Internet Draft
Expires May 22, 1997
File: draft-ietf-rsvp-policy-arch-01.ps

Shai Herzog
IBM T.J. Watson Research Center
November 1996

Policy Control for RSVP: Architectural Overview

November 22, 1996

Status of Memo

Abstract

This memo provides insight into some possible approaches for policy enforcement in resource reservation protocols. We present sample scenarios for each of these approaches as a way to demonstrate their feasibility, and to motivate the development of supporting architectures.

1 Introduction

Reservation protocols, by definition, discriminate between users, by providing some users with better service at the expense of others. Therefore, it is reasonable to expect that these protocols be accompanied by mechanisms for controlling and enforcing access policies. In this document, we refer to such policies as "policy control". The term "policy control" is quite broad; it ranges from simple access control to sophisticated accounting and debiting mechanisms.

Throughout this document we use the terms "reservation protocols" and "RSVP" interchangeably. However, the contents of this document could be applied to similar resource reservation protocols as well. The current admission process in RSVP uses resource (capacity) based admission control; we expand this model to include policy based admission control. Policy admission control is enforced locally at border/policy nodes (also see [LPM]). However, the local decision may be based on policy information provided at reservation setup time (e.g., POLICY_DATA objects in RSVP). This information propagates hop-by-hop by policy nodes, and may be modified by each of these nodes (subject to the applicable agreements, and local policies). Credentials are among the most useful and common policy information; they allow local policies to discriminate between various usage groups by describing the "originator/s" of the reservation request.[1] For scaling reasons, we concentrate on policies that follow a bilateral-agreement model. The bilateral model assumes that network clouds (providers) 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. The bilateral model has similar scaling properties to multicast and is easier to maintain in distributed environments.

In this document, we outline a few sample scenarios for access control and accounting; we provide these scenarios as motivation and as needed context for the development of policy control architectures for resource reservation protocols. These scenarios are based on two simple assumptions: (1) RSVP provides the transport service which carries policy control state (POLICY_DATA objects), hop-by-hop. (2) Policies are enforced locally, and can be based, among other factors, on bilateral agreements between neighboring providers, local policies, and the contents of POLICY_DATA objects; as well as other factors.

2 Simple access control

To provide simple access control, the local node attempts to match incoming policy objects with one or more of the pre-configured policies or bilateral agreements.

Consider the following network scenario: one receiver from ISI and two from MIT listen to

---

[1] Clearly, in the multicast case, credentials may describe only a merged aggregate rather than the original credentials of all the end-receivers.

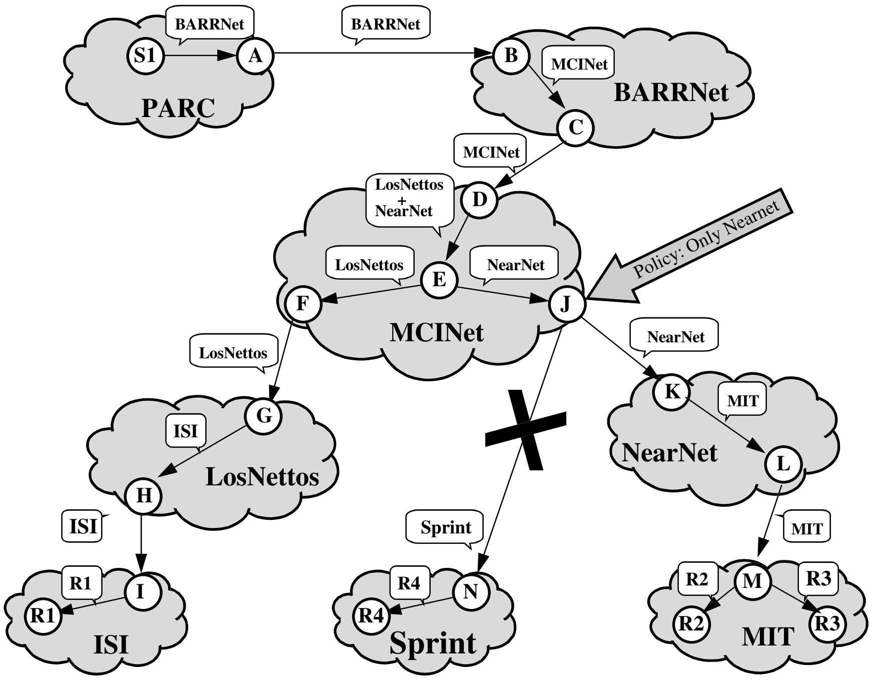

Figure 1: Simple access control

a PARC seminar. For simplicity we limit ourselves to **receiver based** access control.

