Scalable Design of a Policy-Based Management System and its Performance

K. L. Eddie Law, University of Toronto
Achint Saxena, Walker Wireless Limited

ABSTRACT

The recommended policy-based management system by the IETF is a two-tiered architectural design. It exhibits several observable fundamental limitations, for example, system scalability and cross-vendor hardware compatibility problems. In this article, a multitiered architecture, the unified policy-based management system, is proposed. The middle-tier agents, introduced between policy managers and network routers, offer flexibility and scalability to the design. A dynamic unified information model can be achieved between the policy decision and enforcement points of the system by properly extending network protocols, by installing or removing hardware interpretation modules on the fly, and by interpreting and translating request/decision messages at the agents. To complete the UPM design, novel load sharing and balancing communication protocols are implemented to improve system scalability. The system performances of the UPM are compared to the recommended PBM system of the IETF through extensive experiments. The results indicate that the UPM is a high-performance and scalable policy-based management design.

INTRODUCTION

Traditionally, the Internet offers best effort traffic model that does not work effectively for interactive applications. Different quality of service (QoS) classes have been introduced to work for different users' and applications' requirements. These service classes are usually associated with certain performance-related parameters such as the required bandwidth and delay per flow. The ultimate goal is to have different applications, for example, voice over Internet Protocol (VoIP) and transmissions of critical data, deliver harmoniously on the Internet. A policy-based management system can offer an operating platform and a solution to achieve this goal. The Internet Engineering Task Force (IETF) recommended a policy-based management (PBM) system [1] to enforce different QoS guarantees to end users and applications. In the following subsections, the design of the recommended PBM system and its associated operating protocols are explained.

REVIEWS OF IETF'S POLICY-BASED MANAGEMENT

With the PBM system in a policy domain (PD), connections are maintained and admitted to different service classes based on assigned policy rules. Because of these rules, network routers can determine to enforce admission control, traffic shaping, and scheduling mechanisms for different users under different traffic conditions. Usually, the rule parameters may include desired bandwidth, delay, duration, jitter, starting and finishing times, and some other limitations.

Without considering the directory server, the recommended framework [1] is a two-tiered architecture as shown in Fig. 1a. It consists of a policy manager (PM) or policy server (PS),[1] and multiple network routers at the edge or boundary of a domain. The architecture works with the Common Open Policy Service (COPS) protocol [2] and Lightweight Directory Access Protocol (LDAP) [3]. The LDAP server stores all legitimate policy rules, and COPS defines two operating entities in the system: the policy decision point (PDP) and policy enforcement point (PEP). In the framework, the PDP operates in a PS, and a PEP is usually located at an edge router (ER) or boundary router (BR). If there are different network devices from different vendors, there should have multiple PDPs in a domain and each of them is responsible for controlling a set of PEP devices.

The COPS is a lightweight but flexible client/server protocol. It uses Transmission Control Protocol (TCP) and defines communication messages among PEPs and PDPs. Moreover, COPS can be adjusted to function with other network protocols, for example, the COPS-Resource Reservation Protocol (RSVP) [4]. There are two fundamental operating modes for COPS: outsourcing and provisioning. For the standard outsourcing design, a PEP receives a

[1] *Policy manager (PM) and policy server (PS) are used interchangeably in this document.*

0163-6804/03/$17.00 © 2003 IEEE

connection service request and subsequently forwards it to a preassigned PDP. Upon receiving the request, the PDP fetches related policy rules from an LDAP server to make the final decision (i.e., to accept or reject the request). This decision is sent back to the PEP for policy enforcement in the domain. In the policy provisioning model, policy server checks if any policy rules have been updated or altered. If it happens, it enforces the latest decisions at network routers accordingly without any requests from end users. The COPS-PR extension [5] is an example of policy provisioning for differentiated services (DiffServ) networks [6].

Pros and Cons of PBM

In PBM, the two-tiered architecture operates with the standard COPS protocol and message formats. The design is simple and moderately extensible. Moreover, the COPS can be adapted to other protocols easily (e.g., RSVP). Unfortunately, the PBM has several limitations. The most conspicuous limitation is the scalability issue in heterogeneous networks in terms of router manufacturers and models. One PDP can only control a limited number of PEP devices within a domain. Moreover, each router may have different implementations of COPS, ranging from something as minor as different versions to something as crucial as different message formats. Even if certain message contents are standardized, different vendors can still develop proprietary message structures and policy rules. Therefore, intervendor COPS compatibility is always a continuing problem.

Another design problem is that the PBM does not support legacy routers. This becomes a major hurdle to overcome toward QoS deployment on the Internet. To offset startup costs, it is desirable to have the PBM integrate well with legacy equipment. Moreover, it cannot solve the reliability issue by simply connecting a PEP to multiple PDPs. This is because each enforced decision at a PEP has a unique client type [2] at a PDP that is not directly transferable among PDPs. Therefore, if a PDP fails, it may cause enforcement discontinuities at its associated PEPs. It should have a backup PDP design in PBM. Unfortunately, a PEP is not automatically aware of the location of a backup system, and the list of backup PDPs should be manually configured into the PEP at registration. Also, the list is not updated dynamically if there are changes of settings in the domain.

