Quality of Service (QoS) Policy Framework

Kaustubh S. Phanse

Bradley Department of Electrical and Computer Engineering, Virginia Tech.

ABSTRACT

For successful deployment of robust quality of service (QoS) framework, the need for a QoS policy system looks inevitable. In this report we explore the various elements, which together form a QoS policy framework and also try to gain insight into the implementation issues of such a framework and provide directives for future research.

1. Introduction

In recent years, the Internet has evolved from its legacy best-effort character to support differentiated service to different applications and customers. This is a result of considerable increase in deployment of IP based network services such as video-conferencing, Internet telephony, audio/video streaming, virtual private networking etc. which have specific performance requirements such as delay or jitter bound, bandwidth reservation, guaranteed delivery of business critical data and so on.

Internet traffic differentiation in turn does create some incentive for unauthorized usage or stealing of available resources. Certain malicious users may want the better service for their traffic without paying the (likely) higher price for the same. Such a *free-for-all* QoS implementation may lead to chaos and possibly result in even worse than best-effort performance. This calls for a set of rules to dictate QoS, 'police' to enforce these rules and 'judges' to decide when they apply. All these elements together form what is known as QoS policy framework, an essential component of a QoS-enabled network. QoS policy can be also looked upon as a subset of a wider area of recent interest, namely policy based network management (PBNM) [1]. Unlike legacy network management, which generally involves configuring and managing each network entity individually, PBNM involves configuring and controlling the various operational characteristics of a network as a whole, providing the network operator with a much simplified and automated control over all the network.

A QoS policy system will typically involve the following. It implements, what we call the three As: *authentication* of the traffic owners (users, end-hosts, applications etc.), checking whether the traffic owners requesting certain level of service are actually *authorized* to do so, and lastly, in most cases, maintain proper *accounting* or billing based on different factors such as the service levels, user-domains, network usage etc. Further, it enables policy-based admission control (PAC) based on various factors such as type of user or application, preemption priority, time of day or week etc. in addition to the availability of resources (bandwidth), the metric used in most legacy call admission control (CAC) mechanisms. This in turn can be used to provide an adaptive and robust characteristic to the entire QoS framework as a whole. By adaptive and robust, we mean having considerable flexibility in selection and enforcement of policies, which will in turn allow *graceful degradation* of service and delivery of at least the business or mission critical data under worst circumstances.

The aim of this report is to provide a brief survey of the work done so far in the field of QoS policy (in terms of standardization, research and implementation) and to describe the elements typically found in a QoS policy framework. The report also discusses the different approaches involved in implementing such a framework and the various research issues pertaining the same.

2. Related Work

The growing interest in the field of policy-based networking (PBN) is being reflected in the form of new working groups, conferences and commercial products supporting PBN. The IETF resource allocation protocol (RAP) [2] working group is active in the field of QoS policy. It has defined, among other standards, the PAC framework [3] and the common open policy service (COPS) protocol [4], a simple client-server protocol, to facilitate communication between the policy server(s) and clients. The focus of most related papers [5,6,7] published so far has been to motivate the need for QoS policy, discuss the desirable features of a QoS policy framework and related issues. There is a need to further this work by an experimental or simulation implementation of a QoS policy framework providing insight into the various performance issues (using metrics such as policy signaling overhead, call or connection set-up delay, jitter, throughput etc.).

3. QoS Policy Framework

3.1 Architectural Elements

Figure 1 shows a typical QoS policy framework. The four policy-driven architectural elements are as follows. A network administrator or operator uses the *policy management tool* to define the various policies or policy groups, which are then stored in a *repository*. Such repository, authentication server etc. may be co-located with a *policy decision point (PDP)*. The PDP retrieves the policies, performs complex policy translation and interpretation, which are then used to configure the *policy enforcement points (PEPs)*.

