#### 4.4.1 Distributed Policy Processing

In ABB framework, the evaluation process of a policy request is distributed within the network. S-ABBs are positioned in strategic points of the network to handle service requests and share the policy management workload. As shown in Figure 4.12, the service request procedure can be divided into three steps in the process.
- Step 1: Incoming SLA with the QoS request information is identified at the edge of the network. This QoS information is then converted into a COPS request and sent to the P-ABB. At the same time, S-ABB extracts the LDAP policy search message and retrieves the policy rules on behalf of the P-ABB.

- Step 2: As Policy rules are returning to the P-ABB, S-ABBs along the traversing path can perform optimization and filtering the policy rules. On-the-fly computation of the policy data can lighten the workload of policy processing and expedite the decision making process in the P-ABB.

- Step 3: Upon receiving the policy rules, P-ABB makes the final decision on the policy request. A COPS decision message is replied back to the corresponding PEP, where policy actions will be enforced.

![Figure 4.12: Policy Request and Decision handling in ABB Framework](image-url)
4.4.2 Server Migration

For controlling server migration, the protocol for P-ABB relocation is shown in Figure 4.13. There are two ways to invoke a P-ABB migration. The first one is that when a localized congestion is detected, the P-ABB initiates the migration and determines the new location for the P-ABB. In this situation, the migration process can be separated into three steps.

- Step 1: P-ABB sends a *PDP Relocation message* to DA to initiate the relocation and back up the current network states using *PDP Backup message* to the destination of migration.
- Step 2: After DA registers the new location of P-ABB, an advertisement of the new P-ABB address will be sent out to the network. To simplify the implementation, the message advertisement uses the Reverse Path Flooding algorithm to broadcast the message.
- Step 3: Once the function used for policy decision is dynamically loaded, the destination of the relocation will be converted to a new P-ABB point in the framework.

Another way to invoke a P-ABB migration is when the DA fails to contact with P-ABB with *PDP Keep Alive message*. In this case, DA is the point to initiate the migration and to determine the new location of P-ABB. The choice of P-ABB location is arbitrary in the current development stage. The procedure of migration will start with DA broadcasting the new P-ABB to the network in the step 2 of Figure 4.13. After that, a new P-ABB is activated with the dynamic loading of policy decision function.
4.5 Summary

The Active Bandwidth Broker framework brings two new functionalities into the policy-based management framework. They are 1) the distributed policy evaluation, and 2) the mobility of policy decision point. The distributed design and mobile PDP in the ABB framework are introduced to achieve better scalability and reliability of the policy management. Furthermore, ASP (active signalling protocol) execution environment supports dynamic program loading and version resolution, which allow ABB to have multiple versions of code for different
feature sets and dynamic class binding to find the newest version of source code. The mechanism helps to release the laborious administrative work in providing QoS.
Chapter 5

PERFORMANCE EVALUATION

This chapter shows the performance of Active Bandwidth Broker framework and its advantages over the traditional policy-based management model. The evaluation emphasizes on the two new functionalities of ABB framework, which are respectively the distributed processing of policy decision and the congestion avoidance mechanism with the mobile P-ABB. In order to evaluate the performance of the Active Bandwidth Broker (ABB) framework, a prototype is developed in NIT² laboratory at the University of Toronto. An active network testbed is configured with six active nodes for the test environment. The specification of active node is shown in the Table 5.1.

² Nortel Institute for Telecommunication
Table 5.1: Active Node Specification

<table>
<thead>
<tr>
<th>Active Application</th>
<th>ABBd (Active Bandwidth Broker Daemon)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Environment</td>
<td>ASP EE v1.3 (Active Signalling Protocol)</td>
</tr>
<tr>
<td>Java Virtual Machine</td>
<td>JDK 1.3 (Java Development Kit)</td>
</tr>
<tr>
<td>Node Operating System</td>
<td>Mandrake 8 with Linux kernel 2.4.3</td>
</tr>
</tbody>
</table>

Topology of the test-bed network is shown in Figure 5.1. Node 172.16.5.73 is configured as the P-ABB for final policy decision. The rest of the nodes are set up as S-ABBs, which accept SLA (Server Level Agreement) requests from end-users, and convert them to COPS requests and LDAP policy queries. Directory Agent is located at node 172.16.5.79 to indicate the location of P-ABB and coordinate P-ABB migration with Location Protocol. The configuration of each node is summarized in Table 5.2.

