Network Working Group tbd
Request for Comments: 14ZZ tbd
September 1993

Integrated Service in the Internet Architecture

***** DRAFT *****

September 28, 1993

Status of Memo

This memo provides information for the Internet community. It does not specify an Internet standard. Distribution of this memo is unlimited.

Abstract

This memo discusses an extension to the Internet architecture and protocols to provide "integrated service", i.e., to support real-time as well as best-effort IP service. This memo outlines a proposed technical framework and recommends steps towards engineering and deployment of Internet integrated service. This extension is necessary to meet the growing need for realtime service for multimedia applications.

1 Introduction

The Internet originally offered only a very simple quality of service (QoS), service model, point-to-point *best-effort* data delivery. This service model is not adequate for several new classes of distributed applications now under development, including remote video, multimedia conferencing, visualization, and virtual reality. These applications typically require a *real-time* QoS that provides some control over end-to-end packet delays. They also require IP multicasting to provide multi-destination delivery.

Over the past year, the IETF meeting broadcasts across the Internet have constituted a large-scale experiment in sending digitized voice and video through a packet-switched infrastructure. These highly-visible experiments depended upon three enabling technologies. (1) Many modern workstations now come equipped with built-in multimedia hardware, including audio codecs and video frame-grabbers, and the necessary video gear is now inexpensive. (2) IP multicast, which is not yet generally available in commercial routers, could be provided using a temporary "multicast backbone" known as the MBONE [ ]. (3) Highly-sophisticated digital audio and video applications were developed [VAT, etc.]

These experiments also illustrated that there is a missing technology; realtime applications do not work well across the Internet in general because of variable queueing delays and congestion losses. Before realtime applications such as remote seminars and teleconferencing can be broadly used, the Internet infrastructure must be modified to provide a realtime quality of service.

Realtime QoS manages network resources to benefit those Internet users who need realtime service. Network operators are requesting another new service, the ability to control the sharing of bandwidth on a particular link among different traffic classes. They want to be able to divide traffic into a few administrative classes and assign to each a minimum percentage of the link bandwidth under conditions of overload, while allowing "unused" bandwidth to be available at other times. These classes may represent different user groups or different protocol families, for example. Such a management facility is called *controlled link-sharing*. We use the term "integrated service" for an Internet service model that includes best-effort, realtime service, and controlled link sharing.

The requirements and mechanisms for integrated service have been the subjects of much discussion and research over the past several years [CSZ92,Floyd92,Jacobson91,JSCZ93,Partridge92,SCZ93,RSVP93]. This work has led to a unified approach to integrated service support that is described in this memo. It is now time to do the necessary engineering and deployment to extend the Internet architecture to provide integrated service.

Section 2 of this memo introduces the elements of an integrated service extension of the Internet. Section 3 discusses real-time service models. Section 4 discusses traffic control, the forwarding algorithms to be used in routers. Section 5 describes a resource setup protocol RSVP.

1.1 Resource Guarantees and Reservations

The essence of realtime service in a best-effort world is the requirement for some service guarantees. The term "guarantee" here is to be broadly interpreted; they may be absolute or statistical, strict or approximate. However, the user must be able to get a service whose quality is sufficiently predictable that the application will operate in an acceptable way over some duration of time determined by the user. Again, "sufficiently" and "acceptable" are vague terms. In general, stricter guarantees have a higher cost in resources that are made unavailable for sharing with others.

The guarantees of most importance to realtime applications will concern end-to-end packet delays. To control packet delays, we advocate extending the Internet architecture to include reservation of communication resources for particular end-to-end packet flows. This implies putting control state in the routers, which represents an important and fundamental change to the Internet model. The Internet architecture has been founded on the concept that all flow state should be in the end systems [DDC SIGCOMM]. This led to protocol robustness that is one of the keys to success of the TCP/IP protocol suite, and there is a natural reluctance to backing off from it. Changing the basic model in this manner needs to be justified.

The following arguments have been raised against resource reservation in the Internet.

1. Bandwidth will be infinite.

   The incredibly large carrying capacity of an optical fiber leads some to conclude that in the future bandwidth will be so cheap and ubiquitous that there will be no delays other than the speed of light, and therefore there will be no need to reserve resources. However, we believe that this is impossible in the short term and unlikely in the medium term. Even in the long term ,it seems very unlikely that the entire Internet will have such high capacity that reservation will be unnecessary anywhere.

