

Internet Engineering Task Force
Internet Draft
[draft-rosenberg-rtpproto-00.txt](#)
March 6, 1998
Expires: September 6, 1998

AVT
J.Rosenberg,B.Aboba,H.Schulzrinne
Bell Labs/Microsoft/Columbia

Elevating RTP to Protocol Status

STATUS OF THIS MEMO

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as ``work in progress''.

To learn the current status of any Internet-Draft, please check the ``[1id-abstracts.txt](#)'' listing contained in the Internet-Drafts Shadow Directories on [ftp.is.co.za](#) (Africa), [nic.nordu.net](#) (Europe), [munnari.oz.au](#) (Pacific Rim), [ds.internic.net](#) (US East Coast), or [ftp.isi.edu](#) (US West Coast).

Distribution of this document is unlimited.

[1](#) Abstract

This document discusses the issues involved in elevating RTP to the status of protocol, equivalent to TCP or UDP. This will result in all RTP packets being explicitly labeled as such in the packet header. This vastly simplifies the problem of classifying real time streams. Such classification operations are essential for successful deployment of RTP header compression, differentiated services, and traffic isolation. We define the format of the RTP protocol header, and discuss issues of backwards compatibility.

[2](#) Introduction

Over the last few years, RTP [[1](#)] has gained widespread acceptance as the transport protocol for real time services in the Internet. It has been adopted by the ITU-T as part of the H.323 protocol suite [[2](#)], it

is used in many commercial and public software packages, and it is used extensively on the mbone. Its success is due in large part to the growing presence of real time traffic on the Internet, a trend which is likely to continue.

At the same time, Internet routers and hosts are becoming more aware of the traffic that they are carrying. Commercial routers can drop, maintain logging records, and isolate in separate queues traffic based on UDP and TCP port numbers. The integrated services architecture [3,4,5], and the recently defined differentiated services, both require routers to classify traffic based on fields and parameters outside of the IP header.

Unfortunately, RTP has always been difficult to classify. Its packets are encapsulated in UDP, and do not use a single, static port number. This makes them very difficult to identify correctly. Hosts have resorted to heuristics, such as looking for periodic packets with certain byte values remaining static or incrementing predictably. These heuristics are both stateful and complex, and do not scale well at all.

Elevating RTP to protocol status, equivalent to TCP and UDP, would imply that RTP packets are explicitly labeled as such in the IP packet header. Such a change has a wide variety of useful implications, but also comes at a cost. This document discusses the motivations for such a change, discusses what the format of the new RTP would look like, describes implementation approaches, and discusses the important issue of backwards compatibility.

[3](#) Motivations

We see a number of important applications where large scale classification of RTP packets is necessary. We discuss each in detail, and weigh the usefulness of elevating RTP to protocol status as a solution.

[3.1](#) Differentiated Services

Work has recently begun to define differentiated services on the Internet. These services generally allow an ISP to mark packets at the periphery of the network. Within the network, the packets receive special treatment depending on the marking. For example, a premium service class has been defined. This service emulates a completely unloaded network, giving minimal delays and loss. Such a service is ideal for real time applications, for example. An ISP could use the premium service to offer improved quality for real time traffic only. In order to implement this, packets must be classified at the periphery so that they can be appropriately marked.

By having RTP packets clearly identified, the periphery routers of the ISP will be able to mark them for premium service within the

ISP's network. Making RTP a protocol is an efficient way to accomplish this.

On the other hand, clients could mark their packets as being real time by setting bits in other fields of the packet. In fact, the very same TOS bits used by diffserv could be used by clients to express how they would like the packets to be treated. These bits could then be modified at the periphery of the network by the ISP to provide the desired level of service.

[3.2](#) Static Router Configuration

Most commercially available IP routers allow administrators to configure the router to queue packets separately based on the value of the source IP address, destination IP address, source port, destination port, protocol tuple. It would be very beneficial to an ISP to be able to identify real-time traffic, isolate it from other flows, and provide it with some amount of bandwidth. Currently, since RTP cannot be classified by 5-tuple, it is placed in the default queue with lots of other things that need much different handling compared to real time.

Having RTP be an IP protocol would improve the situation in two ways. First, RTP packets could be identified by the protocol field in the IP header. Secondly, the payload type field in the RTP header could be used to infer information about the codec used for the media compression. This would allow a router to make a very educated guess about the bandwidth and QoS requirements for the flow.

As with differentiated services, one could simply use a TOS value to indicate that the traffic was real time (which may or may not be RTP). However, having RTP packets explicitly labeled would allow routers to look in RTP-specific header fields for additional information.

