Internet Engineering Task Force
INTERNET-DRAFT

Raj Yavatkar, Intel
Dimitrios Pendarakis, IBM
Roch Guerin, IBM

November 21, 1997
Expires: May 20, 1998

A Framework for Policy-based Admission Control

Status of this Memo

This document is a product of the RSVP Admission Policy (RAP) working group in the Transport Area of the Internet Engineering Task Force. Comments are
solicited and should be addressed to the working group's mailing list at ipc@iphighway.com, and/or the author(s).

Abstract

TO BE FILLED IN.

1. Introduction

The IETF working groups such as Integrated Services (called ''int-serv'') and RSVP [1] have developed extensions to the IP architecture and the best-effort service model so that applications or end users can request specific quality (or levels) of service from an internetwork in addition to the current IP best-effort service. The int-serv model for these new services requires explicit signaling of the QoS (Quality of Service) requirements from the end points and provision of admission and traffic control at Integrated Services routers. The proposed standards for RSVP [RFC 2205] and Integrated Services [RFC 2211, RFC 2212] are examples of a new reservation setup protocol and new service definitions respectively. Under the int-serv model, certain data flows receive preferential treatment over other flows; the admission control component only takes into account the requester's resource reservation request and available capacity to determine whether or not to accept a QoS request. However, the int-serv mechanisms do not include an important aspect of admission control: network managers and service providers must be able to monitor, control, and enforce use of network resources and services based on policies derived from criteria such as the identity of users and applications, traffic/bandwidth requirements, security considerations, and time-of-day/week.

This document is concerned with specifying a framework for providing policy-based control over admission control decisions. In particular, it focuses on policy-based control over admission control using RSVP as an example of the QoS signaling mechanism. Even though the focus of the work is on RSVP-based admission control, the document outlines a framework that can provide policy-based admission control in other QoS contexts. For instance, it may be argued that policy-based control must be applicable to different kinds and qualities of services offered in the same network and our goal is to consider such extensions whenever possible.

We begin with a list of definitions in Section 2. Section 3 lists the requirements and goals of the mechanisms capable of controlling and enforcing access to better QoS. We then outline the architectural elements of the framework in Section 4 and describe the functionality assumed for each component. Section 5 discusses example policies, possible scenarios, and policy support needed for those scenarios. Section 6 evaluates existing solutions such as RADIUS and LDAP and discusses their applicability to the problem of policy control.

2. Terminology

The following is a list of terms used in this document.

- Administrative Domain: A collection of networks under the same administrative control and grouped together for administrative purposes.

- Network Element or Node: Routers, switches, hubs are examples of network nodes. They are the entities where resource allocation decisions have to be made and the decisions have to be enforced. A RSVP router which allocates part of a link capacity (or buffers) to a particular flow and ensures that only the admitted flows have access to their reserved resources is an example of a network element of interest in our context.

  In this document, sometimes we use the terms router, network element, and network node interchangeably, but should be interpreted as reference to a network element.

- QoS Signaling Protocol: A signaling protocol that carries an admission control request for a bandwidth resource, e.g., RSVP.

- Policy: The combination of rules and services where rules define the criteria for resource access and usage.

- Policy control: The application of rules to determine whether or not access to a particular resource should be granted.

- Policy Object: Contains policy-related info such as policy elements and is carried by the QoS signaling protocol.

- Policy Element: Subdivision of policy objects; contains single units of information necessary for the evaluation of policy rules. A single policy element carries an user or application identification whereas another policy element may carry user credentials or credit card information. Examples of policy elements include identity of the requesting user or application, user/app credentials, etc. The policy elements themselves are expected to be independent of which QoS signaling protocol is used.

- Policy Decision Point (PDP): The point where policy decisions are made.

- Policy Enforcement Point (PEP): The point where the policy decisions are actually enforced.

- Policy Ignorant Node (PIN): A network element that does not explicitly support policy control using the mechanisms defined in this document.

- Resource: Something of value in a network infrastructure to which rules or policy criteria are first applied before access is granted. Examples of resources include the buffers in a router and bandwidth on an interface.

- Service Provider: Controls the network infrastructure and may be responsible for the charging and accounting of services.

- Soft State Model - Soft state is a form of the stateful model that times out installed state at a PEP or PDP. It is an automatic way to erase state in the presence of communication or network element failures. For example, RSVP uses the soft state model for installing reservation state at network elements along the path of a data flow.

- Installed State: A new and unique request made from a PEP to a PDP that must be explicitly deleted.

- Trusted Node: A node that is within the boundaries of an administrative domain (AD) and is trusted in the sense that the admission control requests from such a node do not necessarily need a PDP decision.

3. Policy-based Admission Control: Goals and Requirements

In this section, we describe the goals and requirements of mechanisms and protocols designed to provide policy-based control over admission control decisions.

- Policies vs Mechanisms: An important point to note is that the

framework does not include any discussion of any specific policy behavior or does not require use of specific policies. Instead, the framework only outlines the architectural elements and mechanisms needed to allow a wide variety of possible policies to be carried out.

