Internet Draft
Expires September 26, 1997
File: draft-zappala-multicast-routing-arch-00.ps

Daniel Zappala
Bob Braden
USC Information Sciences Institute
Deborah Estrin
USC Computer Science Department
Scott Shenker
Xerox Palo Alto Research Center
March 1997

Interdomain Multicast Routing Support for Integrated Services Networks

March 26, 1997

Status of Memo

Abstract

This document describes an architecture for interdomain multicast routing support of integrated services networks. The key features of this architecture are a multicast route setup protocol and local route construction agents. Together, these two components enable multicast routing to install alternate paths and pinned routes on behalf of receivers that request these services.

1 Introduction

The Internet, with its best-effort service and adaptive routing, has been extremely successful supporting elastic applications [1]. However, best effort service can result in large, and widely varying, end-to-end packet delays. To better support real-time and other inelastic applications for which such vagaries are detrimental, the IETF is in the process of adopting extensions to the Internet's service model and architecture that would allow flows or flow aggregates to reserve specific qualities of service (QoS). These extensions, commonly known as "integrated service", allow state to be installed in routers along the data delivery paths to provide preferential QoS for particular flows or aggregations of flows. This aspect of the architecture is usually known as "resource reservation".

Most of the research on integrated services networks has focussed on the service model extensions and the resource reservation protocol. In contrast, this note is concerned with the routing mechanisms that can improve integrated service. We concentrate on routing-in-the-large, that is, the interdomain routing problem, for which issues of scaling are more acute than for the intradomain routing case. The purpose of this note is to briefly outline our line of research into this problem.

An important component of this new integrated services architecture is the reservation protocol RSVP [16]. This protocol is designed to operate on top of the current routing infrastructure. The current routing infrastructure of the Internet provides *opportunistic shortest-path routing* [5, 8, 6, 10, 12, 9]. By *opportunistic* we mean that routing always utilizes the current shortest path, even if the previously shortest path is still functioning. By *shortest path* we mean that routing uses a single "cost" metric (often just hop-count) and then chooses the "least-cost" path.

At least two problems arise when one relies on the current routing infrastructure to determine the route along which reservation requests and data flow: the *failed primary route problem* and the *opportunistic routing problem.*

- *Failed primary route problem*: An application is limited to using the shortest path as defined by the routing metric. If service is not acceptable along this path, the application has no alternative. In particular, if a flow has requested service along this path and been denied, the application cannot obtain service along any other path. This may lead to situations where many flows are denied service even though other paths could accommodate their service requests.
- *Opportunistic routing problem*: When the shortest path route changes (which may happen not only when the current route fails, but when a "lower cost" route becomes available), the data will immediately change over to this new path, even though no reservation has yet been established. Thus, an application may experience a service disruption even if the previous route is still usable. The length of the service disruption will depend on how quickly RSVP can re-establish the reservation along the new route. If the reservation request is denied, the service disruption could be indefinite. The

extent of this problem is determined by the persistence of routes in the Internet; a recent study [11] indicates that while a significant portion of routes persisted for days, 9% of the routes studied underwent frequent route changes (with a mean time between changes on the order of an hour). With opportunistic routing, there is no guarantee a given route will remain unchanged.

There are two possible approaches to the failed primary route problem: (1) try to choose a primary route that is known a priori to have sufficient resources, thereby ensuring that the service delivered over the path is adequate and that few reservation requests will be denied, or (2) provide a mechanism to choose an alternate route if the primary route has (or other alternate routes have) failed.

There is an effort to realize the first of these approaches, a priori computation of a resource-adequate route, for intradomain routing [15]. This approach requires global distribution and synchronization of link resources and individual flow reservations. Global distribution of such rapidly varying quantities is not, we believe, viable for interdomain routing where issues of scale are paramount.