As a result, the recommended two-tiered architecture is not scalable to operate in large networks. It also does not adapt itself to controlling different routers from different vendors easily. There are no load sharing and balancing mechanisms at PDPs, which may lead to uneven distributions of congested and idle PDPs in large networks. Since a PEP always connects to one PDP, it can only wait for a timeout to expire between each connection request if the PDP is too busy to reply. All these problems must be addressed before QoS and PBM become widely adopted for use on the Internet. An improved policy-based management system should then be investigated.

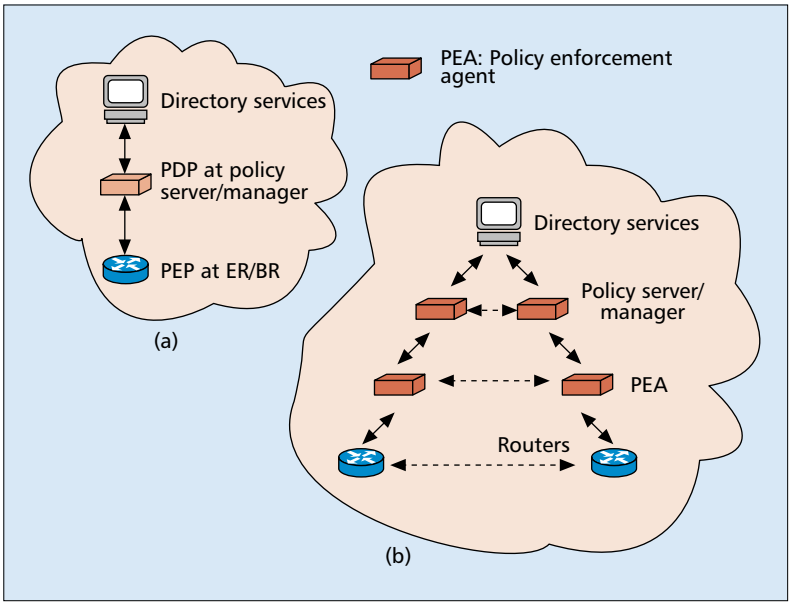

Figure 1. *Within a policy domain: a) IETF's recommended model; b) the proposed UPM model.*

System Architecture of Unified Policy-Based Management

In this article a novel policy-based management system, unified policy-based management (UPM), is proposed. Apart from the directory server, the architecture has three tiers, as shown in Fig. 1b. Fundamentally, the top and bottom tiers operate similarly to the ones in the standard PBM framework. The policy managers at the top tier contain PDP entities for making final policy decisions. Network routers, ERs and BRs, are at the bottom tier for policy enforcement. Currently, the core routers (CRs) are not managed in PBM but can easily be controlled in the UPM framework in the future. A new middle-tier entity is introduced in UPM to coordinate the communications, and establish transparency between routers and policy servers. The entity is known as a policy enforcement agent (PEA) and has several design goals. A PEA may enforce received policy decisions from PSs to provide guaranteed QoS services at ERs for end hosts within its domain or at BRs for connections from other domains. PEAs should:

- Relay protocol messages between PSs and ERs/BRs
- Translate different policy rules
- Distribute different policy sessions to PSs
- Enforce guaranteed QoS traffic at ERs/BRs
- Inform ERs/BRs about other PEAs if required

As an example, if a router cannot interpret a policy decision, the PEA should operate as a proxy to translate and enforce the decision by executing line commands at the router. The UPM works with the outsourcing and one-way provisioning models. A PS can receive COPS requests from either the routers or the system administrators with domain-level decisions. With fetched policy rules from the repository, the PS finalizes a to-be-applied policy rule and delivers it to a PEA at the middle tier for enforcement.

The proposed UPM design is mainly an architecture enhancement of PBM. It avoids altering any approved and recommended network protocols, e.g., the COPS protocol, thereby expediting the time to deploy UPM without going though the time consuming process in extending system protocols.