2. Simple priority is sufficient.

   It is true that simply giving higher priority to realtime traffic would lead to adequate realtime service at some times and under some conditions. It would fail as soon as there are too many realtime streams competing for the higher priority. When demand exceeds available bandwidth, everyone's service will degrade unless some new requests receive "busy signal".

3. Applications can adapt.

   The development of adaptive realtime applications such as the VAT audio program [VJ] eases this problem, but does not solve it. There are limits to adaptation, set by the human requirements for interaction and intelligibility.

Based upon these considerations, we believe that an integrated service extension that includes additional flow state in routers and an explicit setup mechanism is necessary to provide the needed service. We do not believe that a partial solution short of this point would be a wise investment. We do believe that the extensions we propose preserve the essential robustness and efficiency of the Internet architecture. They also allow efficient management of the resources even if bandwidth is very inexpensive.

1.2 Requirements for Integrated Service

Our underlying assumption is that it is desirable to reshape the Internet into a common infrastructure to support both non-realtime and realtime communication. One could alternatively build an entirely new, parallel infrastructure for realtime services, leaving the Internet unchanged. This would lose the significant advantages of statistical sharing between realtime and BE traffic, and would be much more complex to build and administer than a common infrastructure.

We suggest the following requirements for an Internet IS architecture.

- Support Network Diversity

  The extension must obey one of the fundamental tenets of the Internet architecture: accomodate diverse network technologies in a highly heterogeneous network, under the control of many different administrations.

- Support Multicast

  The very popular IETF broadcasts have demonstrated a requirement for delivering the same realtime packet stream to a large number of receivers. IP multicasting will be essential for acceptable network efficiency.

  We believe that the IS extension must be designed from the beginning for multicasting; simply generalizing from the unicast (point-to-point) case does not work. We therefore design the IS architecture for multi-destination delivery, considering unicast as a limiting case.

- Backward Compatibility and Interoperation

  For compatibility, all Internet paths and hosts must continue to support the current minimal best-effort service (BES) as the universal interconnection protocol, with realtime service (RTS) added as an additional facility.

The IS architecture must be adaptable to a wide latitude of conformance. In particular, we should expect only partial coverage of the Internet by the real-time extensions. It will take many years to upgrade the entire Internet. In addition, some parts of the Internet, e.g., local Ethernets, may never be upgraded to support the full integrated service. However, excess capacity may still allow adequate support for real-time traffic across segments that have not been upgraded. It must be possible to provide (perhaps degraded) realtime service across paths including such non-conforming segments.

- Feasibility

  The extension must be feasible in an engineering sense. For example, it must be reasonably implementable in the hardware assists used for packet forwarding in commercial routers. The extensions must not require re-engineering and re-engineering of many existing protocols. In particular, they should not place an additional burden on the current generation of routing protocols, whose complexity is approaching the limit of feasible implementation and debugging (although future generations of routing protocols may incrementally improve operation of the extensions).

- Flexibility and Extensibility

  The IS architecture must provide flexibility for future extensions and for adaptation to shifting requirements.

  We know very little about the future requirements of realtime application. For example, current video codecs provide variable picture quality in order to operate at a constant bit rate; future codecs may instead provide constant picture quality by creating data at a variable rate. In addition, future realtime applications are likelyt to be capable of some adaptation to delay variations. We can only conjecture how much variable-rate operation and how much adaptation will occur in practice.

- Multiple Protocol Stacks

  The Internet supports multiple protocol stacks, and an IS must be potentially applicable to all of them. Of particular importance are the present IPv4 protocol in the TCP/IP stack, the OSI equivalent CLNP protocol, and any next-generation IP.

- Advance Reservations

  Some parts of the Internet will have not have sufficient capacity to support the potential real-time load. Since it will not always be possible to establish a teleconference on demand, advance reservation will be required to provide an adequate service. The architecture must therefore provide for advance reservation of capacity and for the preemption that is implied.

- Accounting

  Realtime service will involve preferential use of network resources. It will be necessary to maintain usage records in order to ensure appropriate use of this preferred service.