[3.3](#) Firewalls

Having RTP packets be labeled would provide additional information that could be used by firewalls. Firewalls could be configured to

drop or accept all RTP traffic coming into or leaving a domain. Currently, RTP traffic cannot be distinguished from other applications which use dynamic UDP ports. This would no longer be the case.

Firewall administrators would also be able to block RTCP traffic, while admitting RTP. This is extremely useful for mbone sessions. Typically, these will have one sender of data, with hundreds of listeners, each of which sends only RTCP. These RTCP packets can cause a lot of network state (in the form of (S,G) entries) to be created,

and can also cause significant network traffic (due to flooding in DVMRP). Filtering incoming RTCP, but not RTP, would allow a network administrator to offer mbone service without needing to worry about these RTCP problems.

[3.4](#) RTP Header Compression

RTP header compression [\[6\]](#) is typically used over dialup modem lines where bandwidth is at a premium. Without it, the RTP/UDP/IP headers require 10 kbps alone when used with the G.729 speech coder with a 30ms packetization delay. With it, this rate is reduced to less than half a kilobit per second. This reduction can mean the difference between transmitting video and having no video at all. The compression algorithm itself relies on the compressor classifying RTP packets, and maintaining state for the flow. Unfortunately, when the compression is performed in the downstream direction (from the access device to the host), the access device must decide which packets to apply compression to. Having RTP packets be explicitly labeled makes this process vastly simpler. The compressor could also look at the SSRC field to decide to which compression context the packet belongs.

We observe that the access device could instead decide which packets are RTP based on signaling from the host. An additional PPP packet type could be created indicating the port number that the host is expecting RTP packets on. Hosts would need to send these packets after opening a socket for RTP. This would require explicit support in the OS, but it is probably no more complicated than the OS support required to elevate RTP to protocol status.

[4](#) Alternate Solution: RTP Port Numbers

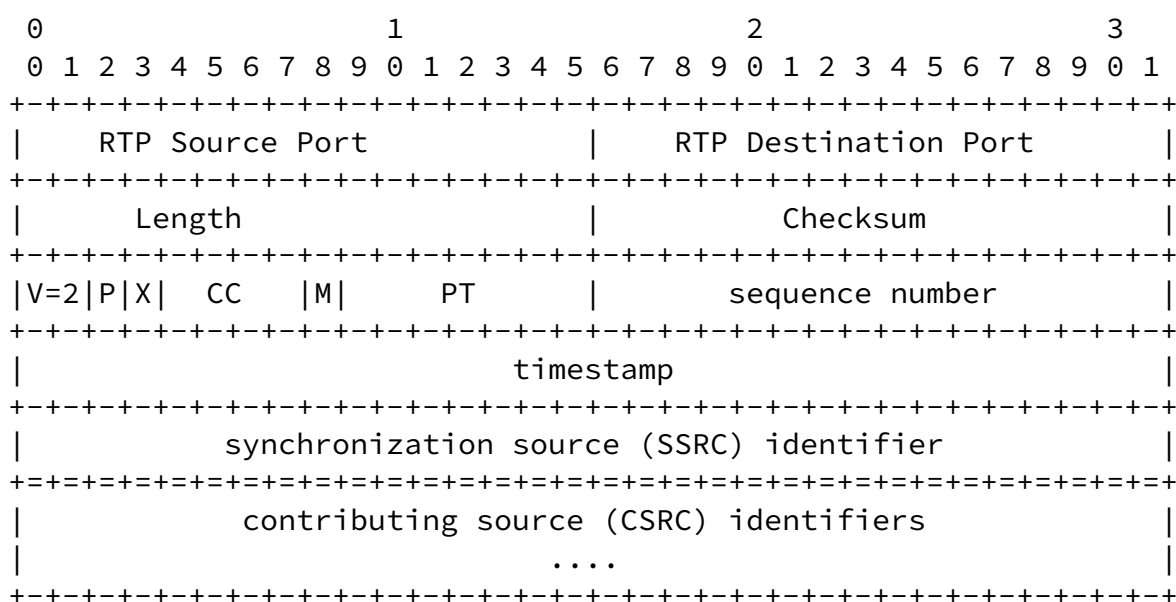
One approach to identifying RTP packets could be to assign it a static pair of port numbers (one for RTP, one for RTCP). This is not

acceptable, however. In some cases, a host may run several real time applications (such as a voice conferencing tool and a video conferencing tool), each of which may independently require its own ports. An alternative would be to assign a block of port numbers to RTP. This would allow for multiple simultaneous applications and still allow RTP traffic to be identified.

The weakness of this approach is twofold: (1) it requires a choice of an upper bound on the number of simultaneous RTP sessions in a client, and (2) other UDP-based applications with dynamic port numbers may randomly choose an RTP-assigned port. The latter problem can be made to eventually disappear if new applications are written to avoid the UDP space. In the interim (which may be forever), some non real time traffic could be mistakenly classified as real time.