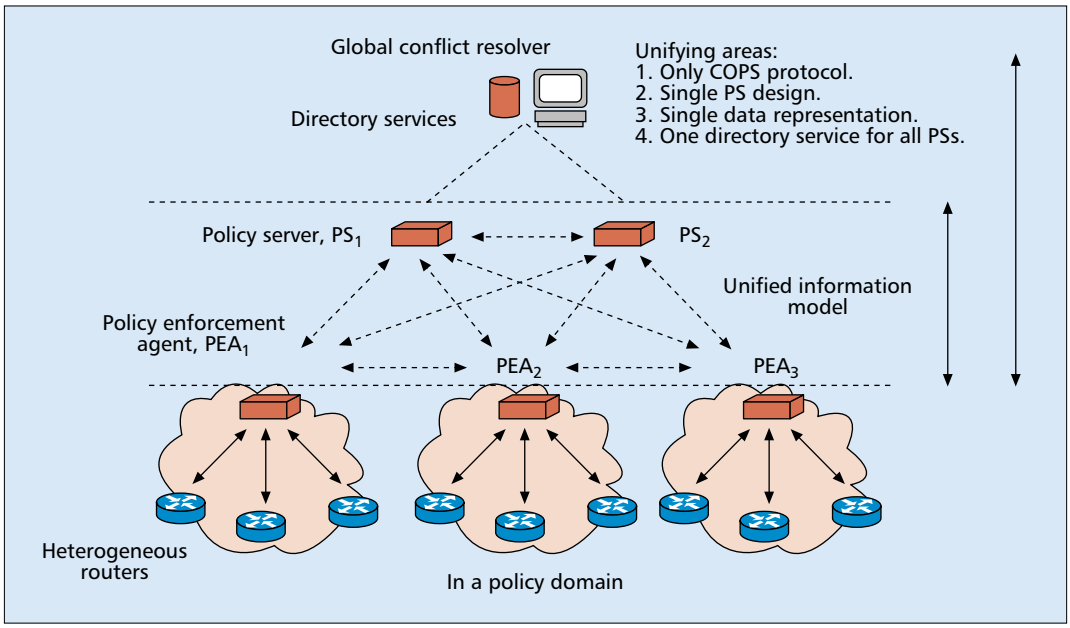

Figure 2. *The implementation model of the UPM system.*

Another enhancement is to offer load balancing and sharing services at the middle and upper tiers in the architecture. If there are many routers to control, there may be multiple PEAs to coordinate in a domain. UPM provides designs for inter-PS, inter-PEA, and PS-PEA communications. Readers may consider that the work at a PEA may introduce unnecessary delays and workloads if, indeed, both a router and the corresponding PS can interpret the contents of messages fully. Consequently, a bypass enhancement at the transport layer of a PEA is introduced to expedite message delivery in which packets are passed between routers and PSs without any intervention from the middle-tier agents. Indeed, all routers are required to communicate via PEAs to PSs for security and load monitoring reasons. This permits a PEA to measure traffic loads on its associated routers and PSs regardless of whether the bypass mechanisms are carried out or not. Subsequently, load sharing can be carried out with PEAs for future policy requests.

Besides, a unified information model (UIM) [7] is constructed between PSs and PEAs. UPM can operate with different equipment from multiple vendors dynamically. Readers can consult [7] for more information on implementing UIM. Basically, it requires the PSs and PEAs to communicate with the same COPS version. Subsequently, we can add more PSs and PEAs as required without disrupting any enforced services.

The proposed UPM design is mainly an architecture enhancement of PBM. It avoids altering any approved and recommended network protocols (e.g., COPS), thereby expediting the time to deploy UPM without going through the time-consuming extension of system protocols.

System Components

UPM reuses several components that exist in the recommended PBM framework. While some of these components need enhanced capabilities, others may require less complicated features. A complete design framework for UPM is shown in Fig. 2. All PSs and PEAs should operate with the same COPS version, but the network routers at the bottom tier are not required to understand COPS; they can run on different versions of COPS or with different message contents. The policy rule database [8] is comparatively static because recently added policy rules are rarely modified. Hence, only one centralized directory server is used in the current design.

Routers and Policy Enforcement Points

Different from the two-tiered model, UPM allows non-COPS-compliant PEPs to exist at the bottom tier. This is because these routers always connect via an PEA with proper message conversions. Three different operating modes for network routers can be considered.

Native COPS PEP: A router and PS use the same COPS version and message contents. This PEP uses COPS to connect to a relevant PS through a PEA via the bypass function with traffic monitoring operations only.

Non-native COPS PEP: A router uses a COPS protocol, but with a different version or message format. It requires proper message conversions at a PEA. This is a common scenario when there are routers from different vendors.

Non-COPS router: As in many legacy routers, a router has no COPS for policy requests. A PEA is then set up to offer certain QoS services, although it cannot pull any policies at the router. These routers are usually configured, via telnet, with a command line interface (CLI) and monitored via Simple Network Management Protocol (SNMP) [9].

The Policy Enforcement Agent

The PEA is a multifunctional unit that enables system flexibility and scalability. To non-COPS routers, a PEA acts as a COPS proxy. Indeed, the PEA's primary goal is to provide seamless

integration of routers and PSs in a domain. The presence of PEAs should not be noticeable by other network components. It works like a PDP when it communicates with routers. To the policy servers, it operates as a PEP. Since the PEA is an ideal point to monitor COPS traffic distributions, it distributes a service request to a noncongested PDP upon receiving a new request. All policy-related connections are required to pass through PEAs for load measurements. In the future, various network monitoring protocols such as SNMP can be integrated into the PEA. Several unique features will be further discussed later.

Policy Servers and Directory Server

The PSs are responsible for making the decisions PEP devices enforce. To the PSs in UPM, PEAs operate like PEPs; hence, the design of PSs can be simplified as a PEA can translate any non-PS-compliant messages. With the simplified design, PSs can be added on the fly to the system at ease. It enables easy establishment of multiple PSs and PEAs in a domain when required.

The role of a directory server is identical to that in PBM framework, and it only allows registered PSs to access the information saved in its policy rules repository [1].

Novel Designs In UPM

System Improvements with PEA

COPS and Content Translation — One important function of the PEA is to translate unintelligible COPS messages from network routers into ones understood at a PS. Although network routers are previously partitioned into three classes, they are reclassified into two classes for proper implementations at PEAs.

Non-COPS-compliant routers: In this group, there can be "push" only routers that are usually configured manually. User-based input or a client program is created at the PEA to imitate the pull mechanism for the router to communicate with a PS. Enforcements are delivered to a router through its acceptable communication means (e.g., the CLI on telnet). On the other hand, there are pull-capable routers equipped with other, possibly proprietary, policy protocols. A basic store, convert, and forward mechanism can be used at the PEA. The forwarding mechanism consists of opening a new connection or using an existing connection to a PS to send and receive request and decision messages.