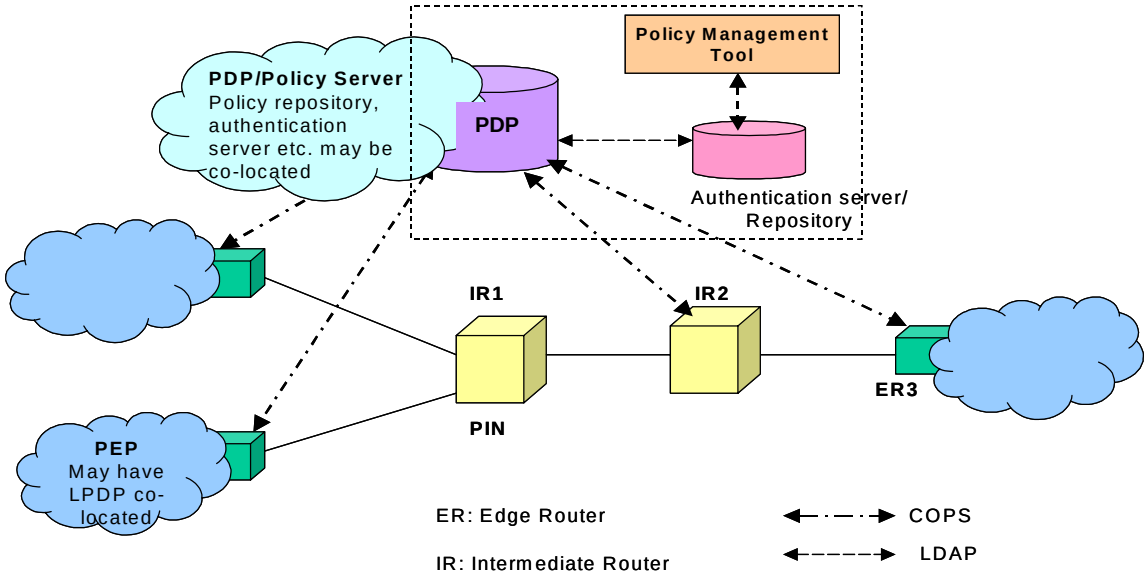

Figure 1. Typical QoS policy framework

3.2 QoS Policy Models

Depending on the type of QoS architecture being used, one of the two policy models [4], namely the *outsourcing* and *provisioning* models can be adopted. The outsourcing policy control model is fitting for a dynamic or signaling-based QoS architecture such as that using RSVP, while the provisioning model (also known as the configuration model) is apt for static QoS architectures such as Diffserv. We briefly describe each of these models below.

Outsourcing: In the outsourcing model, when the PEP receives an *event* (e.g. in case of RSVP, a new or modified PATH or RESV request message) that requires a new policy decision it sends a request (REQ) message to the remote PDP. The PDP then makes a decision and sends a decision (DEC) message (e.g. *accept* or *reject*) back to the PEP. Since the request is stateful, the request is installed at the remote PDP. The unique handle (unique per TCP connection and client-type), specified in both the REQ and its corresponding DEC messages identifies this request state. The PEP is responsible for deleting this request state once the request is no longer applicable. It can also update a previously installed request state by reissuing a new REQ message for the previously installed handle. The remote PDP is then expected to send an updated DEC message back to the PEP. Likewise, the server may change a previously issued decision at any time by issuing an unsolicited or asynchronous DEC message. At all times the PEP module is expected to abide by the PDP's decisions and notify the PDP of any state changes.

Provisioning: Even in the provisioning model, similar to the outsourcing scenario, the PEPs make a configuration request to the PDP for a particular interface, module or functionality. In response the PDP sends the appropriate provisioning data to each of the PEPs (e.g. in Diffserv based network, the PDP may configure access routers using tables mapping users/hosts/applications to DSCP markings, and similarly it may configure the core routers with tables mapping DSCPs to PHBs). Every PEP is expected to install and use the configuration locally. A particular named configuration can be updated or removed by simply sending additional DEC messages specifying to update or remove the named configuration. In all cases, the PEP may notify the remote PDP of the local status of an installed state using the report (RPT) message where appropriate. The RPT message is used to signify periodic updates for monitoring and accounting purposes depending on the client.

3.3 Implementation Approaches

In this section, we describe and compare the various approaches that can be used to implement a QoS policy system. Note that apart from some basic definitions (e.g. centralized vs. distributed) mentioned in [3], the details and analysis of the various approaches that can be adopted are implementation-specific and open to further research.

Centralized: A centralized policy system typically will have a single policy server or PDP and a centralized repository and authentication server (which may be co-located with the PDP). One of the major issues that needs to be handled when defining and enforcing policies is the *policy conflict check* and it is important for the network operator(s) to ensure the mutual co-existence of policies. It is desirable to have a centralized policy system from such policy management viewpoint, since it considerably simplifies the job of

consistency check of the various policies stored and maintained at a central location.