- RSVP-specific: The mechanisms must be designed to meet the policy-based control requirements specific to the problem of bandwidth reservation using RSVP as the signaling protocol. However, our goal is to allow for the application of this framework for admission control involving other types of resources as long as we do not diverge from our central goal.

- Support for preemption: The mechanisms designed must include support for preemption. By preemption, we mean an ability to remove a previously installed state in favor of accepting a new admission control request. For example, in the case of RSVP, preemption involves the ability to remove one or more currently installed reservations to make room for a new resource reservation request.

- Support for many styles of policies: The mechanisms designed must include support for many policies and policy configurations including bi-lateral and multi-lateral service agreements and policies based on the notion of relative priority. In general, the determination and configuration of viable policies are the responsibility of the service provider.

- Provision for Monitoring and Accounting Information: The mechanisms must include support for monitoring policy state resource usage and provide access information. In particular, mechanisms must be included to provide usage and access information that may be used for accounting and billing purposes.

- Fault tolerance and recovery: The mechanisms designed on the basis of this framework must include provisions for fault tolerance and recovery from failure cases such as failure of PDPs, disruption in communication including network partitions (and subsequent merging) that separate a PDP from its peer PEPs.

- Support for Policy-Ignorant Nodes (PINs): Support for the mechanisms described in this document should not be mandatory for every node in a network. Policy based admission control could be enforced

at a subset of nodes, for example the boundary nodes within an administrative domain. These policy capable nodes would function as trusted nodes from the point of view of the policy-ignorant nodes in that administrative domain.

- Scalability: One of the important requirements for the mechanisms designed for policy control is scalability. The mechanisms must scale at least to the same extent that RSVP scales in terms of accommodating multiple flows and network nodes in the path of a flow. In particular, scalability must be considered when specifying default behavior for merging policy data objects and merging should not result in duplicate policy elements or objects. There are several sensitive areas in terms of scalability for policy control over RSVP. First, not every policy aware node in an infrastructure should be expected to contact a remote PDP. This would cause potentially long delays in verifying requests that must travel up hop by hop. Secondly, RSVP is capable of setting up resource reservations for multicast flows. This implies that the policy control model must be capable of servicing the special requirements of large multicast flows. Thus, the policy control architecture must scale at least as well as RSVP based on factors such as the size of RSVP messages, the time required for the network to service an RSVP request, local processing time required per node, and local memory consumed per node.

- Security and denial of service considerations: The policy control architecture must be secure as far as the following aspects are concerned. First, the mechanisms proposed under the framework must minimize theft and denial of service threats. Second, it must be ensured that the entities (such as PEPs and PDPs) involved in policy control can verify each other's identity and establish necessary trust before communicating.

4. Architectural Elements

The two main architectural elements for policy control are the PEP (Policy Enforcement Point) and the PDP (Policy Decision Point). Figure 1 shows a simple configuration involving these two elements; PEP is a component at a network node and PDP is a remote entity that may reside at a policy server. The PEP represents the component that always runs on the policy aware node. It is the point at which policy decisions are actually enforced. Policy decisions are made primarily at the PDP. The PDP itself may make use of additional mechanisms and protocols to achieve additional functionality such as user authentication, accounting, policy information

storage, etc. This document does not include discussion of these additional mechanisms and protocols and how they are used.

The basic interaction between the components begins with the PEP. The PEP will receive a notification or a message that requires a policy decision. Given such an event, the PEP then formulates a request for a policy decision and sends it to the PDP. The request for policy control from a PEP to the PDP may contain one or more policy elements (encapsulated into one or more policy objects) in addition to the admission control information (such as a flowspec or amount of bandwidth requested) in the original message or event that triggered the policy decision request. The PDP returns the policy decision and the PEP then enforces the policy decision by appropriately accepting or denying the request. The PDP may also return additional information to the PEP which includes one or more policy elements. This information need not be associated with an admission control decision. Rather, it can be used to formulate an error message or outgoing/forwarded message.

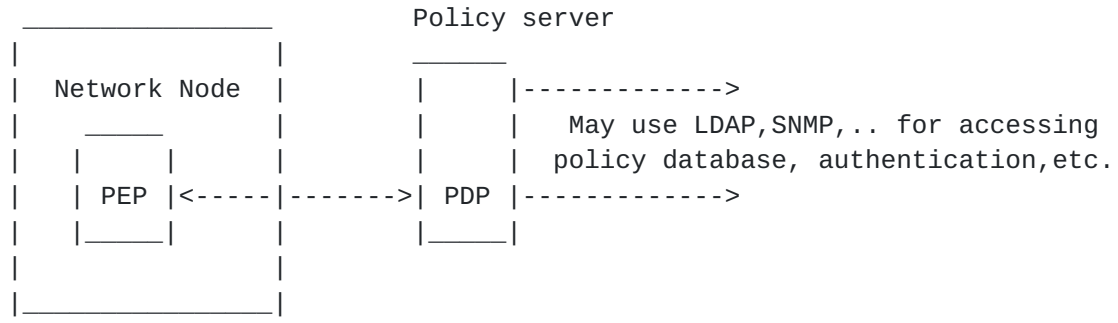


Figure 1: A simple configuration with the primary policy control architecture components. PDP may use additional mechanisms and protocols for the purpose of accounting, authentication, policy storage, etc.