COPS-compliant routers/PEPs: Different vendors may have different implementations of COPS. Even if they are compliant with the standard, the request and decision message structures may be different. In these cases, translations at the PEA must be required. There are two levels to do translations:

- Message format translation is required for nonstandard COPS implementation.
- Content conversion and selective forwarding may be required for individual COPS messages.

The setup of a translation module repository will be discussed next.

Translation Module Repository — The UPM can work with network devices with different protocols and command interfaces because of the translation module repository design. Whenever a PEA finds new network routers under its enforcement control, it reaches the repository to check for control modules for the new devices. Therefore, different translation and control modules can be loaded dynamically in PEAs when required to provide architectural flexibility.

Since there may be more than one PEA, it is convenient to keep the modules in a separate repository. The repository can be locally or remotely located. Licensing issues may necessitate keeping the modules at a fixed location. In this case, only certain PEAs may have translation modules; we call these local, static, or nonshareable modules. If there are nonshareable modules, inter-PEA communications can be used to redirect network routers to achieve policy control. In the testbed implementation, the Remote Method Invocation (RMI) in the Java programming language is used to dynamically load modules from a repository in prototype. Common Object Request Broker Architecture (CORBA) is another alternative, but its operations are comparatively slower.

In the testbed implementation, the RMI in the Java programming language is used to dynamically load modules from a repository in the prototype. CORBA is another alternative, but its operations are comparatively slower.

TCP-Bypass Mechanism — If router and PS communicate using identical COPS message structures and information contents, no intermediary translations are needed. The TCP-bypass mechanism simply forwards any incoming requests or following packets at the transport layer to a PS known to the PEA. It removes the unnecessary translation overhead and latency at the PEA.

Based on the first received COPS data packet, the PEA may determine to forward an incoming connection without processing. From the information in the COPS header, the version and implementation signature may be deduced. If this matches the COPS formats and content types at a PS, a new TCP connection is opened to this selected PS, and the finite state information regarding this TCP connection is cached. Connection switching without processing at the PEA can be done at either the TCP or COPS level, or at both. It definitely offers faster switching time at the transport layer. This creates a "total transparent connection" between a PEP at a router and a PDP at a PS for they can then maintain their own persistent connections. To handle the bypassed COPS messages, careful designs are required to mirror and switch:

- COPS connection establishment messages to PSs
- Corresponding replies from PSs
- Unsolicited PS messages

The Client Handle [2] and other information in a COPS header can be used to uniquely identify COPS requests.

Load Balancing and Sharing Solutions — An PEA may only know one PS when it is activated. It learns dynamically of the presence of all PSs through the inter-PS communication pro-

It may be simpler to extend COPS extensible messages for passing system loading and location information. The PEA should serve as a secure gateway to PSs; therefore, message hashing may be required to improve the system security.

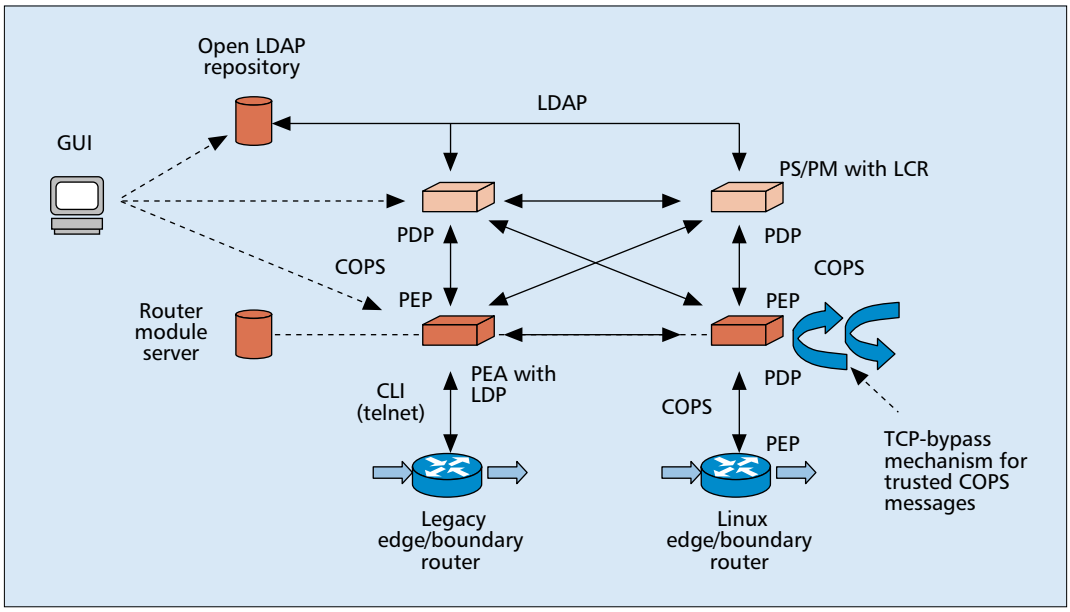

Figure 3. *The UPM testbed prototype.*

tocols. Similarly, each PEA monitors PS-related performance variables such as CPU utilization, memory usage, link utilization, ping times, and link capacity/congestion through the PEA-PS communication protocols. Different weighted values can be assigned to different parameters and different PSs for load sharing and balancing operations. Subsequently, new requests to each PEA can be served in a weighted round-robin (WRR) fashion.