2 ELEMENTS OF THE ARCHITECTURE

We define the "flow" abstraction as a distinguishable stream of related IP datagrams that result from a single user activity and require the same QoS. For example, a flow might consist of one

transport connection or one video stream between a give host pair. It is the finest granularity of packet stream distinguishable by the IS. We define a flow to be simplex, i.e., to have a single source but N destinations. Thus, an N-way teleconference will generally require N flows, one originating at each site.

2.1 Components

An "architecture" should represent a unified whole. However, it is useful to try to decompose a complex problem into roughly orthogonal issues. We find it useful to identify five components to the IS architecture: the real-time service model, the packet scheduler algorithm, the admission control algorithm, the classifier, and the reservation setup protocol.

1. Integrated Service Model

   It is important to have an abstract model of the service qualities to be supported, independent of any particular mechanism used to realize that service.

   In particular, the service model includes the parameter list or "flow spec" used by a host to request real-time service [Partridge92]. The model must also consider the role of cost and/or accounting for the scarce resources being consumed.

   We are proposing a particular integrated service model that includes guaranteed and predictive service with controlled link-sharing ((XXX Is this too strong a claim at this point? DE)). This model is discussed in Section 3.

2. Packet Scheduler

   In today's Internet, IP forwarding is completely egalitarian; all packets receive the same quality of service, and packets are typically forwarded using a strict FIFO queueing discipline. For integrated service, a router must implement an appropriate QoS for each flow, in accordance with the service model. The router function that creates different qualities of service is called "Traffic control". Traffic control in turn is implemented by three components: the packet scheduler, the classifier, and admission control.

   The packet scheduler manages the forwarding of different packet streams using a set of queues and perhaps other mechanisms like timers. The packet scheduler must be implemented at the point where packets are queued; this is the output driver level of a typical operating system, and corresponds to the link layer protocol. The details of the scheduling algorithm may be specific to the particular output medium. For example, the output driver will need to invoke the appropriate link-layer controls when interfacing to a network technology that has an internal bandwidth allocation mechanism.

   We have built a scheduler to implement the CSZ IS model. This is discussed further in Section 4.

3. Admission Control

   Admission control implements the decision algorithm that a router or host uses to determine whether a requested service increment can be granted without impacting the earlier guarantees. The admission control algorithm must be consistent with the service model, and it is logically part of traffic control.

Admission control is a decision made at the time a host requests a realtime service logical channel across the Internet. It is sometimes confused with *policing* or *enforcement*, which is a packet-by-packet function at the "edge" of the network to ensure that a host does not violate its promised traffic characterisics. We consider policing to be one of the functions of the packet scheduler.

Although there are still open research issues in admission control, a first cut exists [JCSZ93].

4. Classifier

   The basic unit of traffic control (and accounting) is a *class*. A router must classify each incoming packet, i.e., map it into some class, in order to know when (and perhaps whether) to forward it. The packet forwarding process treats all packets in the same class equally. Classification may be based upon either the contents of the existing packet header(s) and/or some additional classification number added to each packet, e.g., in an IP option. In any case, classification will require new state information in a router.

   A class might correspond to a broad category of flows, e.g., all video flows or all flows attributable to a particular government agency. On the other hand, a class might hold only a single flow. A class is an abstraction that may be local to a particular router; the same packet may be classified differently by different routers along the path. For example, backbone routers may choose to map many flows into large aggregate classes, while routers nearer the periphery, where there is much less aggregation, may use a separate class for each flow. One motivation for this is to control the overhead of classification, as described later.

   The Classifier selects an appropriate QoS class for each packet to be forwarded. In addition, a route must be selected for the packet, in the conventional manner. The route implies which output driver will be used, and the packet is passed to this driver with the route and the QoS class.

   In principle, we can allow a very general definition of the header fields used in classification. Within an IP header, for example, the source and destination IP addresses, the protocol field, and the TOS bits are obvious candidates. In addition, the Classifier may be allowed to look into transport protocol fields, depending upon the IP protocol field. Thus, video streams might be recognized by the use of a particular well-known port field in the UDP header, or a particular flow might be recognized by looking at both the source and destination port numbers.

   A Classifier must be both general and efficient. One way to gain efficiency is to combine routing decisions and classification to the extent possible.

5. Reservation Setup Protocol

   It will typically be necessary to have some state specific to a flow both in the endpoint hosts and in routers along the path of that flow. The host and router state required for a flow will be created and maintained by a reservation setup protocol. Advance reservation also falls into this area.