![Figure 5.1: Topology of Active Network](image)
<table>
<thead>
<tr>
<th>Node</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>172.16.5.70</td>
<td>It is set up as an S-ABB (Secondary ABB) at the edge of the network. It can accept SLA (Service Level Agreement) request from end-hosts, and dispatch COPS and LDAP request for policy decision making. S-ABBS work in a co-ordinated manner to reduce the workload in P-ABB (Primary-ABB). It solves conflicted and duplicated policies in each LDAP search. It also operates as the policy enforcement point in the network.</td>
</tr>
<tr>
<td>172.16.5.71</td>
<td>Same as node 172.16.5.70</td>
</tr>
<tr>
<td>172.16.5.72</td>
<td>Same as node 172.16.5.70</td>
</tr>
<tr>
<td>172.16.5.73</td>
<td>It is the initial P-ABB in network system. With static policies from policy repository and dynamic network information, a policy decision can be made and sent back to the PEPs.</td>
</tr>
<tr>
<td>172.16.5.74</td>
<td>Same as node 172.16.5.70</td>
</tr>
<tr>
<td>172.16.5.79</td>
<td>This network node is configured as the Directory Agent (DA) and the class loading source. It functions as DA for location protocol to indicate the location of P-ABB (Primary ABB) and to provide server migration to avoid service degradation due to network malfunction. Class loading source is a designated point of active network, which used to dynamically update and load function over the network.</td>
</tr>
</tbody>
</table>
5.1 Measurement

For measurement tools, we use a GNU open source program called Ethereal [9] to observe network traffic activity. Ethereal provides a complete interface for analysing network protocols. It allows user to capture data from network in real-time and provides summary and detail information for each packet. The analytical data extracted from experiments includes bandwidth consumption in individual link, the amount of data traffic and dispatching time of each message. Ethereal has a powerful feature, which allows construction of customized filters to measure particular application performances such as response time and packet loss rate. Furthermore, Ethereal provides modules to identify most of the common protocols including COPS and LDAP used in our ABB framework. In the aspect of time measurement, Ethereal may not provide an accurate real-time stamping due to processing overhead. However, it satisfies our purpose in comparing performances in different solutions, as long as all measurements are given the same kind of overhead.

5.1.1 Measurement Metrics

In the following, we define four performance metrics for the experiments.

Request Loss Rate:
This is measured at the edge router. As mentioned in Section 4.3.2, COPS protocol in ABB is implemented on UDP. Therefore, any message loss due to congested network or buffer overload will not be retransmitted. We consider a SLA request successfully reaches S-ABB and fails to receive any policy decision
from P-ABB as a request loss. The request loss rate is defined as the number of request loss divided by the total amount of SLA request messages received by S-ABB.

\[ R_L = \frac{N_{SLA} - N_{DEC}}{N_{SLA}} \]

\( R_L = \text{Request Loss Rate} \)

\( N_{SLA} = \text{Amount of SLA received} \)

\( N_{DEC} = \text{Amount of policy decision arrived} \)

**Response Time:**

It is measured at the edge router. The response time defined as the period of time from when a SLA request is received at the edge router, to the time when its policy decision reply is returned. The response time includes the overhead such as data transport and operating system activities.

\[ T_{RESP} = T_{DEC} - T_{SLA} \]

\( T_{RESP} = \text{Response Time} \)

\( T_{DEC} = \text{Arrival time of policy decision from P-ABB} \)

\( T_{SLA} = \text{Arrival time of SLA from user} \)

**Policy Process Time:**

It is measured at the P-ABB. The policy process time is defined as the period of time from when policy rules arrive to the time a policy decision is made in P-ABB.

\[ T_{PROC} = T_{DEC} - T_{POLICYRULE} \]

\( T_{PROC} = \text{Policy process time} \)

\( T_{DEC} = \text{Departure time of policy decision from P-ABB} \)

\( T_{POLICYRULE} = \text{Arrival time of policy rules from policy repository} \)

**Network Traffic:**

It is measured over the network. The network traffic is defined as the amount of traffic (KByte) generated by ABB messages. This measurement is to show the power of distributed processing of policy rules in ABB framework. When the policy rules are retrieved from policy repository to P-ABB, these policy rules are
optimized “on-the-fly” in the S-ABBs. Therefore, the amount of policy data traffic flowing in the networks will be reduced.

5.2 Performance Results

The performance evaluation is divided into two sessions. In the first session, congestion avoidance mechanism with the mobile P-ABB is examined, and its performance is used to compare with the stationary server model. In the second session, the advantage of “on-the-fly” policy optimization is investigated for the distributed policy management structure. The purpose of this evaluation is to demonstrate the fundamental design advantages of scalability in distributed system of ABB framework over the standard client/server model.