In some cases, the simple configuration shown in Figure 1 may not be sufficient as it might be necessary to apply local policies (e.g., policies specified in access control lists) in addition to the policies applied at the remote PDP. In addition, it is possible for the PDP to be co-located with the PEP at the same network node. Figure 2 shows the possible configurations.

The configurations shown in Figures 1 and 2 illustrate the flexibility in division of labor. On one hand, a centralized policy server, which could be responsible for policy decisions on behalf of multiple network nodes in an administrative domain, might be implementing policies of a wide scope, common across the AD. On the other hand, policies which depend on information and conditions local to a particular router and which are more dynamic, might be better implemented locally, at the router.

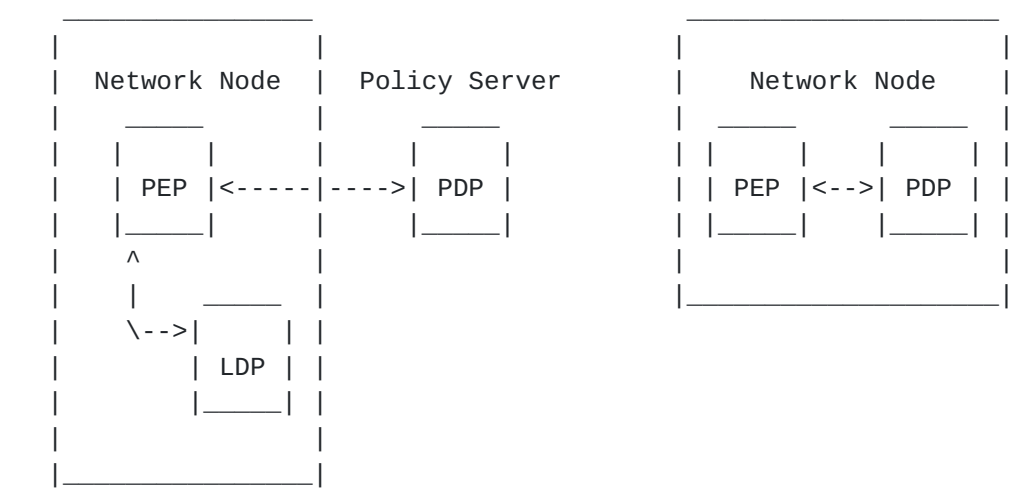


Figure 2: Two other possible configurations of policy control architecture components. The configuration on left shows a local decision point at a network node and the configuration on the left shows PEP and PDP co-located at the same node.

If it is available, the PEP will first use the LDP to reach a local decision. This partial decision and the original policy request are next sent to the PDP which renders a final decision (possibly, overriding the LDP). It must be noted that the PDP acts as the final authority for the decision returned to the PEP and the PEP must enforce the decision rendered by the PDP. Finally, if a shared state has been established for the request and response between the PEP and PDP, it is the responsibility of the PEP to notify the PDP that the original request is no longer in use.

Unless otherwise specified, we will assume the configuration shown on the left in Figure 2 in the rest of this document.

Under this policy control model, the PEP module at a network node must use the following steps to reach a policy decision:

1. When a local event or message invokes PEP for a policy decision, the PEP creates a request that includes information from the message (or local state) that describes the admission control request. In addition, the request include appropriate policy elements to the request as described below.

2. The PEP may consult a local configuration database to identify a set of policy elements (called set A) that are to be evaluated locally. The local configuration specifies the types of policy elements that are evaluated locally. The PEP passes the request with

the set A to the Local Decision point (LDP) and collects the result of the LDP (called "partial result" and referred to as D(A)).

3. The PEP then passes the request with ALL the policy elements and D(A) to the PDP. The PDP applies policies based on all the policy elements and the request and reaches a decision (let us call it D(Q)). It then combines its result with the partial result D(A) using a combination operation to reach a final decision.

4. The PDP returns the final policy decision (one after the combination operation) to the PEP.

Note that in the above model, the PEP *must* contact the PDP even if no or NULL policy objects are received in the admission control request. This requirement would help ensure that a request cannot bypass policy control by omitting policy elements in a reservation request. However, ``short circuit'' processing is permitted, i.e., if the result of D(A), above, is ``no'', then there is no need to proceed with further policy processing at the policy server. Still, the PDP must be informed of the failure of local policy processing. The same applies to the case when policy processing is successful but admission control (at the resource management level due to unavailable capacity) fails; again the policy server has to be informed of the failure.

It must also be noted that the PDP may, at any time, send an asynchronous notification to the PEP to change its earlier decision or to generate a policy error/warning message.

4.1. Example of a RSVP Router

In the case of a RSVP router, Figure 3 shows the interaction between a PEP and other int-serv components within the router. For the purpose of this discussion, we represent all the components of RSVP-related processing by a single RSVP module, but more detailed discussion of the exact interaction and interfaces between RSVP and PEP will be described in a separate document [3].