There have been several proposals for a simplified version of a priori computation that does scale adequately for interdomain routing. In this approach, several different static routing metrics are used; the routers calculate the "least-cost" path for each such metric and keep a routing table for each. This enables routing to provide a few alternate routes with distinctive static service characteristics, such as maximal bandwidth or minimal latency. For example, it can distinguish satellite paths from terrestrial paths, or distinguish very small links from large links. In these proposals, which we refer to as *QoR routing*, applications would specify the appropriate static metric, i.e., the desired quality-of-route (QoR), and the routers would use the appropriate routing entry when forwarding data. Because these metrics are static, this approach has none of the scaling problems of the more dynamic approaches discussed above. QoR routing could provide significant benefits for best-effort service by allowing, for instance, interactive applications to avoid routes involving satellite links, while applications involving asynchronous bulk data transfers could seek out maximal bandwidth paths. For similar reasons, QoR routing could benefit real-time and other inelastic applications. However, the essence of the failed primary route problem remains; an application could only avail itself of one minimal-latency route, and one maximal-bandwidth route, etc.. If a service request was denied along one of these a priori routes, there would be no way of utilizing available bandwidth along other routes with similar properties.

To provide interdomain routing for integrated service, we therefore turn to the second approach, computing an alternate route when necessary. That is, when a reservation request fails, receivers can request that an *alternate path* be constructed. This approach does not have the scaling problems of the first approach, since alternate routes are only computed on demand, and are computed locally, so they do not require globally synchronized state. We propose to *augment* existing routing services with (1) localized route construction, and (2) interdomain routing mechanisms to install and maintain the constructed routes on demand. In particular, we focus on the use of alternate paths for multicast routing.

Even with these architectural extensions, we are still left with the opportunistic routing problem, whereby routes may change while a flow is in progress resulting in service disruption. Our approach to this problem allows receivers to request that routes be *pinned* so that they will only change if the current route ceases to function. We integrate this mechanism into receiver-based multicast routing, allowing receivers with reservations to choose between opportunistic and pinned routes, depending on their preference.

The purpose of this note is to provide a very brief overview of alternate path routing and route pinning within the context of a routing architecture; a more detailed description of the route setup mechanism can be found in [14]. While these routing enhancements were originally motivated by the need to better support certain applications using reservations, they might also be valuable for applications whose service requirements are flexible enough to use (i.e., adapt to) best effort packet delivery, but not flexible enough to cope with arbitrarily long queueing delays that can occur along best effort paths. The rest of the paper is organized as follows. In Section 2 we discuss how the routing enhancements fit into our routing architecture. In Section 3 we discuss how applications use alternate paths and route pinning and briefly describe some of the mechanism involved. Finally, in Section 4 we outline the status of our research in this area, including other possible routing services under consideration.

2 Routing Architecture

We divide routing into three components: a routing protocol, a local route construction agent, and a route setup protocol. The routing protocol uses static routing metrics to install opportunistic routes based on resource capability. The local route construction agent supplements these routes by computing alternate paths. A route setup protocol installs these alternate paths for use by applications and in doing so may override the opportunistic nature of current routing. We separate these facets of routing from reservation setup so that routing services may be used from any reservation protocol, and also so that they may used by applications that don't require a resource reservation.

2.1 Routing Protocol

As discussed above, an internet QoR routing protocol can use static characteristics to calculate and install a limited number of paths conforming to different routing metrics, such as maximal bandwidth and minimal latency. A multicast routing protocol can build QoR multicast trees by using unicast QoR routes to construct the multicast tree. For example, the PIM multicast protocol [2] builds shortest-path multicast trees by sending Join messages along the shortest-path route from receivers to the sender. To instead build a minimal latency multicast tree, PIM could send the Join messages along the minimal latency unicast routes. Note that this leaves open a key design choice. Multicast routing could keep separate trees for each QoR and use ToS bits to determine the tree a given packet uses. Alternatively, multicast routing could keep a single tree, and use some mechanism to enforce homogeneity among the QoRs that receivers choose.

2.2 Route Construction Agent

When needed, a router can also use a local route construction agent to take advantage of a wider range of alternate paths than those the routing protocol could scalably install via a priori calculations. This agent operates independently from a route setup protocol and from other route construction agents. Because it does not need to coordinate its routing decisions with other agents, it can utilize information not captured by the routing protocol's static metrics. This information can include the status of local reservation requests, or resource availability information that is not flooded globally. Because of its independence, the route construction agent may also choose routes that are not necessarily "least-cost"; i.e., the route computation is not distributed and thus need not follow the least-cost route computation approach inherent in the distributed routing.

The local route construction agent also serves an important role in multicast route construction. Due to the dynamic nature of multicast group membership, it is difficult to perform scalable internet multicast route construction. Both centralized computation and global distribution of membership changes are inefficient solutions. To perform scalable route construction, we distribute the computation to the first-hop routers of each local receiver. By using a local route construction agent, a router can construct a route to the source of a multicast tree for any local receivers. A receiver-oriented route setup mechanism can then resolve any conflicts between the computed routes as it installs each route.