5.2.1 Evaluation of Congestion Avoidance Mechanism

In order to test the performance of congestion avoidance with the mobile server, a localized congested environment is set up and CBQ (Class-based Queuing) is used to simulate this environment. As mentioned in Section 2.1.3, CBQ is a queue scheduling algorithm to differentiate the packet treatment and provide QoS control. Different queues can be assigned with a range of bandwidth shares and priorities to provide required service levels. CBQ is used to organize traffic streams with different QoS levels, which is divided into a hierarchy of classes and each class is assigned to a set of bandwidth priorities. With CBQ, ABB messages are marked with a special QoS level, which indicates the maximum amount of bandwidth allowed. In the experiments, the QoS level of ABB messages is
lowered to reduce the bandwidth allocation to simulate a scarce bandwidth environment during congestion. The patch file used to configure the CBQ is shown in Appendix B.

![Diagram of network configuration]

Figure 5.2: P-ABB Migration in the Congested Network Configuration. The dotted line shows the localized congested link in the network. The blocked arrow shows the P-ABB migration direction of the P-ABB from node 172.16.5.73 to 172.16.5.71.

As shown in Figure 5.2, an active node at 172.16.5.73 is set up as the P-ABB and another active node at 172.16.5.70 is configured as the only edge router. This edge router accepts all SLA requests from end-hosts. The designated link between 172.16.5.79 and 172.16.5.73 is congested and only 1Kbps of traffic is allowed. The average response time is measured over one thousand requests. Three sets of measurements are taken. First, the experiment is performed in a non-congested network. In the second experiment, the designated link is congested but the P-ABB mobility mechanism is disabled. This implies the P-ABB cannot migrate to a less congested area in the network even the localized congestion is detected. In the last experiment, the mobile congestion mechanism is enabled and P-ABB migrates to the new location at 172.16.5.71 to avoid the congestion. The response time is shown in Table 5.3.
Table 5.3: Response Times for Static and Mobile Server of ABB Model

<table>
<thead>
<tr>
<th></th>
<th>Response Times (msec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
</tr>
<tr>
<td><strong>Non-congested Network</strong></td>
<td>172.4293</td>
</tr>
<tr>
<td><strong>Stationary Server in Congested Network</strong></td>
<td>1486.908</td>
</tr>
<tr>
<td><strong>Mobile Server in Congested Network</strong></td>
<td>189.6955</td>
</tr>
</tbody>
</table>

From the observation of experimental results, huge variations of response time are found in the second experiment. This large variance in response time shows an inconsistency of the system performance as a result of the congested network. Histograms of the response time distribution are shown in figure 5.3, 5.4 and 5.5.
Figure 5.3: Histogram of P-ABB Response Time in a non-congested network.

Figure 5.4: Histogram of the stationary P-ABB Response Time in a Congested Network

Figure 5.5: Histogram of the Mobile P-ABB Response Time in Congested Network
In Figure 5.4, a noticeable portion (~6%) of measured response time is extremely high compared to the average. This large response time is due to the traffic delay in the congested network. One important observation for Figure 5.5 is that this poor response time in congested network is eliminated by the mobility of P-ABB. With the mobile server, ABB is managed to avoid the poor response time performance by moving the P-ABB to a non-congested area.

Besides response time, policy request loss rate is measured to evaluate performance with the mobile P-ABB. In this experiment, five hundred requests are generated randomly and sent to each of the six S-ABB points in the test-bed. The same link is congested at various bandwidth levels (bound 10-1kbps). The purpose of this experiment is to investigate the relationship between the congestion level and the policy request loss rate. The measurements of request loss rate are shown in Figure 5.6 and 5.7. In Figure 5.6, by lowering the available bandwidth from 10k to 1k, dramatic increases of the request loss rate are observed in all S-ABB points. However, by employing a mobile server in the network, the congestion effect on request loss rate results can be eliminated. Figure 5.7 and 5.8 show that mobile P-ABB manages to avoid the congested area in the network. Therefore the increasing congestion level in the designated link does not affect the experimental performance with the mobile P-ABB.

Concisely, the mobility of P-ABB enhances the performance of both response time and request loss rate. This proves congestion avoidance mechanism can greatly improve the availability and consistency of policy management service.
Figure 5.6: Histogram of Request Loss Rate at Each S-ABB with stationary P-ABB. It shows the relationship between the Request Loss Rate and the available bandwidth: as the providing bandwidth decreases, the request loss rate increases.

Figure 5.7: Histogram of Request Drop Rate at Each S-ABB with Mobile P-ABB. It shows the dependency between request loss rate and bandwidth is eliminated by the mobility of P-ABB.
Figure 5.8: Comparison of the Effect of Limited Bandwidth to Request Loss Rate between Stationary Server and Mobile Server. It shows that congestion avoidance mechanism with mobile P-ABB releases the degrading effect of limited bandwidth on the performance in terms of request loss rate.