Bilateral agreements between each two neighboring providers (e.g., R1, R2 with ISI, ISI with LosNettos,... BARRNet with PARC) are simple: the first provider obtains permission to make reservations over the second provider's network. The notation $PD(cr, uid)$ represents a policy data object of type "cr" (credential) verifying that the reservation is associated with uid. Credentials may have an underlying hierarchy; Quite often, individual end-users belong to a stub or local service provider. These in-turn, contract with regional service providers who connect with backbone providers. Such hierarchy provides a natural way of aggregating receiver credentials along the reserved tree (by rewriting and merging credentials on a hop-by-hop basis according to locally configured conversion tables).

Figure 1 illustrates a reservation scenario. A typical example of a bilateral agreement could be between MCI and LosNettos: MCI would allow the LosNettos users to use its backbone. A policy data object $PD(cr, LosNettos)$ would be interpreted by MCI as a green light to accept the reservation. In this scenario, reservations from R1, R2, and R3 carry policy data objects which propagate hop-by-hop (encapsulated in reservation messages) toward S1.

In this scenario, policy objects are rewritten in nodes B,D,G,I,K, and M (which are entry points to clouds). There are two main reasons for rewriting credentials along the reserved tree. First, effective credentials may loose their validity when a flow crosses administrative boundaries. Second, for scaling reasons, large lists of individual credentials may be merged into hierarchically higher-level credentials.

In our scenario, let us assume that D is configured with the following conversion table:

$PD(cr, LosNettos) \Longrightarrow PD(cr, MCI)$
$PD(cr, NearNet) \quad \Longrightarrow PD(cr, MCI)$

Node D first checks if LosNettos and NearNet are authorized to reserve on their corresponding links and responds accordingly. Assuming authorization is granted, it merges and rewrites these policy objects as $PD(cr, MCI)$ and forwards the reservation to C.

To complicate the example, assume the conversion table was:

$PD(cr, LosNettos) \Longrightarrow PD(cr, MCI1)$
$PD(cr, NearNet) \quad \Longrightarrow PD(cr, MCI2)$

Then node D would forward $PD(cr, MCI1) + PD(cr, MCI2)$ to C instead.

We illustrates a plausible scenario where rewriting of credentials may only takes place at boundary nodes. What is the responsibility of non-policy nodes? In our scenario, node E receives the following policy data objects: F→E: $PD(cr, LosNettos)$ and J→E: $PD(cr, NearNet)$. Because it has no authority to merge or rewrite these credentials, node E must concatenate the two objects and send them as-is ($PD(cr, LosNettos) + PD(cr, NearNet)$) to D.

Reservations arriving with insufficient credentials are subject to rejection. In our scenario, the arrow pointing to node J in Figure 1 illustrates J's local policy to allow only NearNet traffic. As a result, the reservation made by R4 using the credential $PD(cr, Sprint)$ is rejected.

3 Advanced reservation and preemption control

Advanced reservations can be built on top of simple access control: consider the case where every advanced reservation consists of a set of bilateral agreements between different service providers, reserving network capacity at some future period of time. When advanced reservations are not public (i.e., only authorized users can use them), three classes of reservations exist: (1) walk-ins (where the session itself does not have advanced reservations, (2) advanced reservation with unauthorized users, and (3) advanced reservation with authorized users. These numbers (1..3) can define a "preemption priority" (i.e., walk-ins are preempted first, unauthorized pre-reserved second, and authorized pre-reserved are never preempted).

The advanced reservation scenario is almost identical to simple access control: let us assume that each bilateral pre-registration is identified by a PRID (Pre-Registration confirmation ID). Policy data objects of type AR (Advanced Reservation) would take the following form: $PD(ar, prid, uid)$. When an AR object arrives, the local node verifies the existence of pre-reservation *prid*, and checks that *uid* is permitted to use it. Finally, the flow is classified into one of the above three preemptive priorities and RSVP is notified.

4 Quota enforcement/accounting/debiting

The simplest form of policy control performs a binary task: accept or reject based on credentials associated with a reservation. The next step is to allow for more sophisticated policy enforcement that is based on usage feedback. Here we add two additional mechanisms which determines (1) what **debiting mechanism** should be used (if any), and (2)**how much** should be debited for the reservation. The following scenarios assume a preexisting set of local accounts. These accounts are established by bilateral agreements that pre-purchase network capacity and set applicable debiting rules. The role of accounting mechanism is to verify the availability of funds/quotas in these accounts for maintaining the reservation.