2.3 Route Setup

Some routing protocols perform route setup and reservation setup simultaneously [4, 3, 13]. One of our goals is to offer route setup to any resource reservation protocol. In addition, we also want applications not using reservations to have access to these routing services. Moreover, we would like to offer reservation services over opportunistic routes, as well as those using route setup. This requires that we not embed route setup in the reservation establishment protocol itself, but rather incorporate it into the basic routing infrastructure.

We split route setup into several pieces. First, *alternate path setup* installs an alternate path for forwarding. The alternate path may specify a complete path between a source and destination, or it may specify only a portion of that path. Second, *route pinning* allows routing to choose to install "non-opportunistic" routes. If a route is pinned, it will remain fixed unless a failure occurs somewhere along the route. Upon failure, the route may be removed or degraded to an opportunistic route.

Most route setup protocols assume that every alternate path must also be pinned. However,

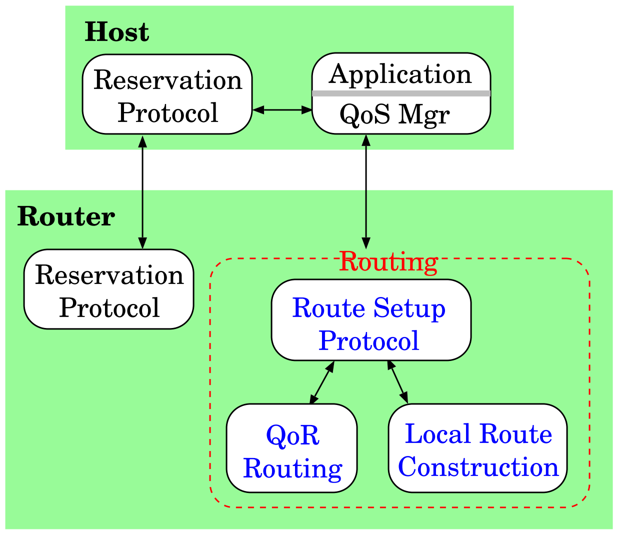


Figure 1: Routing and Reservation Architecture

in some cases an application may simply want to reach a destination by routing through ProviderA rather than ProviderB. In this case, routing may pin only the portion of the route through ProviderA. This will result in an alternate path of two pieces: an opportunistic path from the source to Provider A and then another opportunistic path from Provider A to the destination.

In addition, while route pinning may be used with alternate paths, it may also be used to convert a current opportunistic route into a pinned route. For example, in some cases an application may be using an opportunistic route and be satisfied with its service over this route. While some routes may remain stable for a long period of time [11], there is no guarantee that the route will remain unchanged. Particularly if the application has obtained a reservation along this route, it may want to convert the route to a pinned route. Having this capability allows the network to use current scalable routing protocols to support route pinning.

3 Using the Architecture

Figure 1 shows a simple block diagram of the routing architecture and its application interfaces. Applications have access to route setup and reservation setup through two separate interfaces. Note that there is no application interface to the local route construction agent; the application's first-hop router contacts the agent when it needs a route. By not allowing the application to determine the routes being used, we prevent a malicious or malfunctioning user from providing its own route and undermining the integrity or efficiency of a given tree.

In the most simplistic model of how this architecture would be used, applications would make requests for these services directly from routing or from RSVP. In fact, given the complexity of the service options available, we assume that many operating systems will offer some form of support for managing quality of service; such a *QoS manager* could act as an agent on behalf of an application, managing the reservation establishment and the routing services procurement process. A user (or a monitoring program) may simply indicate that service is unacceptable. The QoS manager could then choose from a number of actions, including asking for any of the available routing services, depending on the application. Thus, the QoS management function could reside either in the application itself, or in a QoS manager, or in some other form of operating system support. All of these possibilities fit within our proposed architecture.