5.2.2 Evaluation of Distributed Policy Management

In the investigation of the advantages with distributed policy management, the policy processing time in the P-ABB is measured for the performance evaluation. The study is focus on the advantage of policy “on-the-fly” processing with S-ABBs. Five hundred arbitrary SLA requests are generated and sent to each of the six S-ABB points in the test-bed. First, all S-ABB nodes in the framework are deactivated. In this case, S-ABB nodes will act as legacy network nodes, which forward the traversing packets without any modification on policy contents. In the second experiment, all S-ABBs are enabled with the “on-the-fly” computation on policy data. The P-ABB policy processing times are measured and the results are summarized in Table 5.4.
From comparison of Table 5.4, an improvement of 11% reduction in the P-ABB process time is observed with the network computing model. The decrease in the P-ABB process time is the result of the “on-the-fly” pre-processing of policy rules. The distribution of P-ABB process time is illustrated in the histograms of Figure 5.9 and 5.10. In Figure 5.9, it shows a fairly even distribution of process time. With the network computing power enabled, a much higher distribution over the lower process time region can be found in Figure 5.10. The enhancement of the P-ABB process time demonstrates the effort of load-sharing with the S-ABBs on-the-fly computation. The distributed design expedites the policy processing time in P-ABB, and hence reduces the P-ABB computation power and resource consumption for each policy request.

<table>
<thead>
<tr>
<th>Without Network Computing</th>
<th>With Network Computing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>Std dev</td>
</tr>
<tr>
<td>8.4833</td>
<td>9.8986</td>
</tr>
<tr>
<td>7.6346</td>
<td>3.2178</td>
</tr>
</tbody>
</table>
Besides the improvement of policy processing time in ABB framework, an optimization of policy traffic is achieved by the “on-the-fly” network computation. The analysis of the policy related traffic flowing in the network is studied and a summary of these traffic data is shown in Table 5.5. The result shows that a 16%
reduction of traffic is observed with the "on-the-fly" network computation. This reduction is rather significant if we consider policy data traffic taking over 40% of the total traffic generated by the ABB system.

Table 5.5: Observation of Traffic related to Policy Rule Searches and Results

<table>
<thead>
<tr>
<th>Policy Search Ordinances</th>
<th>No Network Computation</th>
<th>With Network Computation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Data Rate (packet/s)</td>
<td>7.311</td>
<td>7.38</td>
</tr>
<tr>
<td>Traffic Amount (kbyte)</td>
<td>447.241</td>
<td>384.972</td>
</tr>
<tr>
<td>Traffic Rate (byte/s)</td>
<td>2210.867</td>
<td>1904.18</td>
</tr>
<tr>
<td>Percent of the total Traffic</td>
<td>42.12%</td>
<td>38.15%</td>
</tr>
</tbody>
</table>

The performance results of the ABB distributed system shows a great improvement in policy processing time of P-ABB and a reduction of policy traffic load in the network. This enhancement proves the distributed ABB framework can reduce and share workload of PDP in decision process, and hence provides an advantage of better scalability over the traditional client/server model.

5.3 Summary

Investigating the evaluation results, the ABB framework shows a superior performance with the distributed system design and mobility of policy decision point. In particular, faster response time and lower request loss rate are measured with the mobile policy server in congested environment. This demonstrates mobile server greatly improves the availability and robustness of the system management. On the other hand, the distributed model with "on-the-fly" policy computation reduces the policy processing time in the policy decision point, and optimises the
policy data traffic in the framework. These enhancements show the distributed
design of ABB can achieve better scalability and availability for policy-based
management.
Chapter 6

CONCLUSION

Active networking introduces evolitional changes not only to the concept of network, but also to policy-based management. In this project, we have designed and implemented a new reliable and scalable framework called Active Bandwidth Broker, which integrates the Policy-Based Management (PBM) with active network technology.

Importing the “active networking” concept to the policy-based management brings solutions to some problems that can be solved in an alternative way. In particular, dynamic program loading and version resolution mechanism in active network releases the problem of laborious administrative work of providing QoS in the heterogeneous network environment. ABB framework supports a distributed policy management system for decision making of policy requests. This distributed design has overcome the barrier of poor scalability in the traditional client/server PBM structure. ABB also introduces mobility of policy decision point in the framework, which enables the policy server to migrate and void the poor performance due to congested network. A mobile policy server can evade the
breakdown of centralized management service, and dynamically reload a new server as a backup service.

