A NEW APPROACH FOR PROVIDING QUALITY-OF-SERVICE IN A DYNAMIC NETWORK ENVIRONMENT

M. Mirhakkak
N. Schult
D. Thomson

The MITRE Corporation
Network and Communications Engineering Department
Reston, Virginia

ABSTRACT

This paper looks at issues involved in attempting to provide Quality of Service (QoS) support in a dynamic environment. We focus on a resource reservation based approach, which we believe is attractive for military applications, but becomes especially difficult in a dynamic network environment. This is because available resources reserved for a particular flow may contract after they have been "committed" to the flow, causing the reservation to be dropped. Our approach is to expand the semantics of the reservation, so that instead of being a single value indicating the level of service needed by an application, it becomes a range of service levels in which the application can operate, together with the current reserved value within that range. This provides the network flexibility so that reservations can be maintained as network conditions change. Rather than forcing the network to make a binary "admit/fail" decision for each flow, the network provides feedback to applications on the current reservation level. Based on this feedback, applications can adapt their behavior to what the network can support. We have developed a prototype implementation of this concept, running as an extension to the Reservation Setup Protocol (RSVP) protocol. We are currently evaluating the implementation in a testbed network where we can vary link bandwidth. The testbed also includes several adaptive applications (audio, video, data transfer) running over the User Datagram Protocol (UDP). The paper discusses our approach, testbed, experiences to date, and current plans.

1. INTRODUCTION

Quality-of-Service (QoS) support in a network implies that the network is capable of providing different services that are appropriate for different applications or users. Two complementary approaches have been proposed for providing this support within the network layer. One is based on reserving resources for individual data flows end-to-end over heterogeneous links, based on signaling by the applications (an Integrated Services or "intserv" approach [1]). The Resource Reservation Setup Protocol (RSVP) is a signaling protocol typically associated with this approach that applications can use to request resources from the network [2]. It conveys information to nodes along the path between flow endpoints, so that the nodes may provide the resources and services the flows need to meet their specification. The second approach is based on the treatment of individual IP data packets at a node according to how the packets are marked in the IP header (a Differentiated Service or "diffserv" approach [3]). When an IP packet arrives at a node, it is classified into one of a (small) number of classes, based on the value of the Differentiated Services Code Point (DSCP) in its IP header, and then subject to a Per Hop Behavior (PHB) associated with that code point.

Both approaches have been implemented and are being studied in relatively traditional and static environments, where nodes rarely move and are connected by wired links. One concern mentioned with the intserv approach is scalability, since per flow state and processing is required at each node to support end-to-end QoS per flow. A primary benefit of diffserv is scaling, since it eliminates per flow state and processing. However, intserv provides an end-to-end QoS solution, which diffserv does not (currently).

A dynamic network environment, with potentially moving nodes connected by wireless links, presents additional challenges to providing QoS support in the network. These network dynamics can be attributed to variable link characteristics, node movement, and/or variable application demand. We believe that QoS support should adapt to these changes. In the next section we present a brief discussion of these dynamics and some proposed methods to address them. In section 3, we describe our approach for providing QoS in a dynamic network environment, which is reservation-based and falls within the intserv framework. We have implemented this approach as an extension to the RSVP. In section 4, we briefly discuss our testbed, adaptive applications, and

evaluation plans. Finally, section 5 presents some final comments.

2. DYNAMIC NETWORK ENVIRONMENT

In static networks, traditionally-wired links generally have a stable transmission quality. In contrast, wireless links are subject to variations in transmission quality due to factors such as interference and fading. (In military networks, the possibility of jamming must also be considered.). There are different ways to deal with these problems at different layers of the protocol stack. Assuming that changes in transmission quality are not handled by the physical layer (for example by increasing transmit power), the result observed by the link layer will be changes in the bit error rate. The effect observed by the network layer depends on whether the link layer mechanisms in use recognize and respond to these changes.