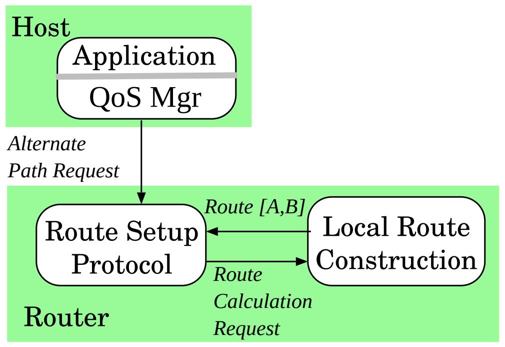
To illustrate the use of our routing architecture, we will discuss several examples, showing how applications may request alternate path setup and route pinning. In these examples, we focus on multicast routing. In order to perform scalable internet route setup for multicast groups with dynamic sets of receivers, we assume that the route setup protocol must use a receiver-oriented mechanism.

To initiate alternate path setup, an application or a QoS manager prompts routing for a different route, as shown in Figure 2. This signal may be prompted by many behaviors, including an admission control failure along the shortest path or a user's unhappiness with packet delays. The first-hop router then contacts the local route computation agent to get an alternate path, which returns an explicit (or source) route. The first-hop router embeds the explicit route in a Join message, and forwards the Join along the route, re-configuring the multicast tree at each hop (subject to constraints described later).

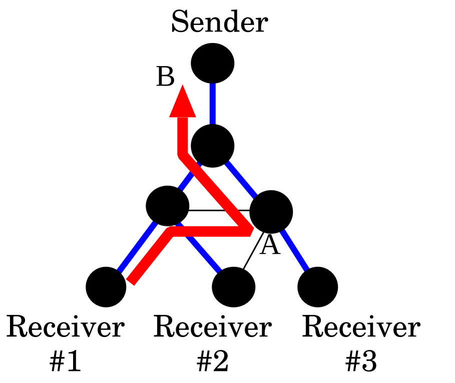
The QoS manager can request route pinning at the same time it requests an alternate path. Normally, when asked for an alternate path, the local route construction agent can return an explicit route that lists only the few hops necessary to reach an alternate route. However, when asked for route pinning as well, the local route agent returns a strict explicit route, which lists every hop between the first-hop router and the source of the multicast tree. Because the alternate path setup mechanism in effect pins each hop listed in the alternate path, a strict route necessarily implies that the entire route is pinned.

If an application is satisfied with its current opportunistic route, it can request route pinning alone, meaning that routing will convert its route into a non-opportunistic or pinned route. This situation can occur when the QoS manager has invoked RSVP over shortest path routing (perhaps according to a specific QoR request). As shown in Figure 3, the application or QoS manager prompts routing to pin the current route. The first-hop router then probes the multicast tree to determine the current route, and encodes this route as a strict explicit route. Note that the extra trip for probing the route allows routing to prevent loops during pinning by using an explicit route. The first-hop router finally embeds this route in a Join message, which it sends upstream to notify each router not to adapt unless its parent fails.

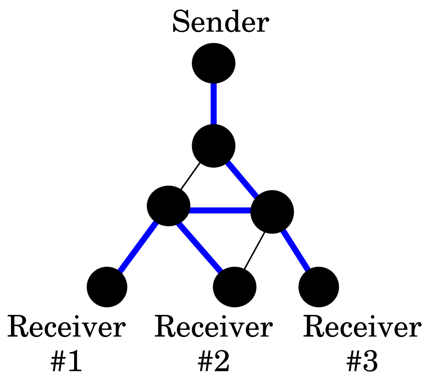
It may appear that an even simpler mechanism for route pinning would be for RSVP to



(a) Interactions between architecture components.

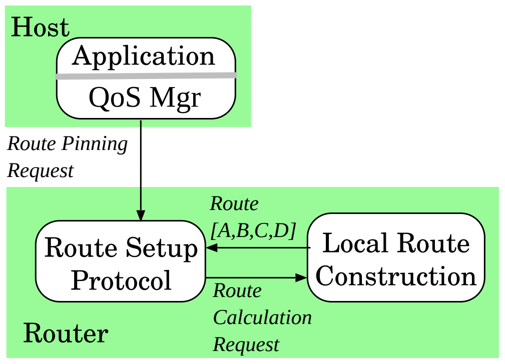


(b) First-hop router sends alternate path join for Receiver #1

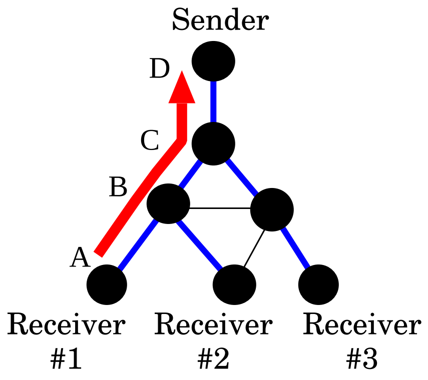


(c) Re-configured multicast tree

Figure 2: Alternate Path Joining Example



(a) Interactions between architecture components.