5 Solution: RTP as Protocol

Our proposal would be to have the new RTP protocol have a header which is identical to the concatenation of the UDP header and RTP header. The new RTP protocol header would then look like:



The source port, destination port, length, and checksum have exactly the same syntax and semantics as in UDP. Here, however, the port numbers refer to RTP ports.

6 Impact on Network Aware Devices

Elevation of RTP to protocol status would have an impact on almost all network aware devices. This section discusses the implications for each.

6.1 Hosts

When a packet arrives at a host, the operating system typically looks at the IP protocol field, and processes the packet based on it. Since RTP would become a new protocol, it is important to consider how to handle it in operating systems. There seem to be three approaches:

1. Long Term. Significant extensions to the standard BSD sockets API can be made to support RTP as another socket type. The kernel would be upgraded to process RTP and RTCP packets. The sockets extensions would provide functions for interacting

with RTCP, and setting the various fields and parameters in the RTP header. Applications would need to be written to make use of the new API. It is not clear that having RTP in kernel space is the most efficient approach, but elevating RTP to full protocol makes it a possibility at least.

2. Near Term. A small extension to the BSD sockets API can be made to define a new protocol type (IPPROTO_RTP) which can be used when creating a socket of type SOCK_DGRAM. The resulting socket is identical to all respects to the standard UDP socket, except the value of the protocol field in the IP header is set to that of RTP when sending. When receiving, packets are sent to the socket only when the IP protocol field indicates RTP. When sending data to the socket, the kernel would add only the UDP portion of the RTP header. When receiving, it would strip only the UDP portion. This means that the RTP part of the header would still need to be processed in application space, but the UDP portion would be in the kernel. This solution requires absolutely minimal changes to existing application software, which perform RTP, but not UDP processing. The kernel modifications are minor.

3. Short Term. Before operating systems are changed at all, the new RTP protocol can be implemented purely in user space by making use of the raw socket. This would require applications to implement both the RTP and UDP portions of the protocol. The UDP code can easily be lifted from current operating systems. Even if it cannot, its implementation is fairly straightforward.

We anticipate that implementations would gradually migrate from the short term solution to the near term. It is not clear whether the long term solution is practical.

An unfortunate difficulty with the short term solution is that most operating systems only allow users with root permissions to access the raw socket. This may be problematic for many applications.

[6.2](#) Routers

Routers do not need to understand the IP protocol field in order to forward packets. However, most routers can be programmed with filters that drop or classify packets based on this field. This operation is only a problem if the routers cannot be configured to accept new values for this field. Routers which accept numeric values should operate correctly.

[6.3](#) Firewalls

J.Rosenberg,B.Aboba,H.Schulzrinne

[Page 6]

Internet Draft

RTP as Protocol

March 6, 1998

In all likelihood, most firewalls currently drop all traffic that is not UDP or TCP. This would cause the new RTP packets to be discarded ubiquitously by firewalls. To fix the problem, firewalls would need to be upgraded to recognize the new protocol type, and accept filter rules based on it.

[6.4](#) Tools

A number of host tools rely on examination of the IP protocol header. Most important among these is tcpdump, based on the Berkeley packet filter. Tcpdump would not be able to recognize the new RTP, and would need to be upgraded in order to do this. This issue is minor, but at least worth a mention.

Other similar tools, such as packet sniffers and network analyzers,

would also need to eventually be upgraded to recognize the new protocol. In the long run, this would be very advantageous. Network tools cannot recognize RTP packets at all. With this change, they would be able to recognize and decode RTP packets. They could also recognize and decode RTCP packets, which may provide valuable feedback when doing network debugging.

[6.5](#) Network Address Translators

Network Address Port Translation [[7](#)] allows for many hosts in a stub domain to map their IP addresses (which may be routable or unroutable) to a single IP address. NAPT does this by translating the source IP address and port number. Many NAPT devices currently only support UDP or TCP, and thus would be unable to handle a new protocol without an upgrade. In any case, real-time applications present special difficulties for NAT or NAPT implementations, since protocol such as SDP require IP addresses to be carried in application packets

[7](#) Backwards Compatibility

It is important to consider how to handle interoperability between end systems using the new RTP protocol and those using the old. We can classify RTP applications into two broad types - broadcast and interactive. The interoperability issues are different in both cases.

[7.1](#) Interactive

For interactive applications, there is usually some kind of signaling protocol that establishes communications before the media is transported using RTP. These signaling protocols usually indicate the various codecs that the end systems are capable of decoding and encoding. The use of the new RTP can be considered as just another capability. Both sides express their ability to receive the new RTP.