If the link layer does not detect or respond to changes in bit error rate, the network layer sees an increase in lost or corrupted packets. This makes it difficult to apply network layer QoS mechanisms, which are designed to deal mainly with congestion loss and network layer queueing effects, rather than packet loss due to link errors. Therefore we believe that it is best to deal with transmission dynamics within the link layer.

The link layer can react to the change in transmission quality in several ways. For example, if the link layer protocol includes automatic repeat-request (ARQ), then, as transmission quality decreases, the number of retransmissions will increase. The main affect on the network layer will be a decrease in the effective throughput of the link. A sophisticated link layer could also employ an adaptive error correction mechanism to increase or decrease the amount of error correction coding in response to changes in transmission quality. The affect on the network layer will be variation in effective delay or throughput, depending on the coding algorithm. The link layer could also react by changing its modulation technique, which again would be observed at the network layer as a change in effective bandwidth.

Thus, assuming that the link layer is responding to changes in transmission quality, the network layer will receive varying link throughput. Therefore, in order to appropriately perform admission control, allocate resources, and other functions necessary for providing QoS, the network layer needs updated information from the link layer on the effective data rate of each interface, as well as possibly other parameters such as latency. An example of how this can be done is found in the DARPA funded Global Mobile (GloMo) effort, for which a general framework for internet devices operating over wireless links was developed. This framework [4] defines interfaces between different layers or components, and includes the ability of the network layer to obtain information on current conditions observed by the link layer, including current link speed. In addition to providing current conditions to the network layer, the link layer should also include QoS aware channel access, to make sure that the QoS requirements for each node of a given channel are fulfilled [5].

Another source of variation in dynamic networks is node movement, which has several consequences. First, it exacerbates the problem of variable link characteristics, as nodes move in and out of areas of good signal strength. Second, nodes may have to switch to different media as they move in and out of coverage. A “vertical handoff” approach has been described [6] in which seamless connectivity to mobile nodes is maintained by handing off between small cells with high bandwidth and wide area cells with lower bandwidth. Again, this illustrates the need for QoS mechanisms to deal with variable bandwidth.

Node movement also means that the network topology will likely change. In the simple case, this consists of the movement of end systems through a fixed network infrastructure. Mobile end systems are “handed off” between fixed access points. However, in a more general case of a mobile ad hoc network, intermediate systems (routers) also move, possibly causing relatively rapid routing changes. Clearly, this has a major impact on a resource reservation-based QoS approach, as resources that were available and reserved to support a reservation along one route may not be available on the new route. Various approaches have proposed using “pre-reservation” (in the simpler “handoff” case), or “standby routes” (in the more complex mobile ad hoc network case) to allow the network to make a QoS commitment that can be honored even when movement (and topology change) occurs (e.g., [7], [8], [9], [10]). The notion of treating resource reservations as ranges and adaptively adjusting QoS within this range has also been put forth to accommodate fluctuations in available network bandwidth (e.g., [9], [11], [12]). Still others have investigated how to support QoS routing in the mobile ad-hoc network environment (e.g., [13]).

Finally, another source of dynamics is one that is common to both dynamic and fixed environments. That is: variable demand on network resources by end-system applications. The conventional response to variable application demand is based on admission control. That is, some end users

may be denied access in order to preserve QoS for those that have been admitted. This results in some users being granted the requested QoS, and others being denied service or forced to use a lower grade service-model, for example reverting to best-effort. Admission control may be performed on a "first come, first served" basis, or it may include priority and preemption mechanisms, to implement some desired policy.

Admission control is a valuable tool for dealing with variable application demand for QoS, because it is generally better to deny service to some users in order to grant service to others, rather than providing an unacceptable QoS to all users. The limitation with this approach, however, is that it traditionally provides an "all or nothing" service, which may be less than optimal. For example, in many cases it may be desirable to provide a reduced level of QoS to a larger number of users. In order to do this, the network and applications need some way to communicate on what levels of QoS would be acceptable and can currently be supported.