   There are a number of requirements on a reservation setup protocol. It must be fundamentally designed for a multicast environment. It must give flexible control over the manner in which reservations can be shared along braches of the multicast delivery trees. It should be designed around the elementary action of adding one sender and/or receiver to an existing set, or deleting one.

We propose a reservation setup protocol called RSVP (for "ReSerVation Protocol") to meet these requirements.

Figure 1 shows how these components fit into an IP router that has been extended to provide integrated service. The router has two broad functional divisions: the forwarding path below the double horizontal line, and the background code above the line.

The forwarding path of the router is executed for every packet, and hence must be highly optimized. Indeed, in many commercial routers, its implementation involves a hardware assist. The forwarding path is divided into three sections: input driver, Internet forwarder, and output driver.

The Internet forwarder interprets the internetworking protocol header appropriate to the protocol suite: the IP header for TCP/IP, or the CLNP header for OSI. For each packet, a forwarder executes a suite-dependent Classifier and then passes the packet and its class to the appropriate output driver. The output driver implements the packet scheduler and admission control. A router that supports multiple protocol suites may have multiple different Internet Forwarder modules, e.g., for IP, CLNP, and/or IPng. Each will have a Classifier and a routing function appropriate to its internet-layer (called "network layer" in OSI) protocol.

The background code is simply loaded into router memory and executed by a general-purpose CPU. These background routines create databases that control the forwarding path.

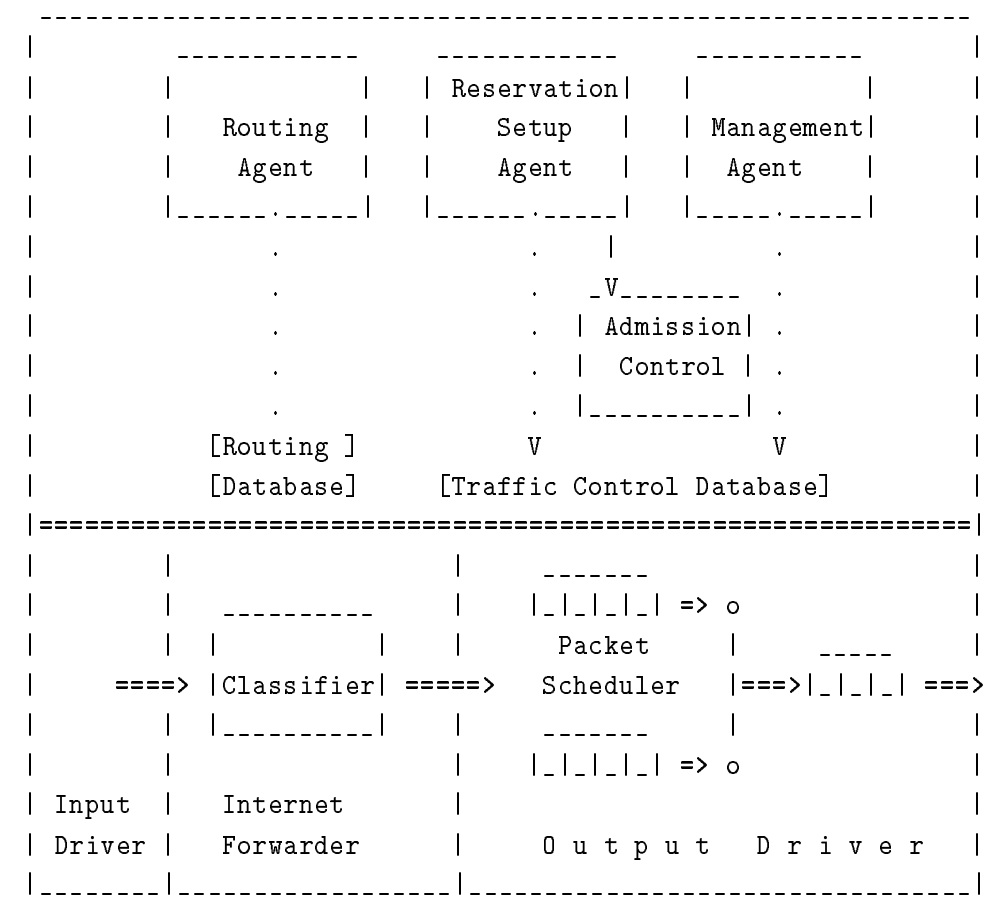


Figure 1. Router Model

The Routing Agent implements a particular routing protocol and builds a routing database. The Resource Control Agent implements the resource control protocol (i.e., RSVP) and makes local reservation requests. If Admission Control gives the "OK" for a new request, the appropriate changes are made to the Classifier and Packet Scheduler control databases to implement the desired reservation.