(b) First-hop router sends pinning join for Receiver #1

Figure 3: Route Pinning Example

notify routing, at each hop where it makes a reservation, that it needs that hop to be pinned [15]. However, this approach has several disadvantages. First, because RSVP is designed to transparently operate through non-RSVP routers, the portions of the route between RSVP routers would not be correctly configured. Moreover, because RSVP operates on top of opportunistic routing, the sequence of hops RSVP notifies for pinning might not, at any instant, form a coherent route. For example, during routing changes, RSVP leaves behind reservation state along the old route that eventually times out. The old (but not yet timed-out) reservations and the new reservations may not only lie along incompatible routes, but may even form a loop. The routing protocol would have a difficult time determining how to form a consistent, pinned route from the collection of reservations made over a set of opportunistic routes. This approach serves to illustrate how important it is that routing control the construction of routes. Requests for route pinning, issued by RSVP or the QoS manager, are limited to the endpoints of the network. The selection of routes is limited to the routing protocol. Note that while the mechanisms in [15] are designed for a link-state protocol, we believe that a hop-by-hop explicit join mechanism can be integrated into such a protocol. The protocol could still distribute link-state advertisements notifying other nodes about pinned hops, so that they could take this into account when constructing opportunistic portions of multicast trees.

Both alternate path setup and route pinning are also applicable to unicast routing. We have concentrated on multicast because the issues in that area are less straight-forward. For unicast applications, the unicast routing protocol can embed an explicit route in the source's packets if the application needs an alternate path or a pinned route. To save on processing overhead, routing could use an IPv6 flow label and some simple flow-setup protocol, with service degrading to regular hop-by-hop forwarding if flow state is lost. While the multicast model focuses on receiver actions, in the unicast case the source could control routing or use some out-of-band signalling to ask the destination about its capabilities.

4 Research Status

As a part of our research, we have designed interdomain route pinning and alternate path joining mechanisms for multicast routing. As shown in Section 3, these mechanisms are based on control messages that carry explicit routes. For an alternate path join, routing uses a loose explicit route; for route pinning it uses a strict route. By restricting the types of explicit routes used, the mechanisms are lightweight and provide loop freedom when re-configuring multicast trees by preventing even temporary loops. Further details of how alternate path joining and route pinning are implemented may be found in [14].

We are currently simulating these two routing services using a simple route construction prototype to examine the effects of these mechanisms on multicast trees. Our route construction prototype finds local access points to the network and constructs alternate paths by building a set of explicit routes listing only a single access point and the target (i.e. the source of the multicast tree). One of our goals is to determine whether local information is adequate to provide alternate paths; for large multicast groups this may produce good results. We are also interested in characterizing the "distortion" a multicast tree may exhibit when re-configured with an alternate path, and examine route construction techniques that limit this effect.

In this note we have focussed on two particular routing issues, failed primary route and opportunistic changes, that arise in integrated services networks. However, there are several other issues that arise, and addressing them may require some support from routing. Below we describe these issues, and possible routing mechanisms that could be used to address them. We are not proposing that these mechanisms be adopted. Our purpose is to evaluate these services and the role of routing in terms of their functional benefits and mechanistic complexity. It seems clear that for some, and perhaps all, of these issues, the benefits of the mechanism do not outweigh the cost of additional complexity. However, an informed decision about these services cannot be made without exploring the complexity of the routing support.