3. DYNAMIC QOS APPROACH

Our approach, which we call "Dynamic QoS", is resource reservation based, similar to that associated with the intserv architecture, except that we use an expanded notion of the meaning of a "reservation". With "Dynamic QoS," a resource reservation request specifies a range of values, and the network makes a commitment to provide service at a specified point within this range. Applications request QoS by specifying the minimum level of service they are willing to accept and the maximum level of service they are able to utilize, and then adapt to the specified point within this range that the network commits to provide, which may change with time. Changes in allocation are signaled to the application, which adapts its behavior to match what is available. If the allocation drops below the minimum level of service specified, the reservation is dropped. Treating reservations as ranges, and providing a mechanism for the network to signal the current allocation within the range, provides the flexibility needed for operation in a dynamic environment in which network conditions are changing.

There are several points to note about our approach. One is that we assume the link layer deals with errors, can provide information on resulting effective link bandwidth, and can provide QoS support in a shared media network environment. Secondly, our QoS approach is decoupled from routing. We assume access to a forwarding table, but do not make any assumptions on what routing protocol is building this table. Last, our approach uses "soft-state" to establish QoS along the new route when a change occurs. (This decoupling was also made in another approach [11]).

The Dynamic QoS approach can be divided into three parts: 1) additions to RSVP to support reservation ranges and bottleneck link discovery, 2) the bandwidth allocation algorithm, and 3) the application interface. These are each briefly discussed below.

3.1 Dynamic RSVP Protocol (dRSVP)

We made several additions to the standard RSVP protocol to support a reservation range. We also made some additions to support bottleneck link discovery for a given flow, in order to determine the bandwidth available on a flow's end-to-end path[1]. We refer to the resulting protocol as "Dynamic RSVP" (dRSVP). These additions are listed below, with the assumption that the reader is familiar with the basic structure and functionality of RSVP [2]. Message processing rules were also added and/or modified to deal with these additions (but are not discussed here).

- To describe ranges of traffic flows, an additional flow specification (flowspec) was added to Resv messages, and an additional traffic specification (tspec) was added to Path messages.
- To allow nodes to learn about "downstream" resource bottlenecks, we added a "measurement specification" (mspec) to the Resv messages.
- To allow nodes to learn about "upstream" resource bottlenecks, we created a new reservation notification (ResvNotify) message, which carries a "sender measurement specification" (smspec) information.
- Admission control processing was changed to deal with bandwidth ranges.
- We added a bandwidth allocation algorithm that divides up available bandwidth among admitted flows, taking into account the desired range for each flow as well as any upstream or downstream bottlenecks for each flow.
- We extended the API to deal with bandwidth ranges.

The process of discovering the bottleneck link on a flow's path is illustrated in figure 1 for a simple network in which node S sends data to node R through intermediate nodes N_1, N_2, N_3, and N_4. The nodes are connected by links, shown in the figure as wide bars, with the width of the bar

[1] Here, we define a flow to be a user packet stream resulting from a single user activity that has the same QoS requirements [1].

corresponding to the bandwidth available on the link. The adaptive application running on node S can generate data at rates within the range from s_l to s_h. These values are communicated in Path messages, which flow through the network hop by hop, following the same route as the data messages, to the receiver R. Upon receipt of the Path messages, the receiving application on R requests a reservation for this flow, with QoS range (s_l, s_h). The request is carried through the network in Resv messages, which is the reverse of the route followed by the Path messages (assuming bi-directional, symmetric links).