Another important feature is to forward connections based on the client-type [2] field in COPS. This can coexist fairly well with the aforementioned load balancing scheme. When a PS registers with a PEA, it notifies the PEA about the special client-types it supports. Hence, load balancing on new connections with specific incoming client-types is restricted to certain PSs in the system.

Multiple PEAs and Inter-PEA Communications — When a PEA is close to being overloaded (e.g., there are too many opened connections with high CPU utilizations or its network links are almost congested), the PEA may decide to issue a COPS redirect message to routers sending new requests. This is a simple load sharing mechanism among PEAs in a domain. Additionally, it may be beneficial for a router to connect to a nearby PEA instead of one that is far away. Indeed, there may be PEAs that are module-specific, vendor-specific, or area-specific if there is a high concentration of a certain kind/brand of routers in one small area.

It is necessary to share information among PEAs to offer a scalable solution. Inter-PEA communication protocol is designed to share information, such as supported vendor modules, list of PSs with associated states, and network monitoring and utilization information (particularly the availability of any local non-shareable modules). There are many methods that can be used to implement this protocol. One method is to deploy multicast services in networks. All PEAs should listen to a specially assigned multicast address. A PEA should join the multicast tree when it is started and multicast its own information at regular intervals. It may be simpler to extend COPS extensible messages for passing system loading and location information. The PEA should serve as a secure gateway to PSs; therefore, message hashing [10] may be required to improve system security.

Design Issues on Policy Servers

Similar to the communication mechanism among PEAs, inter-PS communications should be performed using another specially assigned multicast address. It is beneficial to all PSs to exchange information on servicing client-types and others. The inter-PEA communications indeed negate the need of bulk data sharing among PSs. However, the inter-PS communications become useful when a PS is purposely brought down for maintenance. COPS offers stateless communications by carrying a Client Handle field in messages for PS to carry out appropriate operations according to locally stored information. Therefore, inter-PS communications should enable proper partitioning of the number sets used among PSs in setting the Client Handle values. Subsequently, the stateless information could be transferred to another PS without introducing enforcement confusion during system maintenance.

Related Design Issues on Directory Services

There has been much work related to directory services in a policy-based management system. Some research work is outside the scope of this article and should fall into database management research:

- The high-level definitions of contractual agreements, the service level agreement

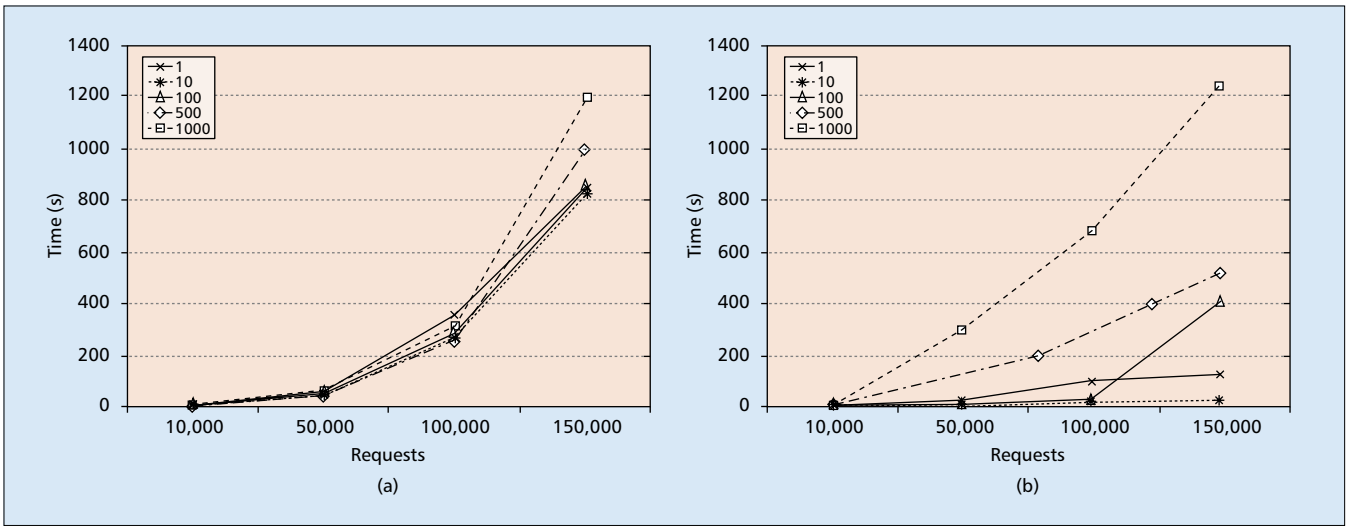

Figure 4. *Nonbursty traffic with 1, 10, 100, 500, and 1000 TCP connections opened: a) no DRQ sent; b) DRQ sent.*

(SLA), among participants should be mapped onto low-level technical parameters in the policy rule formats.

- Whenever a new policy rule is added to the repository, conflict detection and resolution (CD&R) mechanisms inform both the user and administrator if it is accepted. If conflicts with stored policy rules arise, possible resolutions should be suggested, if possible.
- Fast designs on policy rule searching, filtering, and matching with real-time performance are desirable.