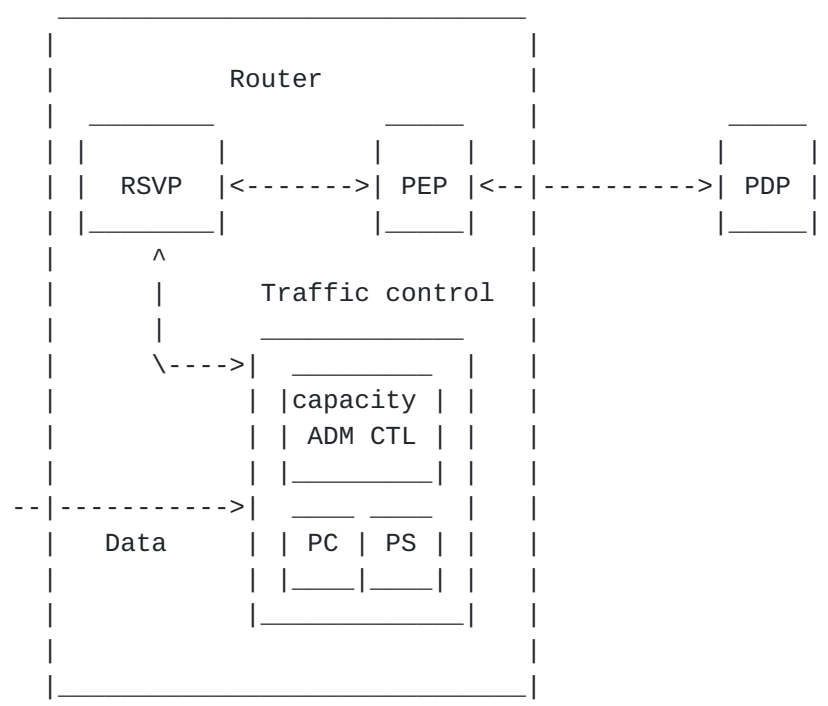


Figure 3: Relationship between PEP and other int-serv components within an RSVP router. PC -- Packet Classifier, PS -- Packet Scheduler

When a RSVP message arrives at the router (or an RSVP related event requires a policy decision), the RSVP module is expected to hand off the request (corresponding to the event or message) to its PEP module. The PEP will use the PDP (and LDP) to obtain the policy decision and communicate it back to the RSVP module.

4.2. Additional functionality at the PDP

Typically, PDP returns the final policy decision based on an admission control request and the associated policy elements. However, it should be possible for the PDP to sometimes ask the PEP (or the admission control module at the network element where PEP resides) to generate policy-related error messages. For example, in the case of RSVP, the PDP may accept a request and allow installation and forwarding of a reservation to a previous hop, but, at the same time, may wish to generate a warning/error message to a downstream node (NHOP) to warn about conditions such as "your request may have to be torn down in 10 mins, etc." Basically, an ability to create policy-related errors and/or warnings and to propagate them using the native QoS signaling protocol (such as RSVP) is needed. Such a policy error returned by the PDP must be able to also specify whether the reservation request should still be accepted, installed, and forwarded to allow continued normal RSVP processing. In particular, when a PDP sends back an error, it specifies that:

1. the message that generated the adm control request should be processed further as usual, but an error message (or warning) be sent in the other direction and include the policy objects supplied in that error message

2. or, specifies that an error be returned, but the RSVP message should not be forwarded as usual.

OPEN ISSUE -- What happens in case of the blockade state in RSVP?

4.3. Interactions between PEP, LDP, and PDP at a RSVP router

[NOTE: This section only reflects the ongoing discussion].

At the interim meeting held in October, the working group members present discussed the issue of defining the interaction (and interfaces) among different policy control components at a RSVP router. The discussion was inconclusive and the following lists the items covered and the issues raised for information purpose:

* PEP is invoked whenever RSVP events happen. Examples of events are timeout, refresh events, a message arrives on an incoming interface or is to be sent on an outgoing interface or reservations have to be merged and installed or forwarded on an interface.

* PEP is responsible for notifying RSVP whenever asynch notifications come from the PDP.

* PEP may cache the results returned by the PDP for later use to avoid checking with the PDP every time. However, this raises an interesting issue as the PEP must be able to detect changes to the previously received policy elements/objects so that the PEP can contact the PDP whenever changes occur.

* LDP is optional and may be used for making decisions based on policy elements handled locally. The LDP, in turn, may have to go to external entities (such as a directory server or an authentication server, etc.) for making its decisions.

* PDP is stateful and may make decisions even if no policy objects are received (e.g., make decisions based on information such as flowspecs and session object in the RSVP messages). The PDP may consult other PDPs, but discussion of inter-PDP communication and coordination is outside the scope of this document

* PDP sends asynchronous notifications to PEP whenever necessary to change earlier decisions, generate errors etc.

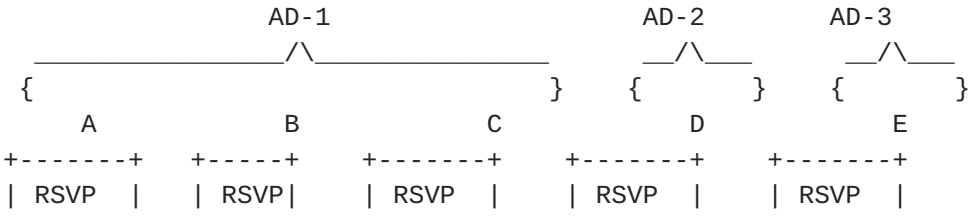
* PDP exports the information useful for usage monitoring and accounting purposes: Examples of such information may include a MIB.