Finally, every router supports an agent for network management. This agent must be able to modify the Classifier and Packet Scheduler databases to set up controlled link-sharing and to set Admission Control policies.

The model of a host is generally similar to that for a router, with the addition of applications. Rather than being forwarded, host data originates and terminates in an application. Realtime applications use an API to a local Resource Control Agent to request the desired QoS for a flow. The IP output routine of a host may need no Classifier, since the class assignment for a packet can be specified in the local I/O control structure corresponding to the flow.

In routers, integrated service will require changes to both the forwarding path and the background functions. The forwarding path of a modern router often depends upon hardware acceleration for performance, and will therefore be relatively difficult and costly to change. It will be vital to choose a router forwarding mechanism that is general and adaptable to a wide variety of policy requirements and future circumstances, and that can be implemented efficiently.

2.2 Unified Protocol Stack

The IS architecture discussed in this memo employs a single unified internet-layer protocol for both realtime and best-effort service. Thus, it uses the existing IP or CLNP protocol for realtime data. Another approach would be to add a new realtime protocol in parallel with the existing BE internet protocol [Topolcic90]. Our unified stack approach is favored by the requirement for handling partial coverage, i.e., to allow easy interoperation between IS-capable Internet systems and systems that have not been extended, without the complexity of tunnelling. It also provides economy of mechanism, and allows us to fold controlled link-sharing in easily.

2.3 On Diversity

One possible requirement on the IS architecture is still open to question: to what extent should it explicitly allow for diversity of the protocols and mechanisms for realtime service?

We take the view that there should be a single service model and a single reservation setup protocol for the Internet, but not necesssarily a single packet scheduling mechanism. Although specific packet scheduling and admission control mechanisms that satisfy our service model have been developed, it is quite possible that other mechanisms will also satisfy the service model.

However, if there were different service models in different parts of the Internet, it is very difficult to see how any end-to-end service quality statements could be made, and this would defeat the purpose of the integrated service extension. Perhaps it is not possible to make such a strong statement about multiple setup protocols, but it is clearly desirable for uniformity to have only one. Finally, notice that evolutionary change may be expected to introduce diversity even if there is only a unified service requirement.

3 REAL-TIME SERVICE MODEL

Scott has agreed to write this section.

4 TRAFFIC CONTROL

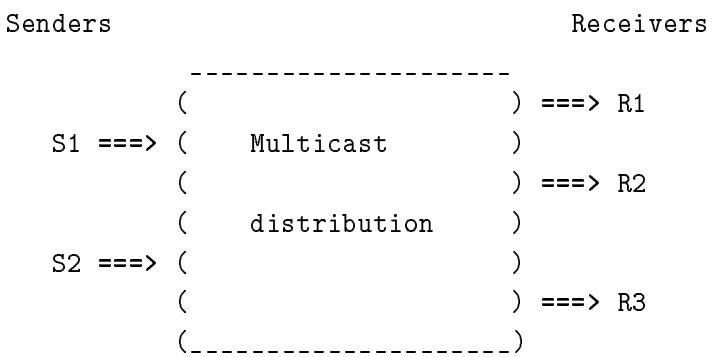
Can I persuade DDC to write this section?

5 RESERVATION SETUP PROTOCOL

5.1 Sessions

Figure 3 shows our basic model for multi-destination data distribution for a shared, distributed application. The arrows indicate data flow from senders S1 and S2 to receivers R1, R2, and R3, and the cloud represents the distribution mesh created by the multicast routing protocol. Multicasting distribution replicates each data packet from a sender Si, for delivery to every receiver Rj (whether a packet actually arrives at Rj depends on the specified QoS and perhaps upon congestion encountered along the path). We call this multicast distribution mesh an *M-session*.

flow



Multicast Distribution Session (M-session)

Figure 3.