A prototype of ABB has been evaluated with some performance tests. The results of evaluation agree with the primary goals of the framework design. In terms of congestion avoidance, the result shows a significant reduction of response time and request loss rate with our mobile model comparing to a stationary server in a congested network. Distributed network computation in decision making also reduces policy processing time in policy server and optimise the policy data traffic in the network. In the analysis of ABB framework versus the traditional PBM model from the IETF, we concluded that the ABB framework design provides a highly available and more robust policy management service that scales in large network system.

6.1 Future work

The prototype of ABB in the current stage is by no means a complete model. A real-time network monitor feeding dynamic network state is required. The network state should be available to be pushed or polled from the monitor to the P-ABB to update management information. One should be able to install only a few monitors in strategic point of the network to report filtered and summarised information to P-ABB.

Furthermore, for a successful deployment of end-to-end QoS service [41], an inter-domain negotiation between policy servers is needed. Local network policy only controls the access to the available services within a provider's domain. To
provide inter-domain services, it is necessary for one domain to co-ordinate the other provider domains along the stream. The neighbouring bandwidth broker can accept or refuse the request of the SLA according to the intra domain status and local policy. Several bilateral agreements may need to negotiate for end-to-end QoS. The inter-domain SLA protocol will need to be defined and standardised. There are some researches studied to establish inter-domain QoS such as Super Policy Server (SPS) and Policy Advertisement Protocol [10] (PAP)

Active networking brings dynamic and automation to network and policy management. We shall consider carefully if active network technology should be employed as a potential solution platform. We shall consider the followings: 1) Network operation must not impose significant performance penalties on network; 2) If limited active network supported, it must not degrade the performance of service. A performance test respecting to these concerns as well as the network computation overhead needs to be evaluated. Regarding evaluation of ABB, it would be appropriate to perform tests with typical traffic patterns for provisioning data. However, this traffic pattern information is not available at this stage.
References


[34] “Policy-Based Networking, Products, Design & Architecture,” IPhighway, January 2000


Appendix A

The following is the LDIF data file used in the configuration of the LDAP Policy directory. The data is used to construct the basic directory structure and policy rules used in the performance test.

dn: o=nit.utoronto.ca
objectclass: top
objectclass: policy
objectclass: organization
o: nit.utoronto.ca

dn: ou=policydb, o=nit.utoronto.ca
objectclass: top
objectclass: policy
objectclass: policyGroup
ou: policydb

dn: ou=policyrule, ou=policydb, o=nit.utoronto.ca
objectclass: top
objectclass: policy
objectclass: policyGroup
objectclass: policyRule
ou: policyrule

dn: ou=policycondition, ou=policydb, o=nit.utoronto.ca
objectclass: top
objectclass: policy
objectclass: policyGroup
objectclass: policyCondition
ou: policycondition

dn: ou=policyaction, ou=policydb, o=nit.utoronto.ca
objectclass: top
objectclass: policy
objectclass: policyGroup
objectclass: policyAction
ou: policyaction

dn: cn=goldrule, ou=policyrule, ou=policydb, o=nit.utoronto.ca
objectclass: policy
objectclass: policyRule
cn: goldrule
policyRule: goldrule
pBandwidthAvailable: 10000
policyRuleName: GoldServiceRule

dn: cn=silverrule, ou=policyrule, ou=policydb, o=nit.utoronto.ca
objectclass: policy
objectclass: policyRule
cn: silverrule
policyRule: silverrule
pBandwidthAvailable: 1000
policyRuleName: SilverServiceRule

dn: cn=bronzerule, ou=policyrule, ou=policydb, o=nit.utoronto.ca
objectclass: policy
objectclass: policyRule
cn: bronzerule
policyRule: bronzerule
pBandwidthAvailable: 250
policyRuleName: BronzeServiceRule

dn: cn=rustrule, ou=policyrule, ou=policydb, o=nit.utoronto.ca
objectclass: policy
objectclass: policyRule
cn: rustrule
policyRule: rustrule
pBandwidthAvailable: 100
policyRuleName: RustServiceRule

dn: cn=172.16.5.27-172.16.6.0/100000-300000, ou=policycondition, ou=policydb, o=nit.utoronto.ca
objectclass: policy
objectclass: policyCondition
cn: 172.16.5.27-172.16.6.0/100000-300000
policyRule: silverrule
ptpConditionTimeStart: 100000
ptpConditionTimeEnd: 300000
pSpecialTimeReq: Always
paddHost: 172.16.5.27
paddDest: 172.16.6.0
policyConditionName: 172.16.5.27 connects to 172.16.6.0 valid always.