* How or when to merge policy elements and default rules on merging? In case of RSVP node without a PEP, what and if default behavior for merging should be specified?

4.4. Placement of Policy Elements in a Network

By allowing division of labor between an LDP and a PDP, the policy control architecture allows staged deployment by enabling routers of varying degrees of sophistication, as far as policy control is concerned, to communicate with policy servers. Figure 4 depicts an example set of nodes belonging to three different administrative domains (AD) (Each AD could correspond to a different service provider in this case). Nodes A, B and C belong to administrative domain AD-1, advised by PDP PS-1, while D and E belong to AD-2 and AD-3, respectively. E communicates with PDP PS-3. In general, it is expected that there will be at least one PDP per administrative domain.

Policy capable network nodes could range from very unsophisticated, such as E, which have no LDP, and thus have to rely on an external PDP for every policy processing operation, to self-sufficient, such as D, which essentially encompasses both an LDP and a PDP locally, at the router.



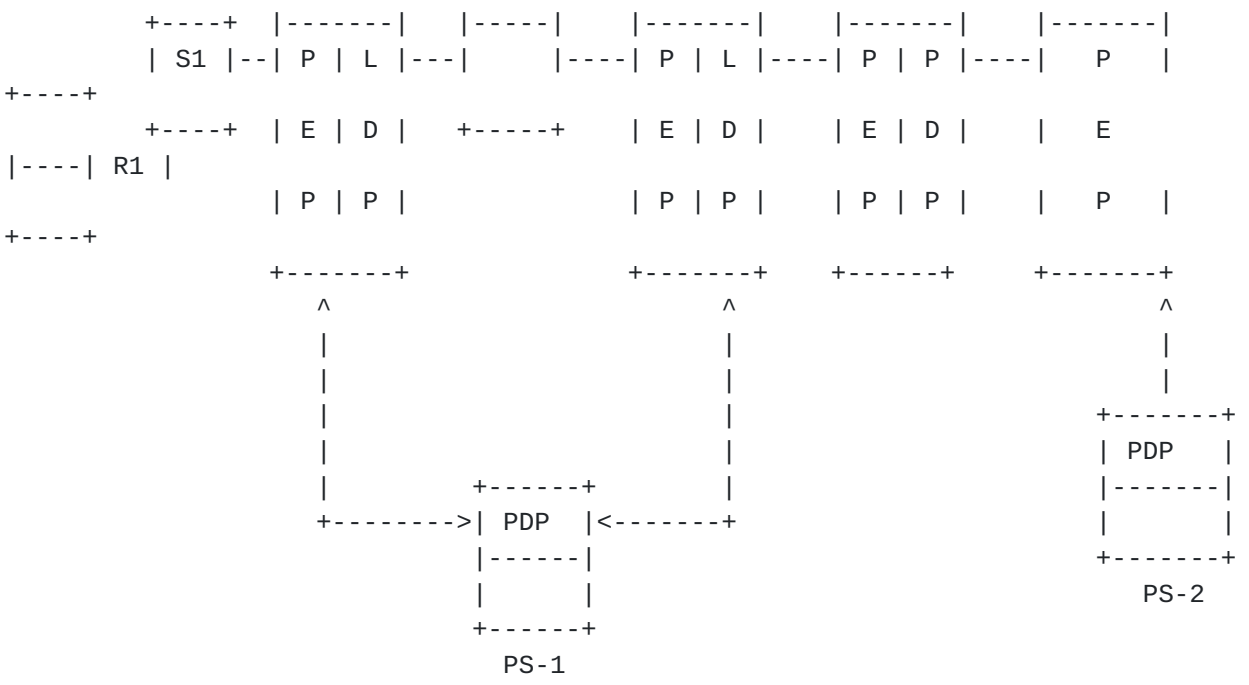


Figure 4: Placement of Policy Elements in an internet

5. Example Policies, Scenarios, and Policy Support

In the following, we present examples of desired policies and scenarios requiring policy control that should possibly be addressed by the policy control framework. In some cases, possible approach(es) for achieving the desired goals are also outlined with a list of open issues to be resolved.

5.1. Admission control policies based on factors such as Time-of-Day, User Identity or credentials

Policy control must be able to express and enforce rules with temporal dependencies. For example, a group of users might be allowed to make reservations at certain levels only during off-peak hours. In addition, the policy control must also support policies that take into account identity or credentials of users requesting a particular service or resource. For example, an RSVP reservation request may be denied or accepted based on the credentials or identity suppled in the request.