The reservation setup protocol is used by hosts and routers to create, modify, and delete resource reservations for individual M-sessions, to support realtime applications. However, an M-session may equally well carry elastic traffic with no realtime guarantees; resource reservations are an added feature.

There are to different possible styles for reservation setup protocols, the "connection-oriented" (CO) or "hard state" approach, and the "connectionless" (CL) or "soft state" approach.

- CO Setup

  Under the CO approach, multicast distribution is performed using connection-like state in each router along the path. This state is created and deleted in a fully deterministic manner by cooperation among the routers. Once a host requests a session, the "network" takes responsibility for creating and later destroying the necessary state. ST-II is a good example of the CO approach [Topolcic90].

  Since management of CO session state is completely deterministic, the CO setup protocol must be reliable, with acknowledgments and retransmissions. In order to achieve deterministic cleanup of state after a failure, there must be some mechanism to detect failures, i.e., an "up/down" protocol. The router upstream (towards the source) from a failure takes responsibility for rebuilding the necessary state on the router(s) along an alternate route.

- CL Setup

  In the CL approach, reservation state in the routers is regarded as cached information that is installed and periodically refreshed by the end hosts; unused state is timed out by the routers. If the route changes, the refresh messages automatically install the necessary state along the new route.

The CL approach was chosen as the basis of the Internet protocol architecture to obtain simplicity and robustness against failures and errors [DDC SIGCOMM]. We believe that these same advantages will result from choosing a CL approach to the reservation setup protocol RSVP.

Another design issue concerns the roles of senders and receivers in the reservation setup. A sender knows the qualities of the traffic stream it can send, while a receiver knows what it wants to (or can) receive. We want to allow heterogeneous sender and receiver streams, so the distributed computation of resource reservations could require a complex many-sided negotiation between senders and receivers. The design question is how much of this negotiation should be performed by some higher-level application protocol, and how much be the resource setup protocol itself, and how much generality should be accomodated.

One common approach to performing the negotiation in the reservation protocol is the following *two-pass* scheme. An "offered" flowspec is propagated along the multicast distribution tree from each sender Si to all receivers Rj. Each router along the path would record these values and perhaps adjust them to reflect available capacity. The receivers would get these offers, generate corresponding "requested" flowspecs, and propagate them back along the same routes to the senders. At each node, a local reconciliation would then be performed between the offered and the requested flowspec, to create a reservation, and an appropriately modified requested flowspec would be passed

on. This two-pass scheme allows extensive properties like allowed delay to be distributed across hops in the path [Tenet, ST-II].

RSVP uses an even simpler approach, a one-pass setup mechanism in which reservervations are receiver-initiated. A receiver is assumed to learn the senders' offered flowspecs by a higer-level mechanism ("out of band"). The receivers then generate and propagate request flowspecs towards the senders, making reservations in each router along the way. This approach is justified by the observation that in practice most of the queueing delay will not be evenly distributed but will occur at one or a few bottleneck nodes. Furthermore, we do not think it will often be useful (or perhaps possible) to achieve great precision in resource guarantees.

As we noted above, we favor a common datagram delivery mechanism for both elastic and realtime traffic. This means that realtime traffic will use IP multicasting when multi-destination delivery is required. In order to scale well to large groups, IP multicasting addresses datagrams to logical addresses that implicitly name the destination hosts. Any host may send to a group, but they must explicitly ask to join a group in order to receive its packets. This receiver initiation of group membership is consistent with the, receiver initiation of reservations by RSVP.

5.2 Routing vs. Resource Management

There is a fundamental conflict between dynamic routing and the necessity to bind resource reservations to the nodes along a particular route. We could force static routing for real-time traffic [Anderson90a], or we could rebuild the necessary session state on the alternate path when rerouting does occur [Topolcic90]. Static routes for real-time traffic are unacceptable, since they prevent recovery from failures of lines or routers. The ability of the Internet level to bypass link-layer failures is a fundamental property of the Internet architecture that must be retained for integrated service.

A different relationship between routing and resource management occurs when a session is set up: the optimal choice of route may depend upon the resources available along the possible paths. Thus, we might add resources to the attributes of a link for the purposes of link-state computation. The available resource levels would be broadcast to all routers, and all would do an identical resource computation to determine the route. Note that we would base this upon the protected resource levels, which are relatively static, rather than upon the actual utilization, whose dynamics could easily lead to route oscillations, etc.