In multicast environments, with upstream merging, it is very likely that a reservation will be debited against multiple network entities that represent the aggregated credentials of the downstream receivers. (See node D in Figure 1.) This raises the issue of the "sharing model"; the "sharing model" defines how the reservation is shared among the different policy data objects. There are many possible manners for sharing such costs; some examples may be: (1) Each policy object is allocated the full cost, (2) The cost is divided equally between the different objects (3) The cost is attributed to an individual objects according to the local policy, and(4) The cost is allocated relative to some criteria like the number of downstream receivers, the size of the organization, the amount of pre-purchased capacity (remaining quota), etc. The sharing model, and the selection of cost allocation and actual debiting mechanisms is a local configuration issue which is **not** discussed in this document.

We consider several accounting schemes and briefly describe three: simple debiting, limited debiting, Edge Pricing, and MultiCost (MCost).

1. **Simple debiting**

   Consider the following example: lets assume that LosNettos and NearNet each have a debit account (pre-purchased capacity) with MCI for their traffic. When node E receives $PD(cr, LosNettos)$ and $PD(cr, NearNet)$ for flow f, it must decide the following: (1) How much should be debited for flow f, and (2) how would that debit be shared between the account of LosNettos and NearNet. These are local configuration issues left for service providers. In this scenario, the local node would attempt to perform the debiting, and would notify RSVP of success or failure. The other aspects of the scenario (Merging policy data objects and forwarding them) is identical to that of simple access control.

2. **Limited debiting (willingness to pay)**

   Although we do not have a full understanding of the dynamics of willingness-to-pay and its properties, we can outline the basic scenario, as an extension of the simple debiting model. Willingness to pay is manifested as a limit on the policy object that authorizes the debit. For instance, $PD(crwp, ISI, 10\%\ of\ unicast)$ would represent a policy data object of type *crwp* (Credential, Willingness to Pay), that authorizes debiting the ISI account up to 10% of the unicast cost. Here, the basic idea is that market forces would be the driving force behind what users specify as their willingness to pay.

3. **Edge Pricing**

   A new paradigm, "Edge Pricing" [She96] argues that pricing of network services can be performed at a single point which is the edge of the network (i.e., the network access point). Here, users only interact (for prices and accounting) with their immediate (edge) service provider. To price non-local traffic, the edge provider must be able to estimate, at least, both the path and congestion costs of the admitted traffic. In a simplistic example, geographical addresses (e.g., in IPv6) provide an estimated path cost, and time-of-day may provide a good estimation of congestion costs. (See [She96] for more detailed discussion of such approximations). Edge Pricing can be implemented as an extension of simple debiting; Simple debiting could determines **(which account)** to be debited (based on credentials) and Edge Pricing could estimate the prices to be debited, based on locally available information. However, when accounting for multicast, local information may be insufficient to estimate path costs; multicast uses logical addresses which per-se do not reveal the identity or location of receivers nor to they reveal the topology of the multicast tree. This problem and a proposed accounting solution (MultiCost or MCost) was described in [Her95]. MCost has a unique feature: it takes into account the benefits of sharing a multicast tree and distributes these savings among the members of the multicast group, according to configurable policies, basic fairness, and equality. MCost computes the share of the multicast cost allocated to each end-user; this cost is an important component, that may be used for estimating multicast prices at the Edge.

5 Acknowledgment

This document incorporates inputs from Deborah Estrin, Bob Braden, and Scott Shenker and feedback from RSVP collaborators.

References

[Her95] S. Herzog, S. Shenker and D. Estrin, Sharing the Cost of Multicast Trees: An Axiomatic Analysis, *Proceedings of ACM/SIGCOMM '95*, Cambridge, MA, Aug. 1995

[LPM] S. Herzog Local Policy Modules (LPM): Policy Enforcement for Resource Reservation Protocols. *Internet-Draft*, draft-ietf-rsvp-policy-lpm-01.[ps,txt], Nov. 1996.

[She96] S. Shenker, D. Clark, D. Estrin, and S. Herzog Pricing in Computer Networks: Reshaping the Research Agenda, *Telecommunications Policy*, Vol. 20, No. 1, 1996 also published in *Proceedings of the Twenty-Third Annual Telecommunications Policy Research Conference*, 1995.

Author's Address

Shai Herzog
IBM T. J. Watson Research Center,
P.O. Box 704
Yorktown Heights, NY 10598

Phone: (914) 784-6059
Email: herzogwatson.ibm.com