5.2. Bilateral agreements between service providers

Today, usage agreements between service providers for traffic crossing their boundaries tend to be quite simple, for example, two

ISPs might agree to accept all traffic from each other, often without performing any accounting or billing for the ``foreign'' traffic carried. There are strong reasons to expect that such a model cannot survive once traffic differentiation and quality of service guarantees begin to be phased in the Internet. Once ISPs start charging end users based on usage and quality of service, it is only natural that they will also seek mechanisms for charging each other for reservations transiting their networks. One additional incentive in establishing such mechanisms is the potential asymmetry in terms of the customer base that different providers will exhibit: ISPs focused on servicing corporate traffic are likely to experience much higher demand for reserved services than those that service the consumer market. Lack of sophisticated accounting schemes for inter-ISP traffic could lead to inefficient allocation of costs among different service providers. (ISSUE: ISn't this an accounting issue among ISPs? We can only provide mechanism to facilitate resolution of such issues, but not suggest solutions.)

Bilateral agreements could fall into two broad categories. In the first, providers which manage a network cloud or administrative domain contract with their closest point of contact (neighbor) to establish ground rules and arrangements for access control and accounting. These contracts are mostly local and do not rely on global agreements; consequently, a policy node maintains information about its neighboring nodes only. Referring to Figure 4, this model implies that provider AD-1 has established arrangements with AD-2, but not with AD-3, for usage of each other's network. Provider AD-2, in turn, has in place agreements with AD-3 and so on. Thus, when forwarding a reservation request to AD-2, provider AD-2 will charge AD-1 for use of all resources beyond AD-1's network. This information is obtained by recursively applying the bilateral agreements at every boundary between (neighboring) providers, until the recipient of the reservation request is reached. To implement this scheme under the policy control architecture, boundary nodes have to add an appropriate policy object to the RSVP message before forwarding it to a neighboring provider's network. This policy object will contain information such as the identity of the provider that generated them and the equivalent of an account number where charges can be accumulated. Since agreements only hold among neighboring nodes, *policy objects have to be rewritten* as RSVP messages cross the boundaries of administrative domains or provider's networks.

Bilateral agreements could also be established on a *global* scale. In this second category, providers contract with an arbitrary number of other providers, not necessarily neighboring. The objective of global agreements would be to provide a global network

picture that allows a provider to satisfy (most) reservation requests without reliance on third party agreements. This category of agreements can also be supported in the context of the policy control architecture. As above, the originating node includes a policy object containing identity and account information in the RSVP message; however, as long as global agreements are in place, this object is not rewritten as the message crosses administrative boundaries. Instead, each provider ``charges'' to the account the incremental cost incurred in carrying the reservation request through its network.

5.3. Pre-paid calling card or Tokens

A model of increasing popularity in the telephone network is that of the pre-paid calling card. This concept could also be applied to the Internet; users purchase ``tokens'' which can be redeemed at a later time for access to network services. When a user makes a reservation request through, say, an RSVP RESV message, the user supplies a unique identification number of the ``token'', embedded in a policy object. Processing of this object at policy capable routers results in decrementing the value, or number of remaining units of service, of this token.

Referring to Figure 4, suppose receiver R1 in the administrative domain AD3 wants to request a reservation for a service originating in AD1. R1 generates a policy data object of type PD(prc, CID), where ``prc'' denotes pre-paid card and CID is the card identification number. Along with other policy objects carried in the RESV message, this object is received by node E, which forwards it to its PEP, PEP_E, which, in turn, contacts PDP PS-3. PS-3 either maintains locally, or has remote access to, a database of pre-paid card numbers. If the amount of remaining credit in CID is sufficient, the PDP accepts the reservation and the policy object is returned to PEP_E. Two issues have to be resolved here:

* What is the scope of these charges?

* When are charges (in the form of decrementing the remaining credit) first applied?

The answer to the first question is related to the bilateral agreement model in place. If, on the one hand, provider AD-3 has established agreements with both AD-2 and AD-1, it could charge for the cost of the

complete reservation up to sender S1. In this case PS-2 removes the PD(prc,CID) object from the outgoing RESV message.

On the other hand, if AD-3 has no bilateral agreements in place, it will simply charge CID for the cost of the reservation within AD-3 and then forward PD(prc,CID) in the outgoing RESV message. Subsequent PDPs in other administrative domains will charge CID for their respective reservations. Since multiple entities are both reading (remaining credit) and writing (decrementing credit) to the same database, some coordination and concurrency control might be needed. The issues related to location, management, coordination of credit card (or similar) databases is outside the scope of this document.

Another problem in this scenario is determining when the credit is exhausted. The PDPs should contact the database periodically to submit a charge against the CID; if the remaining credit reaches zero, there must be a mechanism to detect that and to cause revocation or termination of privileges granted based on the credit.

Regarding the issue of when to initiate charging, ideally that should happen only after the reservation request has succeeded. In the case of local charges, that could be communicated by the router to the PDP.

5.4. Priority based admission control policies

In many settings, it is useful to distinguish between reservations on the basis of some level of "importance". For example, this can be useful to avoid that the first reservation being granted the use of some resources, be able to hog those resources for some indefinite period of time. Similarly, this may be useful to allow emergency calls to go through even during periods of congestion. Such functionality can be supported by associating priorities to reservation requests, and conveying this priority information together with other policy information.