Each node, upon receiving the Resv message, performs an admission control check and computes the bandwidth, within the range (s_l, s_h), that it can allocate to the flow. Assuming the admission control test passes, the Resv message is propagated upstream towards S. The Resv message also contains a "receiver measurement specification" value, denoted m_r. The value of m_r is initialized at R to s_h, but as each node propagates the Resv message upstream, if the bandwidth allocation that it is able to give the flow is less than the received m_r value, it reduces the m_r value to the allocation. The value of m_r received by each node informs it of any downstream bottlenecks. S applies similar logic to the Application Program Interface (API), treating as if it were an upstream link. In our example, the application receives allocation a, as shown, indicating the bandwidth that has been reserved end-to-end through the network. The application must adapt to a in order to receive the agreed-upon service (controlled-load, in our implementation). If the reservation successfully propagates all the way through the network, node S initiates a ResvNotify message. This contains a "sender measurement specification", denoted m_s. This propagates toward R, and in a similar fashion each node limits the value of m_s to the bandwidth that it is able to allocate for the flow. The value of m_s received by each node informs it of any upstream bottlenecks. In our example, node N_2 learns that there is a downstream bottleneck of width a_{dn} and an upstream bottleneck of width a_{up}, as shown in the figure.

The dRSVP protocol supports multicast flows and, as a result, aggregating information (i.e., flowspecs, m_r, m_s) for multiple receivers and/or senders needs to done. Processing rules have been defined to support different reservation styles, but are not discussed here (due to space considerations).

3.2 Bandwidth Allocation Algorithm

When a node receives Resv messages for a new flow, it must determine whether to accept the given flow, and what allocation to assign to the flow. This section provides a

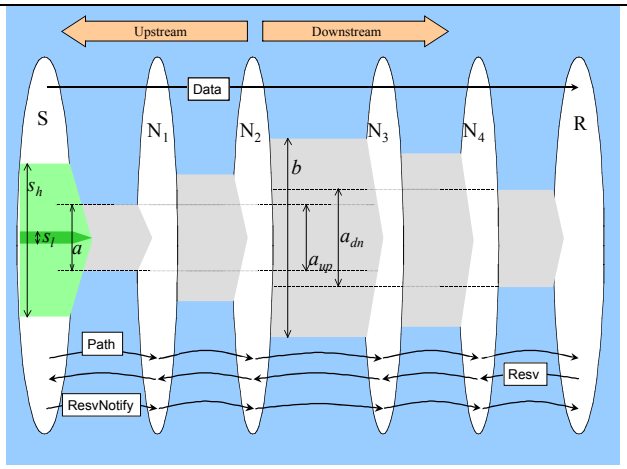

Figure 1. Overview of dRSVP

high-level overview of the bandwidth allocation algorithm that is executed by each QoS-capable node in a flow's path. The goal of the algorithm is to determine how much bandwidth on each downstream interface to allocate to each flow, given knowledge of the upstream and downstream bottleneck bandwidth values per flow (which are signaled by dRSVP). We have chosen to support a QoS approach that does not attempt to guarantee that a fixed level of QoS will be preserved through topology changes. (We believe that to fully ensure that a fixed level QoS can be maintained in a dynamic environment would result in a significant under-utilization of the network.) Instead, when topology changes occur (only bandwidth at the present time), our approach allows the new level of available resources to be discovered and allows QoS levels to be adjusted accordingly. QoS ranges are interpreted here to be the bounds of what the network could provide for a given flow. The network provides service at a signaled QoS allocation point within the range requested in the QoS reservation request.

As the number of application flows competing for resources increases, rather than simply refusing to admit new flows or preempting existing flows, the algorithm attempts to adjust the allocation for each flow, so that all flows can be accommodated. The algorithm attempts to give each flow the minimum bandwidth requested, plus a fraction which is proportional to the requested bandwidth range. The algorithm uses information collected by dRSVP on flow bottlenecks, both upstream and downstream, to determine the amount of bandwidth available for a given flow.

In order to reduce protocol overhead, we execute the bandwidth allocation algorithm only when a new Resv message is received and either a parameter associated with the flow (i.e., flowspecs, m_r, m_s) has changed or the

outbound link's bandwidth has changed. If the bandwidth allocation is invoked, Resv messages will be sent to upstream senders to refresh reservation state for a given flow. Furthermore, even though the algorithm computes new bandwidth allocations for all flows, to avoid creating a cascading "storm" of Resv messages, we only send the Resv upstream for the flow which is being refreshed. This keeps overhead low, at the expense of delaying possible application adaptation. The sender application will only learn about reservation allocation changes at most once per refresh interval. If the interval is long with respect to the rate at which the network resources are changing, the application may over or under utilize the network, possibly resulting in non-conforming application traffic which would be subject to policing.