The system architecture should be enhanced to include replicated directory services [11] to provide reliability and improve performance, since the LDAPv3 server [3] does not notify PSs if the database has been modified. To avoid this problem, all new policy negotiations must take place via PSs to activate the CD&R for potential executions of new rules. The LDAP repository is then updated, and a PEA will be notified immediately for proper rule enforcement.

Performance of UPM

Setup of Current Prototype

A UPM prototype[2] with some basic functionality has been developed (Fig. 3). All designs are implemented in Java and C programming languages. Extensive experiments have been carried out with one or more PSs with a load sharing mechanism. The TCP-bypass at the PEA is implemented in kernel for concept verification. Even though the implementations of most system components have not yet been optimized for heavy use, the results obtained are sufficient to demonstrate the superiority and feasibility of the UPM design. Performance of standard PBM design from IETF will be compared to the proposed UPM design.

System Performance of Two-Tiered PBM

The recommended framework from the IETF (Fig. 1a) is tested for comparison. The system is stressed under a large number of service requests. In the first series of tests, a PEP client only sends request when the decision on its previous request has returned. This series of tests is called nonbursty request stress in graphs. Moreover, if a graph is marked with deletes, it means that a COPS Delete (DRQ) message is sent whenever each decision has just been received. Typically, it reduces the amount of saved state information and improves the performance of a PS.

In Fig. 4a the time required to process the messages grows exponentially with respect to the number of TCP connections as well as the number of service requests. With only one PS, the results demonstrate that performance deteriorates rapidly with increasing number of routers. In these cases, undeleted requests imply more memory used to store the states of different connections. Rehashing hash table for an increased number of connections may be required, and response times slow down. This explains the almost exponential increase in the total time taken to process the requests.

For another set of experiments, request state is deleted with a DRQ message whenever a decision has been made. The server keeps less information than the last test set. We intend to test the computation performance instead of other limiting factors such as memory size. As shown in Fig. 4b, the system performs better with smaller numbers of TCP connections. When there are more TCP connections (e.g., 1000) no improvements are observable compared to Fig. 4a.

Another series of tests is to send a burst of requests simultaneously to the PS without waiting for any COPS decisions. It is a bursty stress test. It has similar measured results to the nonbursty cases. Conclusively, the performance of the standard PBM system deteriorates rapidly with increasing number of service requests and TCP connections. This further demonstrates that the standard PBM system suffers from both scalability and reliability issues.

System Performance of UPM

TCP-Bypass Overhead — This bypass design reduces the loads of the COPS entities within the PEA and allows the PEA to handle more network routers subsequently. In this section the TCP-bypass overhead at a PEA is examined.

[2] *The servers and clients run on 1GHz and 500 MHz Pentium III computers with Linux kernel v. 2.4.5., respectively.*

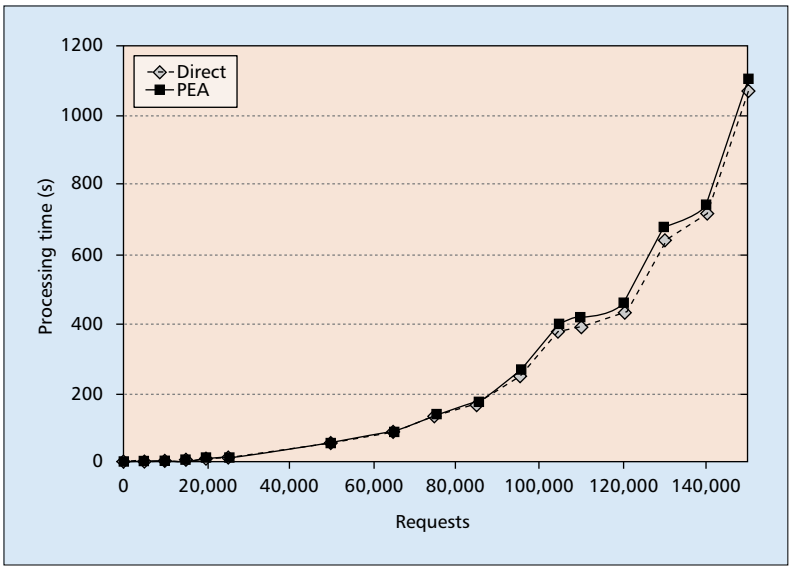

Figure 5. *UPM with PEA.*

The total time spent from PEP to PS through a PEA in UPM will be compared to that in PBM. The measured processing times spent in both configurations are shown in Fig. 5. The overhead is almost unobservable for a small number of requests. Upon increasing the number of requests, say, > 12,000, the measured overhead of current implementation is less than 5 percent. In future implementations, a faster indexing structure such as on a hash table in kernel should be used.

Some observed overhead is due to added routing latency via the PEA. In the experimental setup, influences such as delay to and from a PEA are negligible, since all machines are connected via a dedicated hardware switch. In some network situations, if the connection via a PEA is congested, the overhead will be much larger. Redirection of new COPS connections at a PEA to another uncongested PEA is therefore useful, and this is one of the designed functions provided by the PEA with resource monitoring.