In an outsourcing type of model, since a PEP seeks policy decision for every event, a centralized approach will result in all the PEPs incapable of any policy decision-making and entirely dependent on the central PDP for the decisions. Although this may not entirely be the case in a provisioning scenario, since PEPs are expected to install the configuration information after it is received for the first time from the PDP. However, note that this cannot still be considered to be a distributive system, since the PEPs do not have co-located PDPs (which can aid the central PDP in the decision-making process) and still depend on the central PDP to timely receive updated provisioning information as required. Further, most of the information at each PEP is only of local significance.

Thus, a major disadvantage of the centralized approach is that the smooth functioning of the entire policy framework is largely dependent on the central PDP. In event of the PDP becoming non-functional (e.g. shutting down, security attack etc.), the entire policy system (and possibly the QoS system) may breakdown! This indirectly makes the need for security and integrity of the PDP and its stored information of prime importance. One way to prevent such a failure is redundancy, i.e. to have one or more back-up PDPs, which can handle the policy system if the main policy server breaks down. Another problem that such a centralized system may encounter is the amount overhead resulting from policy-related signaling. The lack of any decision making at the PEPs may result in additional signaling, which could be undesirable especially for networks with constrained bandwidth (e.g. wireless links).

Distributed: In addition to a primary or central PDP, a distributed policy architecture will typically have a local PDP (LPDP) co-located with every PEP. In such architecture, the PEP always first seeks a decision from its LPDP. The PEP then sends its request along with the LPDP's decision say D(L) to the primary PDP. The PDP makes a decision say D(P) based on the PEP's request message. It then performs a combination operation of D(P) and D(L) to reach a final decision, which is sent out to the PEP. One of the advantages of having an LPDP is the *short circuit processing* [3], i.e. if the result of D(L), above, is a *reject*, then there is no need to proceed with further policy processing at the PDP. This may considerably help in alleviating the *overhead problem* encountered in the centralized approach (e.g. especially under congested link conditions when most of the decisions of the central PDP would have been a *reject* due to lack of resources).

In all cases, however, the primary PDP must be informed of the failure of local policy processing as well as in a case when policy processing is successful but admission control (at the resource management level due to unavailable capacity) fails.

Hybrid or Adaptive: To further enhance the usefulness of a policy system, there is a need to evaluate mechanisms, which will help maintain the integrity of the policy system under certain demanding conditions and in turn provide the adaptive and robust character to the entire QoS framework. For example, in case of temporary or long-term failure of the primary PDP, one of the LPDPs could be elected to function as the central PDP, or another option could be a *zoning* approach, where in a group of LPDPs together form a *policy manager* and individually be in-charge of making decisions for specific zones or domains within the network. Another example could be of a heterogeneous network (e.g. involving wireless and wireline links), where in the wireline portion of the network without any bandwidth constraints and which possibly has a secured environment can adopt a centralized approach, while the wireless portion of the network with low bandwidth and security constraints may use a distributed approach.

4. Conclusions

In this report, we briefly motivated the importance of a QoS policy system. We discussed about the various architectural components found in a typical QoS policy framework, the two policy models and their applicability to different types of QoS architectures, and finally compared the pros and cons of adopting different approaches to define a QoS policy architecture. We found that there is a need for further research to gain insight in the field of QoS policy, especially based on quantitative analysis and results.

References

[1] D. Verma, *Policy-Based Networking: Architecture and Algorithms*, New Riders Publishing, 1st edition, November 2000.

[2] IETF resource allocation protocol (RAP) working group: http://www.ietf.org/html.charters/policy-charter.html

[3] R. Yavatkar, D. Pendarakis and R. Guerin, "A Framework for Policy-based Admission Control," RFC 2753.

[4] D. Durham et al., "The COPS (Common Open Policy Service) Protocol," RFC 2748.

[5] H. Huang, A. Meissner, W. Schoenfeld and R. Steinmetz, "QoS Policy Framework and its Implementation," in Proc. of IEEE International Conference on Communication Technology, vol. 1, pp. 860-867, August 2000.

[6] F. Bernbai, L. Gratta and R. Pietroiusti, "A policy based architecture for guaranteed QoS multimedia services," in Proc. of IEEE Conference on High Performance Switching and Routing, pp. 401-409, 2000.

[7] W. Changkun, "Policy-based Network Management," in Proc. of IEEE International Conference on Communication Technology, vol. 1, pp. 101-105, August 2000.