- The purpose of route pinning is to prevent service disruption. However, by sticking with the current route even though "better" ones have become available in the meantime means that routing pinning can result in flows using sub-optimal routes. One proposal to increase network efficiency is to introduce *smooth switching*, whereby routing gracefully transitions a flow from a reserved pinned route to a reserved opportunistic route. This would require routing to install a second route for a flow, allow RSVP to reserve that route, and then switch the forwarding to that route. This prevents the service disruption without requiring the flow to always stick with the route it first used. While this would be a beneficial service we currently doubt that this service can be implemented efficiently in multicast routing.
- Multicast applications involving a number of senders, for example a set of video sources, may want to rapidly switch between senders. RSVP has considered, but does not currently support, a *dynamic filter* style to allow a receiver to keep a reservation in place for each sender, thus facilitating the switch between senders. Without routing support, data from all sources would still be delivered to the receiver, but only the data associated with the selected source would be given the reserved service. To keep reserved but temporarily unneeded data from overwhelming the links close to the receiver (and thereby perhaps severely perturbing the best effort service there), routing could use a proposed service called *sender deactivation* to inactivate the forwarding state for a sender, but allow RSVP control traffic to still maintain a reservation for the sender. The modifications needed for this service appear to be a simple extension of current multicast routing protocols; however, it is still unclear whether this is a reservation style that RSVP needs to support explicitly. Receivers may be able to obtain adequate service simply by dynamically changing their explicit filters as a means of channel switching. Because of this uncertainty, it is premature to assume the need for this service.
- Finally, applications using hierarchical encoding of video may send the data for different encoding levels over separate multicast groups [7]. Some have proposed that

the multicast trees for each encoding level should use the same routes, thus allowing the network to use priority dropping during times of congestion. Routing could use a service called *bundling* to denote such a set of multicast groups. The routing protocol could then handle routing control messages uniformly for the bundled set to ensure that their trees use the same routes. On the other hand, bundling will generally happen by default for multicast trees using shortest path routes. It is unclear whether support for bundling is needed for alternate path joining and route pinning, or whether receivers could instead issue separate requests for each group.

We plan to continue investigation of these and other proposals and evaluate the appropriate role (or non-role) of routing in their support.

5 Acknowledgments

We wish to thank Lee Breslau for comments on an earlier draft, and the IETF community for helpful discussion of these issues.

References

[1] David D. Clark. “The Design Philosophy of the DARPA Internet Protocols”. In *ACM SIGCOMM*, August 1988.

[2] Stephen Deering, Deborah Estrin, Dino Farinacci, Van Jacobson, Ching-Gung Liu, and Liming Wei. An Architecture for Wide-Area Multicast Routing. In *ACM SIGCOMM*, August 1994.

[3] L. Delgrossi and L. Berger. “Internet Stream Protocol Version 2 (ST2): Protocol Specification Version ST2+”. RFC 1819, August 1995.

[4] Domenico Ferrari, Anindo Banerjea, and Hui Zhang. “Network support for multimedia: A Discussion of the Tenet Approach”. *Computer Networks and ISDN Systems*, 1994.

[5] C. Hedrick. “Routing Information Protocol”. RFC 1058, STD 0034, June 1988.

[6] Charles L. Hedrick. “An Introduction to IGRP”. Technical report, The State University of New Jersey, October 1989.

[7] Steven McCanne, Van Jacobson, and Martin Vetterli. “Receiver-driven Layered Multicast”. In *ACM SIGCOMM*, August 1996.

[8] D. L. Mills. “Exterior Gateway Protocol for Formal Specification”. RFC 904, April 1984.

[9] J. Moy. “OSPF Version 2”. RFC 1583, March 1994.

[10] International Standards Organization. “Protocol for the Exchange of Inter-Domain Routing Information among Intermediate Systems to Support Forwarding of ISO 8473 PDUs”. ISO/IEC JTC1/SC6 CD10747, 1993.

[11] Vern Paxson. “End-to-End Routing Behavior in the Internet”. In *ACM SIGCOMM*, October 1996.

[12] Y. Rekhter and T. Li. “A Border Gateway Protocol 4 (BGP-4)”. RFC 1654, July 1994.

[13] Burkhard Stiller. ”A Survey of UNI Signalling Systems and Protocols for ATM Networks”. *ACM SIGCOMM Computer Communication Review*, 25(2), April 1995.

[14] Daniel Zappala. “A Route Setup Mechanism For Interdomain Multicast Routing”. work in progress, March 1997.

[15] Zhang, Sanchez, Salkewicz, and Crawley. “Quality of Service Extensions to OSPF”. work in progress, June 1996.

[16] Lixia Zhang, Steve Deering, Deborah Estrin, Scott Shenker, and Daniel Zappala. “RSVP: A New Resource ReSerVation Protocol”. *IEEE Network*, September 1993.