3.3 Application Interface

For the application to signal its requirements and to adapt to dynamic network conditions, the API between dRSVP and the application needed to be modified. We extended a current RSVP API to include this capability, which is called the dRSVP Application Programming Interface (dSCRAPI). This API is based on the SCRAPI interface provided with ISI's RSVP implementation [14]. The API allows an application to specify the range of bandwidths within which it is capable of operating, and to request QoS support for operation within this range. A "callback" mechanism is provided to allow the application to learn the status of a reservation request, and to learn the current allocated bandwidth within the requested range. The application can then adapt its transmission rate to the allocated level, and will receive QoS support for its traffic.

4. IMPLEMENTATION STATUS AND DISCUSSION

We have implemented dRSVP by making additions to ISI's version of the RSVP protocol running under FreeBSD. We run dRSVP on Intel-based PCs with the altq package installed[15][16]. Class Based Queueing (CBQ) [17] is configured per interface, with a percentage of the available bandwidth allocated to the reserved traffic class, control traffic class, and best-effort traffic class. Our implementation provides a Controlled Load service [18], which means that if the application adjusts its transmission to stay within the current level of service, it will not experience congestion. In this regard, our approach could be considered a form of congestion control with explicit rate indication. However, our approach could also be extended to other service models (e.g., Guaranteed QoS).

We have created a testbed in which we can emulate changing link bandwidths. Currently, the testbed consists of an internetwork of Ethernet LANs, with a maximum of 2 nodes per LAN; a Control LAN that is connected to all testbed nodes; a testbed controller process running on the Control LAN that sends commands to different nodes to modify interface bandwidths; and a bandwidth manager process running on all testbed machines that accepts bandwidth commands from the testbed controller and informs the dRSVP process. To reflect the change in link bandwidth for a given interface, the dRSVP process modifies the corresponding CBQ parameters in the kernel to reflect the change in bandwidth for a given (outbound) interface. In subsequent testing we found these changes actually resulted in the anticipated bandwidth changes on the outbound interface. Eventually, we would like to migrate to a radio environment.

Also part of the testbed are a set of adaptive applications. We modified a streaming audio application (vat) and video application (vic) to support bandwidth conditions that work with the new API. Also modified were several RSVP-aware test applications created by the Naval Research Laboratories, called mgen and drec. We implemented the new API, dSCRAPI, as an extension to SCRAPI [14].

Tests and demonstrations performed in our testbed comparing standard and dynamic RSVP have illustrated the benefits of the adaptive QoS approach in a varying bandwidth environment. Specifically, we have demonstrated that dRSVP divides available bandwidth among several applications, that applications adapt to variable bandwidth allocations, and that they receive QoS support under degraded conditions. A quantitative analysis is currently underway. Two items in particular we want to study are the overhead of dRSVP and its impact in the dynamic network environment, and the impact of dRSVP on user/application performance. We are also interested in the stability of protocol when node movement occurs and the effectiveness of dRSVP under these conditions. Since dRSVP relies on "soft state" to re-establish QoS along a new route, when a route does change, there will be a period during which a flow receives only best-effort service.

5. CONCLUSION

The problem of supporting QoS in a dynamic network environment is complex and will require QoS support at multiple layers of the protocol stack. The applications running in this environment will influence the type of support provided. We believe there is a large class of applications that can tolerate transient periods of degraded service, yet benefit from the Dynamic QoS mechanisms proposed here. To adapt to changing network conditions, these applications need to be informed of changes.