Applications implementing the new RTP should be prepared to also transmit using the old RTP. As a result, a common RTP version can always be found.

In the case of H.323, this would require the addition of a single new ASN.1 element in H.245 [[8](#)], which expresses the ability to receive the new RTP.

As an alternative, backwards compatibility can be achieved purely at

the RTP layer by making use of ICMP errors. Any host implementing the new RTP would also be required to implement the old one. Whenever a host listens to an RTP socket, the operating system accepts packets which are either UDP or the new RTP with the given port number. This will allow new RTP implementations to receive packets from both old and new without explicit application signaling. This comes at the expense of an effective sharing of port space between UDP and RTP.

When a host implementing the new RTP wishes to send a packet, it sends it using the RTP protocol number. Hosts which do not understand the new RTP protocol should generate an ICMP protocol unreachable error message, and return it back to the source. The sender's operating system must then use the old RTP for sending packets on that socket from that point forward. This will allow a new RTP implementation to talk to either a new or old one, thus achieving full backwards compatibility. The cost is additional smarts in the operating system, and potential loss of the first few RTP packets until the host switches back to RTP. This approach fails for multicast.

[7.2](#) Broadcast

For broadcast applications, such as mbone conferences, there is no capabilities exchange. However, there is usually some kind of session advertisement (using SAP [\[9\]](#) and SDP [\[10\]](#), for example). These announcements include the port numbers and multicast group used for the media. They could be extended to also indicate whether the new RTP protocol is being used. This would allow end systems to know whether they can correctly receive the media or not. If a broadcast is taking place using the new RTP, and there are old receivers trying to listen, they will not be able to receive the media. However, this is no different than the case where a broadcast is taking place using a codec that some client applications cannot decode. The only solution is a software upgrade of the clients.

[8](#) Conclusion

We have discussed the issues related to elevating RTP to protocol status, which would give it its own unique IP protocol number like TCP and UDP. The main motivation for this is to allow for

classification of RTP packets in end systems, routers, and other devices. Classification is needed for services such as RTP header compression, real time flow isolation, and differentiated services.

We have proposed that the new RTP protocol have a header which is the concatenation of the current UDP and RTP header fields. This simplifies implementation, and leaves open a smooth migration path for deployment. We have also discussed backwards compatibility, and how it can be handled for broadcast and interactive applications.

[9](#) Full Copyright Statement

Copyright (C) The Internet Society (1998). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works.

However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."

[10](#) Authors Addresses

Jonathan Rosenberg
Bell Laboratories, Lucent Technologies
101 Crawfords Corner Rd.
Holmdel, NJ 07733
email: jdrosen@bell-labs.com

Bernard Aboba
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
email: aboba@microsoft.com

Henning Schulzrinne
Columbia University
M/S 0401
1214 Amsterdam Ave.
New York, NY 10027-7003
email: schulzrinne@cs.columbia.edu

11 Bibliography

- [1] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, RTP: a transport protocol for real-time applications, Request for Comments (Proposed Standard) [1889](#), Internet Engineering Task Force, Jan. 1996.
- [2] ITU-T, Recommendation H.323 - Visual Telephone Systems and Equipment for Local Area Networks which Provide Non-Guaranteed Quality of Service , February 1996.
- [3] J. Wroclawski, The use of RSVP with IETF integrated services, Request for Comments (Proposed Standard) [2210](#), Internet Engineering Task Force, Oct. 1997.
- [4] J. Wroclawski, Specification of the controlled-load network element service, Request for Comments (Proposed Standard) [2211](#), Internet Engineering Task Force, Oct. 1997.
- [5] R. Guerin, C. Partridge, and S. Shenker, Specification of guaranteed quality of service, Request for Comments (Proposed Standard) [2212](#), Internet Engineering Task Force, Oct. 1997.
- [6] S. Casner and V. Jacobson, Compressing IP/UDP/RTP headers for low-speed serial links, Internet Draft, Internet Engineering Task Force, Nov. 1996. Work in progress.
- [7] P. Srisuresh and K. Egevang, The ip network address translator (nat), (internet draft), Internet Engineering Task Force, Sept. 1997. Work in Progress.
- [8] International Telecommunication Union, Control protocol for mul-

Internet Draft

RTP as Protocol

March 6, 1998

Standardization Sector of ITU, Geneva, Switzerland, Mar. 1996.

[9] M. Handley, SAP: Session announcement protocol, Internet Draft, Internet Engineering Task Force, Nov. 1996. Work in progress.

[10] M. Handley, SDP: Session description protocol, Internet Draft, Internet Engineering Task Force, Nov. 1997. Work in progress.