We propose that this general approach NOT be used. Routing protocols are already reaching the threshold of feasible complexity, and we do not want to add a significant new burden. Instead, we propose that the routing computation find a route based only upon connectivity and independent of resource levels, and that this predetermined route be used for session setup. We therefore propose to keep the reservation setup protocol distinct from, but dependent upon, the underlying uni-/multicast routing protocol. This same routing protocol will be used for forwarding elastic as well as for realtime traffic.

This simplification may occasionally lead to failure to create the best, or even any, realtime session. Fortunately, new routing paradigms still undergoing research may eventually provide a way to find an alternate route when the desired resources are not available on the primary route. Thus, we

advocate an evolutionary approach to solving the problem of routing with resource reservation.

((XXX DE: say something about sticky routes))

Even with the current-generation routing, we believe that a complete failure will be rare; in most cases, the default route will be the highest capacity route and therefore the most appropriate choice for real-time traffic.

5.3 RSVP

In general, an RSVP reservation specifies the amount of resources to be reserved for all, or some subset of, the packets in a particular session. The resource quantity is specified by a *flowspec*, which parametrizes the packet scheduling mechanism. The packet subset to receive those resources is specified by a *filter spec*. A filter spec defines a packet filter that is instantiated in the Classifier.

The RSVP protocol mechanisms provide a very general facility for creating and maintaining distributed reservation state across the mesh of multicast delivery paths. These mechanisms treat flowspecs and filter specs as opaque binary data, simply handing them to the local Admission Control module for interpretation. However, the service model presented to an application must specify how to encode flowspecs and filter specs.

Each receiver host sends RSVP reservation (*Resv*) messages into the Internet, carrying flowspecs and filterspecs requesting the desired QoS. These reservation messages follow the reverse routes of the data packets from all the senders in which the receiver is interested, and are finally delivered to the sender hosts. This allows the senders to set up appropriate Traffic Control parameters to provide the desired QoS on the first hop.

Each sender transmits RSVP *Path* messages forward along the uni-/multicast routes provided by the routing protocol(s). These *Path* messages store *path state* in all the intermediate routers, effectively combining all the routing trees given by the routing protocol for the same DestAddress. The path state is used to route the *Resv* messages sent by the receivers, hop-by-hop along the reverse path(s) to the senders.

5.4 Resource Aggregation

6 ACKNOWLEDGMENTS

The work reported in this memo was initially discussed and organized in the End-to-End Research Group of the Internet Research Taskforce. Many Internet researchers have contributed, especially Steve Casner, Dave Clark, Steve Deering, Deborah Estrin, Sally Floyd, Van Jacobson, Craig Partridge, Scott Shenker, and Lixia Zhang.

References

[ISIP91] Braden, R., *Integrated Service in the Internet Architecture*, High-Performance Network Research Report, October 1991.

[CSZ92] Clark, D., Shenker, S., and L Zhang, *Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanisms*, Proc. SIGCOMM '92, Baltimore, MD, August 1992.

[SCZ93] Shenker, S., Clark, D., and L. Zhang, *A Scheduling Service Model and a Scheduling Architecture for an Integrated Services Packet Network*, submitted to ACM/IEEE Trans. on Networking.

[Floyd92] Floyd, S., *Issues in Flexible Resource Management for Datagram Networks*, Proceedings of the 3rd Workshop on Very High Speed Networks, March 1992.

[Jacobson91] Jacobson, V., *Private Communication*, 1991.

[JSZC93] Jamin, S., Shenker, S., Zhang, L., and D. Clark, *An Admission Control Algorithm for Predictive Real-Time Service*, Extended abstract, in Proc. Third International Workshop on Network and Operating System Support for Digital Audio and Video, San Diego, CA, Nov. 1992, pp. 73-91.

[Partridge92] Partridge, C., *A Proposed Flow Specification*, RFC-1363, July 1992.

[RSVP93] Zhang, L., Deering, S., Estrin, D., Shenker, S., and D. Zappala, *RSVP: A New Resource ReSerVation Protocol*, Accepted for publication in IEEE Network, 1993.

[ST2-90] Topolcic, C., *Experimental Internet Stream Protocol: Version 2 (ST-II)*, RFC-1190, October 1990.

[tenet]