Transient State Load Sharing Performance — The load sharing performance of UPM architecture under network congestion condition should be examined. As shown in Fig. 6a, there is only one PS in PBM; under bursty request loads, the PS is not able to handle requests at time ≈ 15 s. The server recovers to work at time ≈ 100 s. The outage duration can be as long as 85 s.

On the other hand, consider a UPM system that has two PSs with TCP-bypass disabled at the PEA; one PS is intentionally congested with requests. Its loading information is monitored regularly at the PEA through PS-PEA communications. The link utilization to the PS is plotted in Fig. 6b. The response time of the PEA depends on monitoring update time (periodic or event driven) from the PS. For the experiment with load sharing capability, a PS cannot handle more requests around time ≈ 24 s. This is because the PEA distributes the service requests among the two PSs in a round-robin fashion. The maximum bandwidth usage per link is about half of the link in Fig. 6a. Moreover, the outage duration of the PS lasts for about 10 s. This test clearly demonstrates the benefits of the PEA and how it addresses scalability issues.

Multiple-PS and Multiple-PEA System Performances — In this section the scalability of the proposed system is studied. As shown in Fig. 7a, the breakdown of a PS and decision throughput are combined into a single metric. There are 50 TCP connections, and the request count per TCP connection is increased to stress the system.

Defining the average decision returning rate as an evaluation metric, Decisions Received × Average Decision Throughput, it is important to have a high return rate. The measurements are plotted in Fig. 7a with one PEA. There is a short linear period when the decision throughput is close to constant. However, when the request count per connection increases, the PSs get more heavily loaded, and the decision throughput lowers. The benefit of having multiple PSs is clear in this case. Not only does the

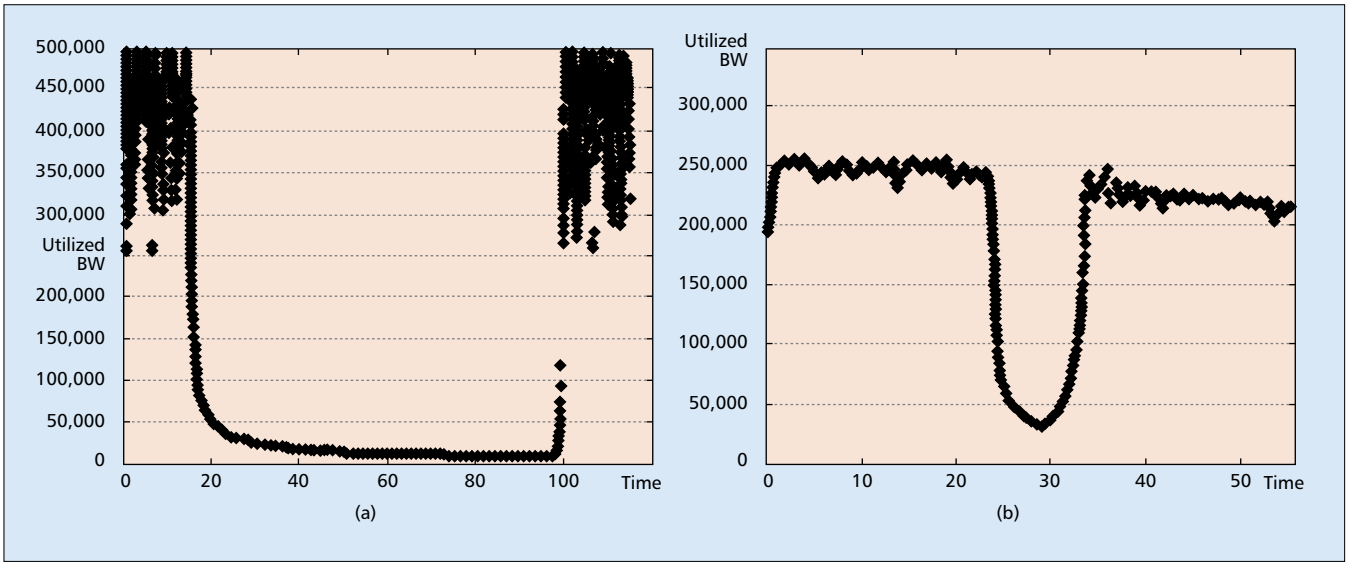

Figure 6. *Bandwidth usage (bytes per second) vs. time (seconds): a) 1 PS in IETF's PBM; b) 2 PSs, 1 PEA in UPM.*