This paper has presented an end-to-end approach to dealing with dynamic network conditions and briefly discusses an implementation of the approach. The Dynamic QoS approach rests on the assumption is that the link layer has the capability to inform the network layer of effective bandwidth changes. This approach needs to be quantitatively evaluated, ideally in a realistic environment. Evaluations are currently underway to evaluate our implementation in an emulated dynamic bandwidth environment, and results should provide some understanding of its effectiveness.

ACKNOWLEDGEMENTS

We gratefully acknowledge the contributions of Bob Moose on this project.

REFERENCES

[1] Braden, R. D. Clark, S. Shenker, *Integrated Services in the Internet Architecture*, IETF, RFC 1633, June 1994.

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

[3] Blake, S., D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, *An Architecture for Differentiated Services*, IETF, RFC 2475, December 1998.

[4] D. Beyer, T. Frivold, J. Hight, M. Lewis, *API Framework for Internet Radios*, September 1998. ftp://ftp.rooftop.com/pub/apis/api_framework.pdf

[5] Cansever, D., A. Michelson, and A. Levesque, *Quality of Service Support in Mobile ad-hoc IP Networks*, Proceedings of MILCOM 1999, October 1999.

[6] Stemm, M., R. Katz, *Vertical Handoffs in Wireless Overlay Networks*, ACM Mobile Networking (MONET), Special Issue on Mobile Networking in the Internet, Winter 1998.

[7] Talukdar, A., B. Badrinath, *MRSVP: A Reservation Protocol for an Integrated Services Packet Network with Mobile Hosts*, Department of Computer Science Technical Report DCS-TR-337, Rutgers University, July 1997.

[8] Talukdar, A., B. Badrinath, and A. Acharya, *On Accommodating Mobile Hosts in an Integrated Services Packet Network*, Proceedings of INFOCOM '97, Kobe, Japan, April 1997.

[9] Lu, S., V. Bharghavan, *Adaptive Resource Management Algorithms for Indoor Mobile Computing Environments*, Proceedings of ACM SIGCOMM '96, Univ. of Illinois at Urbana-Champaign, Stanford, CA, August 1996.

[10] Lu, S., K. Lee, V. Bharghavan, *Adaptive Service in Mobile Computing Environments*, from *"Building QoS into Distributed Systems"*; Campbell and Nharstedt (editors); Chapman & Hall; 1997.

[11] Lee, S. and A. Campbell, *INSIGNIA: In-band Signaling Support for QOS in Mobile Ad Hoc Networks*, Proc of 5th International Workshop on Mobile Multimedia Communications (MoMuC,98), Berlin, Germany, October 1998.

[12] Angin, O., A. Campbell, M. Kounavis, R. Liao, *The Mobiware Toolkit: Programmable Support for Adaptive Mobile Computing*, IEEE Personal Communications Magazine, August 1988.

[13] Sivakumar, R., P. Sinha, V. Bharghavan, *CEDAR: A Core-Extraction Extraction Distributed Ad Hoc Routing Algorithm*, IEEE Journal on Selected Areas in Communications, Vol 17, No. 8, August 1999.

[14] Lindell, B., *SCRAPI – A Simple 'Bare Bones' API for RSVP*; Internet Draft, Work in progress, August 1998.

[15] Cho, K., *A Framework for Alternate Queueing: Towards Traffic Management by PC-UNIX Based Routers*, Proceedings of USENIX 1998 Annual Technical Conference, New Orleans LA, June 1998. www.csl.sony.co.jp/~kjc/kjc/papers.html

[16] Cho, K., *Managing Traffic with ALTQ*; Proceedings of USENIX, 1999 Annual Technical Conference: FREENIX Track, Monterey CA; June 1999. www.csl.sony.co.jp/~kjc/kjc/papers.html

[17] Floyd, S., V. Jacobson, *Link-Sharing and Resource Management Models for Packet Networks*, IEEE/ACM Transactions on Networking, Vol. 3, No. 4, 365-386, August 1995.

[18] Wroclawski, J., *Specification of the Controlled-Load Network Element Service*, IETF RFC 2211, September 1997.