In its basic form, the priority associated with a reservation directly determines a reservation's rights to the resources it requests. For example, assuming that priorities are expressed through integers in the range 0 to 32 with 32 being the highest priority, a reservation of priority, say, 10, will always be accepted, if the amount of resources held by lower priority reservations is sufficient to satisfy its requirements. In other words, in case there are not enough free resources (bandwidth, buffers, etc.) at a node to accommodate the priority 10 request, the node will attempt to free up the necessary resources by preempting existing lower priority reservations.

There are a number of requirements associated with the support of

priority and their proper operation. First, traffic control in the router needs to be aware of priorities, i.e., classify existing reservations according to their priority, so that it is capable of determining how many and which ones to preempt, when required to accommodate a higher priority reservation request. Second, it is important that preemption be made consistently at different nodes, in order to avoid transient instabilities. Third and possibly most important, merging of priorities needs to be carefully architected and its impact clearly understood as part of the associated policy definition.

Of the three above requirements, merging of priority information is the more complex and deserves additional discussions. The complexity of merging priority information arises from the fact that this merging is to be performed in addition to the merging of reservation information. When reservation (FLOWSPEC) information is identical, i.e., homogeneous reservations, merging only needs to consider priority information, and the simple rule of keeping the highest priority provides an adequate answer. However, in the case of heterogeneous reservations, the * two-dimensional nature} of the (FLOWSPEC, priority) pair makes their ordering, and therefore merging, difficult. Two possible cases can be identified:

1. (FLOWSPEC1, priority1)= (high, high);
   (FLOWSPEC2, priority2)= (low, low)
2. (FLOWSPEC1, priority1) = (high, low);
   (FLOWSPEC2, priority2) = (low, high)

In the first case, the ordering is immediate, i.e.,

(FLOWSPEC1, priority1) >= (FLOWSPEC2, priority2),

so that the merged value can readily be chosen as (FLOWSPEC1, priority1). However, in the second case, no obvious ordering is available, and each possible merged combination introduces a problem of its own which we briefly review.

2.a (FLOWSPEC_OUT, priority_OUT) = (high, high) = {FLOWSPEC1, priority2}

The main problem with this selection is the so-called "free-rider" problem, in that the high bandwidth requirement of a low priority request is now entitled to the higher priority that was initially granted to a low bandwidth request. Such "upgrades" can all but disable the ability of priorities to give preference to certain flows, and are therefore not acceptable.

2.b (FLOWSPEC_OUT, priority_OUT) = (high, low) = (FLOWSPEC1,

priority1)

This selection amounts to (initially) giving preference to high bandwidth requests, irrespective of their priority. The main benefit is that larger reservations are allowed to try and succeed in case the resources are available. The main disadvantage is that if a high FLOWSPEC and low priority reservation is preempted, the entire reservation is taken down. As a result, a high priority and low FLOWSPEC reservation is left without resources until a new reservation is ultimately reestablished at the higher priority level.

2.c (FLOWSPEC_OUT, priority_OUT) = (low, high) = (FLOWSPEC2, priority2)

This is the opposite selection to 2.b, and as a result, it has the opposite disadvantages and benefits. Specifically, a low FLOWSPEC but high priority request can preclude higher FLOWSPEC but lower priority requests, *irrespective* of how lightly loaded the network is. On the other hand, the high priority request is *guaranteed* unperturbed service unless it itself needs to be preempted.

2.d (FLOWSPEC_OUT, priority_OUT) = (low, low)

This last option is mentioned for completeness, but is not particularly meaningful as it offers no benefits and only disadvantages.

As can be seen from the above discussion, support for priorities and the associated merging operations raises a number issues. There are possible solutions to most of the above problems, e.g.,

- Disable or limit the amount of merging that can take place, - Select one of the above rules 2.b or 2.c based on the relative difference between the FLOWSPEC being merged,

but it is likely that different environments (set of criteria) may mandate different selections. In addition, it may be desirable to extend the concept of priorities to let applications specify both holding and preemption priorities, with the former being always greater than the latter. The preemption priority would be a function of the urgency of satisfying the reservation request being made, while the holding priority would reflect the application's tolerance to being interrupted.

In general, support for priority provides a powerful tool to manage access to network resources, and should represent one of the key benefits of the introduction of policy support. However, rules determining how priorities are to be handled may differ, and likely to be specified

in the context of each policy supporting priorities. (OPEN ISSUE: Should we specify default merging rules?)

5.5. Sender Specified Restrictions on Receiver Reservations

The ability of senders to specify restrictions on reservations, based on receiver identity, number of receivers or reservation cost might be useful in future network applications. An example could be any application in which the sender pays for service delivered to receivers. In such a case, the sender might be willing to assume the cost of a reservation, as long as it satisfies certain criteria, for example, it originates from a receiver who belongs to an access control list (ACL) and satisfies a limit on cost. (Notice that this could allow formation of "closed" multicast groups).

In the policy based admission control framework such a scheme could be achieved by having the sender generate appropriate policy objects, carried in a PATH message, which install state in routers on the path to receivers. In accepting reservations, the routers would have to compare the RESV requests to the installed state.

A number of different solutions can be built to address this scenario; precise description of a solution is beyond the scope of this document.