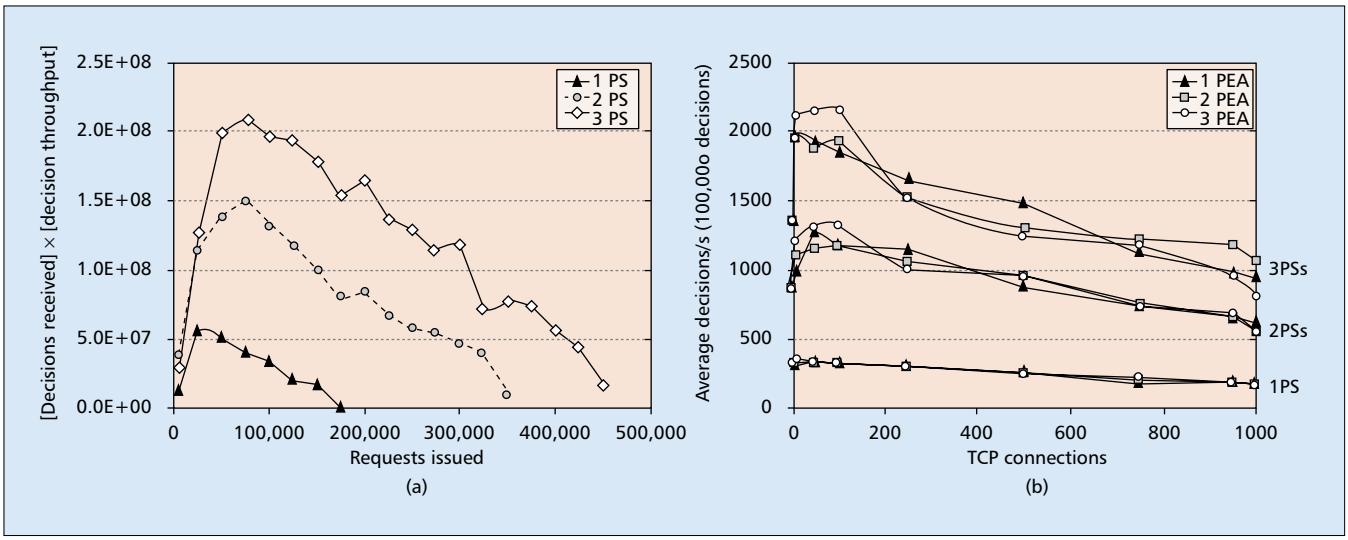

Figure 7. *Average throughput on COPS decisions: a) 1 PEA, multiple-PS; b) multiple-PEA, multiple-PS.*

throughput increase, even during heavy loads, but the maximum request breakdown limit is also prolonged.

Next, it is important to observe the effect of PEAs in the design. As shown in Fig. 7b, given a fixed number of policy servers, the performance of the system is about the same with one, two or three PEAs. This again demonstrates the transparency of the PEA in UPM. As indicated in the last experiment, the throughput improves steadily whenever one more policy server is added to the system. However, the standard PBM system does not have any scaling solution. It is also observable that the decision returning rate grows slower with more PSs in the system because of the requirements to carry out overhead among all PSs and PEAs.

Concluding Remarks

In this article we investigate the performance limitations of the recommended PBM system from the IETF. A new multitiered policy-based management framework is proposed. With extensive experiments, the proposed UPM design is demonstrated to offer high decision throughput performance. Numerous unique features have been introduced in the PEA, for example, the expedited TCP-bypass mechanism with unnoticeable latency. More important, the PEA provides load sharing and balancing mechanisms that make UPM a highly scalable design. The implemented prototype has confirmed these expected improvements. We believe that UPM provides some solutions to the problems facing policy-based QoS today.

References

[1] R. Yavatkar, D. Pendarakis, and R. Guerin, "A Framework for Policy Based Admission Control," RFC 2753, Jan. 2000.
[2] J. Boyle *et al.*, "The COPS (Common Open Policy Service) Protocol," RFC 2748, Jan. 2000.
[3] M. Wahl, T. Howes, and S. Kille, "Lightweight Directory Access Protocol (v3)," RFC 2251, Dec. 1997.
[4] R. Braden *et al.*, "Resource ReSerVation Protocol (RSVP) — Functional Specification," RFC 2205, Sept. 1997.
[5] K. Chan *et al.*, "COPS Usage for Policy Provisioning," RFC 3084, Mar. 2001.
[6] S. Blake *et al.*, "An Architecture for Differentiated Service," RFC 2475, June 1994.
[7] K. L. E. Law, "XML on LDAP Network Database," *Proc. IEEE Canadian Conf. Elec. and Comp. Eng.*, Halifax, Canada, 2000.
[8] R. Sahita *et al.*, "Framework Policy Information Base," RFC 3318, Mar. 2003.
[9] J. Case *et al.*, "Simple Network Management Protocol," RFC 1157, May 1990.
[10] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," RFC 2104, Feb. 1997.
[11] J. Merrells, U. Srinivasan, and E. Reed, "LDAP Replication Architecture," work in progress, Mar. 2003.

Biographies

EDDIE LAW (eddie.law@utoronto.ca) received a B.Sc. degree in electrical and electronic engineering from the University of Hong Kong, an M.S. degree in electrical engineering from Polytechnic University, Brooklyn, New York, and a Ph.D. degree in electrical and computer engineering from the University of Toronto, Canada. He worked in three different groups, Passport Research, Next Generation ATM Systems, and Computing Technology Lab, in Nortel Networks, Ottawa after obtaining his doctoral degree. Since September 1999 he has been an assistant professor in the Edward S. Rogers Sr. Department of Electrical and Computer Engineering at the University of Toronto. His current research interests are in active networks, policy-based management on the Internet, TCP/IP protocol designs, ad hoc MAC designs, reconfigurable network designs, and photonic switch fabric designs.

ACHINT SAXENA (achint.saxena@utoronto.ca) received his B.E. (Hons) degree in electrical and electronic engineering from the University of Canterbury, New Zealand, and his M.A.Sc. degree in electrical and computer engineering from the University of Toronto, Canada, in 1998 and 2002, respectively. He is currently working as a software development engineer on a 3G network deployment for Walker Wireless Ltd., New Zealand. His research interests include broadband networks, and network, systems, and traffic management.