dn: cn=172.16.5.27-172.16.5.43/100000-150000, ou=policycondition, ou=policydb, o=nit.utoronto.ca
objectclass: policy
objectclass: policyCondition
cn: 172.16.5.27-172.16.5.43/100000-150000
policyRule: goldrule
ptpConditionTimeStart: 100000
ptpConditionTimeEnd: 150000
pSpecialTimeReq: Daily
paddHost: 172.16.5.27
paddDest: 172.16.6.42
policyConditionName: 172.16.5.27 connects to 172.16.5.43 valid Daily for 00:00:00-05:45:59.

dn: cn=172.16.5.27-172.16.5.43/200000-350000, ou=policycondition, o=nit.utoronto.ca
objectclass: policy
objectclass: policyCondition
cn: 172.16.5.27-172.16.5.43/200000-350000
policyRule: silverrule
ptpConditionTimeStart: 200000
ptpConditionTimeEnd: 350000
pSpecialTimeReq: Daily
paddHost: 172.16.5.27
paddDest: 172.16.6.42
policyConditionName: 172.16.5.27 connects to 172.16.5.43 valid Daily for 05:46:00-11:59:59

dn: cn=172.16.5.27-172.16.5.43/100000-150000, ou=policycondition, o=nit.utoronto.ca
objectclass: policy
objectclass: policyCondition
ou: policyrule
ou: policydb
Appendix B

This is the patch file for managing the CBQ in Linux 2.4.3. This includes the setup of hierarchy structure of QoS class and their associated bandwidth priorities.

# 2001 Achint Saxena.
# Last modified 14th May 2001.

# test script for linuxpep.

# NOTE: service is not yet best effort as mentioned below. it's a
# "hard" limit, i.e. bounded. for testing.
# switch off the bounded flag on "rust" rule, when best effort
# service is required to all undefined traffic.

# repeat for all interfaces!

# for eth0 or the "gateway" at the edge router.

# initialize interface with cbq
tc qdisc add dev eth0 root handle 10: cbq bandwidth 100Mbit avpkt 1000

# initialize root class of cbq (take entire bandwidth, and use
# blocks of 1Mbit for classes)
tc class add dev eth0 parent 10:0 classid 10:1 cbq bandwidth
100Mbit rate 100Mbit allot 1514 weight 500Kbit prio 8 maxburst 20
avpkt 1000

# gold. 10000Kbit. but its UNBOUNDED.
# generate platinum (70Mbit of 100Mbit interface)
# higher priority, so this is "unbounded" (can borrow from other
# classes if they're underused).
# weight in 1Kbit to make it easier to add new streams.
tc class add dev eth0 parent 10:1 classid 10:100 cbq bandwidth
100Mbit rate 10Mbit allot 1514 weight 1Kbit prio 5 maxburst 20
avpkt 1000
#silver, 1000Kbit.
#generate gold (uses 20Mbit of 100Mbit interface)
#also unbounded.
tc class add dev eth0 parent 10:1 classid 10:200 cbq bandwidth 100Mbit rate 1Mbit allot 1514 weight 1Kbit prio 5 maxburst 20 avpkt 1000 bounded

#bronz, 250Kbit.
#generate silver (uses 9Mbit of 100Mbit interface)
#bounded. cant borrow.
tc class add dev eth0 parent 10:1 classid 10:300 cbq bandwidth 100Mbit rate 500Kbit allot 1514 weight 1Kbit prio 5 maxburst 20 avpkt 1000 bounded

#rustrule = 56Kbit (modem speed!).
#generate rustrule (uses 900Kbit of 100Mbit interface)
#bounded.
tc class add dev eth0 parent 10:1 classid 10:400 cbq bandwidth 100Mbit rate 1Kbit allot 1514 weight 1Kbit prio 5 maxburst 20 avpkt 1000 bounded

# best effort = 10Kbit UNBOUNDED.
#generate best effort (uses 100Kbit of 100Mbit interface) //all
#traffic should go here by default!!!!!!!!!!!!
#unbounded. //best effort service.. so, this is unbounded.
#testing.. keep it bounded.
tc class add dev eth0 parent 10:1 classid 10:500 cbq bandwidth 100Mbit rate 100Kbit allot 1514 weight 1Kbit prio 5 maxburst 20 avpkt 1000

#fix queueing disciplines for the classes.
tc qdisc add dev eth0 parent 10:100 sfq quantum 1514b perturb 15
tc qdisc add dev eth0 parent 10:200 sfq quantum 1514b perturb 15
tc qdisc add dev eth0 parent 10:300 sfq quantum 1514b perturb 15
tc qdisc add dev eth0 parent 10:400 sfq quantum 1514b perturb 15
tc qdisc add dev eth0 parent 10:500 sfq quantum 1514b perturb 15

#############################################################
# Test cases.
#############################################################
#eth0 is the incoming/gateway interface. (upstream to core #router).
#test, everything from vlan5 is platinum.
#everything else is rust.
#this sets any traffic going to vlan5 as Platinum (later we will #use netfilter to block out any SYN/other connection packets to #(such as UDP).

echo "NOTE: rules with lower preference numbers get higher priority."
#oi.. we can do the src = 128.0.0* thingy. (i.e. out of network source ip).
#we can match on tcp too.