6. Evaluation of Existing Frameworks or Solutions

6.1. RADIUS

The Remote Authentication Dial In User Service (RADIUS) [REF] protocol, specifies how authentication, authorization and configuration information is exchanged between a network access server wishing to authenticate its users and a (shared) authentication server. RADIUS has been designed to operate at the level of a * session}, which is defined as the interval between the time that service is first provided and the time service is ended. The latter definition is geared towards login-type service, for example network access for dial up users through PPP.

RADIUS is an important protocol for authenticating login users. However, some of its characteristics make it unsuitable for use, at least in its current form, in the context of policy control for QoS reservations:

* Use of the UDP protocol: The sensitivity of policy control information requires reliable operation. Use of UDP would mandate specifying retry and fallback algorithms, which can be quite

complicated. In addition, while timing requirements for a user logging onto a network might permit a delay of several seconds ([REF]), this will not be the case for reservations proceeding in the network. Moreover, acknowledgments are essential for actions like billing and accounting.

* Difficulty in carrying opaque objects: Use of RADIUS for QoS policy control would require defining additional RADIUS * attributes. To carry opaque objects of variable length, the format of RADIUS attributes would have to be extended; it currently supports four data types of pre-specified length; string, address, integer and time.

* No support for asynchronous notification: Asynchronous notification is required in order to allow both the policy server and client to notify each other in the case of an asynchronous change in state, i.e., a change that is not triggered by a signaling message. For example, the server would need to notify the client if a particular reservation has to be terminated due to expiration of a user's credentials or account balance. Likewise, the client has to inform the server of a reservation rejection which is due to admission control failure.

* Restricted Message Types: The restriction to messages of type ``access-request'' and ``access-accept'' (or reject) does not facilitate information gathering for monitoring and accounting purposes. New message types would have to be defined.

* Multicast Groups: It is not clear how requests from a group of multicast receivers could be handled in the context of RADIUS.

* Identifying Requests: Several changes will be needed in the way requests are identified. Currently this is done using a user name/IP address attribute. This might be too narrow for reservation protocols, i.e. one might need to specify a source port, a destination address/port and provide some user credentials as well.

* QoS Specification: Changes will be needed to incorporate the new service (resource reservation) and the ability to specify the desired level of service (QoS, Tspec)

In short, we believe that if the RADIUS protocol were to be used for the purpose of QoS policy control, the number of changes required would make the task as difficult or more than defining a new protocol designed with QoS reservations in mind.

6.2. LDAP

LDAP is very effective as a directory service protocol. Nevertheless, it restricts the flexibility of an implementation by requiring a very specific and well understood format of policy information and policy rules, common to both a policy server and a client (e.g., a standardized schema). The client must be able to interpret the format of policy information and rules directly. In addition, LDAP is more suitable for accessing directory information, information that is predefined and changes infrequently.

Given the rather experimental state of policy control for QoS reservation protocols, the dynamic nature of policy decisions and rules and the limitations of clients (routers) in interpreting policy information, an LDAP based solution for client-server communication would be inadequate at this point. Also, a wideer variety of policy element types and objects need to be represented using a policy control protocol than supported by LDAP.

However, LDAP could conceivably be used in downloading policy rules from an LDAP server to policy servers. Furthermore, future versions of LDAP promise to make it more suitable for the exchange of dynamic information. As these modifications are introduced the role of LDAP in QoS policy control should be reevaluated.

7. Security Considerations

The communication tunnel between policy clients and policy servers should be secured by the use of an IPSEC [4] channel. It is advisable that this tunnel makes use of both the AH (Authentication Header) and ESP (Encapsulating Security Payload) protocols, in order to provide confidentiality, data origin authentication, integrity and replay prevention.

In the case of the RSVP signaling mechanism, RSVP MD5 [2] message authentication can be used to secure communications between network elements.

8. References

[1] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification ", RFC 2205, September 1997.

[2] F. Baker., "RSVP Cryptographic Authentication", draft-ietf-rsvp-md5-05.txt, August 1997.

[3] S. Herzog., "RSVP Extensions for Policy Control", Internet Draft}, draft-ietf-rsvp-policy-ext-02.[ps,txt], Apr. 1997.

[4] R. Atkinson. Security Architecture for the Internet Protocol. RFC1825, Aug. 1995.

[5] C. Rigney, A Rubens, W. Simpson and S. Willens. Remote Authentication Dial In User Service (RADIUS). RFC 2138.

8. Acknowledgements

This is a result of an ongoing discussion among many members of the RAP group including Jim Boyle, Ron Cohen, Laura Cunningham, Dave Durham, Shai Herzog, Tim O'Malley, Raju Rajan, and Arun Sastry.

9. Authors` Addresses

Raj Yavatkar
Intel Corporation
2111 N.E. 25th Avenue,
Hillsboro, OR 97124
USA
phone: +1 503-264-9077
email: yavatkar@ibeam.intel.com

Dimitrios Pendarakis
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights
NY 10598
phone: +1 914-784-7536
email: dimitri@watson.ibm.com

Roch Guerin
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights
NY 10598
phone: +1 914-784-7038
email: guerin@watson.ibm.com