#NOTE: service is not yet best effort as mentioned below. its a "hard" limit, i.e. bounded. for testing.

#first, the default filters on all the interfaces.
#i.e. by default everyone gets the lowest. Note: the lowest i.e. rust is unbounded, so it can behave like #best effort. for specific sources, new filters can be added later. #note: filters are for outgoing traffic from the interface.
tc filter add dev eth0 parent 10:0 protocol ip prio 100 u32 match ip dst 0.0.0.0/0 flowid 10:500

#allow full speed access to the print server(one way.. but keep two ways for now), dns server, etc.
tc filter add dev eth0 parent 10:0 protocol ip prio 49 u32 match ip dst 128.100.244.35 flowid 10:100

tc filter add dev eth0 parent 10:0 protocol ip prio 49 u32 match ip dst 128.100.244.3 flowid 10:100

#NOTE: for demo!!!!
#allow full speed access in high priority rule, use 40.
tc filter add dev eth0 parent 10:0 protocol ip prio 40 u32 match ip dst 172.16.5.73 match ip src 172.16.5.79 flowid 10:400
tc filter add dev eth0 parent 10:0 protocol ip prio 40 u32 match ip src 172.16.5.73 match ip dst 172.16.5.79 flowid 10:400

###########################################
#NOTE: To delete everything use:
#  tc qdisc del dev eth2 root; tc qdisc del dev eth1 root; tc qdisc del dev eth0 root

# look at info:
#  tc -s -d qdisc ls dev eth0, etc.
#  tc -s -d filter show
#  tc -s -d class show dev eth0, etc.
###########################################
Appendix C

The common header and all typed objects used in the ABB protocols are listed. Each of the messages in the ABB framework contains a common header with client-type, which used to distinguish the messages into four groups: SLA, COPS, LDAP and Location Protocol.

Client-type for ABB Protocol Message

<table>
<thead>
<tr>
<th>Client type</th>
<th>Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SLA</td>
</tr>
<tr>
<td>1</td>
<td>COPS</td>
</tr>
<tr>
<td>2</td>
<td>LDAP</td>
</tr>
<tr>
<td>3</td>
<td>Location Protocol</td>
</tr>
</tbody>
</table>

Common header for ABB message, number shows the byte order

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>Flags</td>
<td>Op Code</td>
<td>Client-type</td>
<td>Message Length</td>
</tr>
</tbody>
</table>

The fields in the header are:

Version: 4 bits

Current version is 0x1.

Flags: 4 bits

This flag is set to 0x0 for all the messages in the prototype.

Op Code: 8 bits
It is the operation code of the message, and defines the response to message.

**Client-type:** 16 bits

The Client-type identifies the type of message. Interpretation of all encapsulated objects is relative to the client type.

**Message Length:** 32 bits

It is the size of the message in octets, which includes the common header and all encapsulated objects. Messages must be aligned on 4 octet intervals.

All objects follow the same object format in ABB framework.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length (octets)</td>
<td>C-Num</td>
<td>C-type</td>
<td></td>
</tr>
<tr>
<td>(object contents)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Length:** 16 bits

It describes the number of octets (including the header) in the object. Padding is added to the end of the object if the length of the object does not fall on a four-octet word boundary.

**C-Num:** 8 bits

It identifies the class of information contained in the object

**C-Type:** 8 bits

It identifies the subtype of the information contained in the object
**SLA messages:** For the common header, Client-type = 0 and Op Code = 1. SLA contains only one Request Identity Object, which define the condition of QoS request.

Request Identity Object

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length = 24</td>
<td>C-Num = 9</td>
<td>C-type = 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Address</td>
</tr>
<tr>
<td>Destination Address</td>
</tr>
<tr>
<td>Start time of policy</td>
</tr>
<tr>
<td>End time of policy</td>
</tr>
<tr>
<td>Bandwidth requirement</td>
</tr>
</tbody>
</table>

**COPS Messages:**

**COPS Request:** Client-type 1, Op Code 1, contains three objects: Handle Object, Context Object, and Client Specified Information Object

Format of Handle Object

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>C-Num = 1</td>
<td>C-Type = 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<user>@<address>

Format of Context Object

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length = 8</td>
<td>C-Num = 2</td>
<td>C-Type = 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

R-Type = 0x01  M-Type = 1
R-Type (Request Type Flag)

0x01 = Incoming message / Admission control request
0x02 = resource Allocation request
0x04 = Outgoing Message request
0x08 = Configuration request

M-Type (Message Type)

16 bit values of protocol message types

Client Specified Information Object

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Length = 24</td>
<td>C-Num = 9</td>
<td>C-type = 2</td>
</tr>
<tr>
<td></td>
<td>Source Address</td>
<td>Destination Address</td>
<td>Start time of policy</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>End time of policy</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bandwidth requirement</td>
</tr>
</tbody>
</table>

COPS Decision: Client-Type = 1, Op Code = 2, contains only decision object

Format of Decision Object

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Length = 8</td>
<td>C-Num = 6</td>
<td>C-Type = 1</td>
</tr>
<tr>
<td></td>
<td>Command-Code</td>
<td>Flag</td>
<td></td>
</tr>
</tbody>
</table>

Command Code is defined for the policy decision: 0 - null decision, 1 - admit request, 2 - remove request and Flag is used to trigger error: 0x01 - error notification

LDAP Messages
LDAP Search: Client type = 2, Op code = 1, contain one Search Condition Object

Format of Search Condition Object

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length = 20</td>
<td>C-Num = 9</td>
<td>C-type = 2</td>
<td></td>
</tr>
<tr>
<td>Source Address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination Address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start time of policy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>End time of policy</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

LDAP Result: Client type = 2, Op code = 2, contains one or more than one Policy Object

Format of Policy Object

<table>
<thead>
<tr>
<th>Length</th>
<th>C-Num = 9</th>
<th>C-Type = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S-req</td>
<td>Op Flag</td>
<td></td>
</tr>
<tr>
<td>Start time</td>
<td></td>
<td></td>
</tr>
<tr>
<td>End time</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start time...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The bandwidth field: 32 bits
Define associated bandwidth of the policy.

S-req (Special requirement): 16 bits
Indicate the specification for policy validation time. Always (S-req = 0) means the policy applies forever; daily (S-req = 1) means the policy is
valid on the same period of time everyday; *none* (S-req = 2) means the policy is only valid for the specified period of time.

Op Flag (Optimization flag): 16 bits

Indicate whether the Policy Object is filtrated and examined in the “on-the-fly” optimization.

Start/end time pairs: 32/32 bits

Specify the valid duration of the policy.

**Location Protocol:**

**PDP Address Message:** Client-Type = 3, Op Code = 1

**PDP Address Request Message Format**

<table>
<thead>
<tr>
<th>Version</th>
<th>Flags</th>
<th>Op Code = 1</th>
<th>Client-type = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Message Length = 12</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TTL = 8</td>
<td></td>
</tr>
</tbody>
</table>

**PDP Location Message:** Client Type = 3, Op Code = 2

**PDP Location Message Format**

<table>
<thead>
<tr>
<th>Version</th>
<th>Flags</th>
<th>Op Code = 2</th>
<th>Client-type = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Message Length = 16</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TTL = 8</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P-ABB Address</td>
<td></td>
</tr>
</tbody>
</table>

**PDP Keep Alive Message:** Client Type = 3, Op code = 3

**PDP keep alive Format**

<table>
<thead>
<tr>
<th>Version</th>
<th>Flags</th>
<th>Op Code = 3</th>
<th>Client-type = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Message Length = 12</td>
<td></td>
</tr>
</tbody>
</table>
PDP Relocation Message: Client Type = 3, Op Code = 4

PDP Relocation Message Format

<table>
<thead>
<tr>
<th>Version</th>
<th>Flags</th>
<th>Op Code = 4</th>
<th>Client-type = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Message Length = 12</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Relocated ABB Address</td>
<td></td>
</tr>
</tbody>
</table>

PDP Backup Message: Client Type = 3, Op Code = 0

PDP Backup Message format

<table>
<thead>
<tr>
<th>Version</th>
<th>Flags</th>
<th>Op Code = 0</th>
<th>Client-type = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Message Length</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Length</td>
<td>C-Num = 9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Backup data</td>
<td></td>
</tr>
</tbody>
</table>

PDP Monitor Message: Client Type = 3, Op Code = 5

PDP Monitor Message Format

<table>
<thead>
<tr>
<th>Version</th>
<th>Flags</th>
<th>Op Code = 5</th>
<th>Client-type = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Message Length</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Length</td>
<td>C-Num = 9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Network information</td>
<td></td>
</tr>
</tbody>
</table>