Network Working Group
Request for Comments: 2210
Category: Standards Track

J. Wroclawski
MIT LCS

September 1997

The Use of RSVP with IETF Integrated Services

Status of this Memo

Abstract

This note describes the use of the RSVP resource reservation protocol with the Controlled-Load and Guaranteed QoS control services. The RSVP protocol defines several data objects which carry resource reservation information but are opaque to RSVP itself. The usage and data format of those objects is given here.

1. Introduction

The Internet integrated services framework provides the ability for applications to choose among multiple, controlled levels of delivery service for their data packets. To support this capability, two things are required:

- Individual network elements (subnets and IP routers) along the path followed by an application's data packets must support mechanisms to control the quality of service delivered to those packets.

- A way to communicate the application's requirements to network elements along the path and to convey QoS management information between network elements and the application must be provided.

In the integrated services framework the first function is provided by QoS control services such as Controlled-Load [RFC 2211] and Guaranteed [RFC 2212]. The second function may be provided in a number of ways, but is frequently implemented by a resource reservation setup protocol such as RSVP [RFC 2205].

Because RSVP is designed to be used with a variety of QoS control services, and because the QoS control services are designed to be used with a variety of setup mechanisms, a logical separation exists between the two specifications. The RSVP specification does not define the internal format of those RSVP protocol fields, or objects, which are related to invoking QoS control services. Rather, RSVP treats these objects as opaque. The objects can carry different information to meet different application and QoS control service requirements.

Similarly, interfaces to the QoS control services are defined in a general format, so that the services can be used with a variety of setup mechanisms.

This RFC provides the information required to use RSVP and the integrated service framework's QoS control services together. It defines the usage and contents of three RSVP protocol objects, the FLOWSPEC, ADSPEC, and SENDER_TSPEC, in an environment supporting the Controlled-Load and/or Guaranteed QoS control services. If new services or capabilities are added to the integrated services framework, this note will be revised as required.

2. Use of RSVP

Several types of data must be transported between applications and network elements to correctly invoke QoS control services.

> NOTE: In addition to the data used to directly invoke QoS control services, RSVP carries authentication, accounting, and policy information needed to manage the use of these services. This note is concerned only with the RSVP objects needed to actually invoke QoS control services, and does not discuss accounting or policy objects.

This data includes:

- Information generated by each receiver describing the QoS control service desired, a description of the traffic flow to which the resource reservation should apply (the Receiver TSpec), and whatever parameters are required to invoke the service (the Receiver RSpec). This information is carried from the receivers to intermediate network elements and the sender(s) by RSVP FLOWSPEC objects. The information being carried in a FLOWSPEC object may change at intermediate points in the network due to reservation merging and other factors.

- Information generated at each sender describing the data traffic generated by that sender (the Sender TSpec). This information is carried from the sender to intermediate network elements and the receiver(s) by RSVP, but is never modified by intermediate elements within the network. This information is carried in RSVP SENDER_TSPEC objects.

- Information generated or modified within the network and used at the receivers to make reservation decisions. This information might include available services, delay and bandwidth estimates, and operating parameters used by specific QoS control services. this information is collected from network elements and carried towards receivers in RSVP ADSPEC objects. Rather than carrying information from each intermediate node separately to the receivers, the information in the ADSPEC represents a summary, computed as the ADSPEC passes each hop. The size of this summary remains (roughly) constant as the ADSPEC flows through the network, giving good scaling properties.

From the point of view of RSVP objects, the breakdown is as follows:

- The RSVP SENDER_TSPEC object carries the traffic specification (sender TSpec) generated by each data source within an RSVP session. It is transported unchanged through the network, and delivered to both intermediate nodes and receiving applications.

- The RSVP ADSPEC object carries information which is generated at either data sources or intermediate network elements, is flowing downstream towards receivers, and may be used and updated inside the network before being delivered to receiving applications. This information includes both parameters describing the properties of the data path, including the availability of specific QoS control services, and parameters required by specific QoS control services to operate correctly.

- The RSVP FLOWSPEC object carries reservation request (Receiver_TSpec and RSpec) information generated by data receivers. The information in the FLOWSPEC flows upstream towards data sources. It may be used or updated at intermediate network elements before arriving at the sending application.

  NOTE: The existence of both SENDER_TSPEC and ADSPEC RSVP objects is somewhat historical. Using the message format described in this note it would be possible to place all of the service control information carried "downstream" by RSVP in the same object. However, the distinction between data which is not updated within the network (in the SENDER_TSPEC object) and data which is updated within the network (in the ADSPEC object) may

be useful to an implementation in practice, and is therefore retained.

2.1 Summary of operation

Operation proceeds as follows:

An application instance participating in an RSVP session as a data sender registers with RSVP. One piece of information provided by the application instance is the Sender TSpec describing the traffic the application expects to generate. This information is used to construct an RSVP SENDER_TSPEC object, which is included in RSVP PATH messages generated for the application.

The sending application also constructs an initial RSVP ADSPEC object. This adspec carries information about the QoS control capabilities and requirements of the sending application itself, and forms the starting point for the accumulation of path properties described below. The ADSPEC is added to the RSVP PATH message created at the sender.

NOTE: For the convenience of application programmers, a host RSVP implementation may allow the sending application not to provide an initial adspec, instead supplying its own default. This usage is most likely when the application sender does not itself participate in the end-to-end QoS control process (by actively scheduling CPU usage and similar means) and does not itself care which QoS control service is selected by the receivers.

Typically the default ADSPEC supplied by the host RSVP in this case would support all QoS control services known to the host. However, the exact behavior of this mechanism is implementation dependent.

The ADSPEC is modified by subsequent network elements as the RSVP PATH message moves from sender to receiver(s). At each network element, the ADSPEC is passed from RSVP to the traffic control module. The traffic control module updates the ADSPEC, which may contain data for several QoS control services, by identifying the services mentioned in the ADSPEC and calling each such service to update its portion of the ADSPEC. If the traffic control module discovers a QoS control service mentioned in the ADSPEC but not implemented by the network element, a flag is set to report this to the receiver. The updated ADSPEC is then returned to RSVP for delivery to the next hop along the path.

Upon arrival of the PATH message at an application receiver, the data in the SENDER_TSPEC and ADSPEC objects is passed across the RSVP API to the application. The application (perhaps with the help of a library of common resource-reservation functions) interprets the arriving data, and uses it to guide the selection of resource reservation parameters. Examples of this include use of the arriving "PATH_MTU" composed characterization parameter [RFC 2215] to determine the maximum packet size parameter in the reservation request and use of the arriving Guaranteed service "C" and "D" parameters [RFC 2212] to calculate a mathematical bound on delivered packet delay when using the Guaranteed service.

An application receiver wishing to make a resource reservation supplies its local RSVP with the necessary reservation parameters. Among these are the QoS control service desired (Guaranteed or Controlled-Load), the traffic specifier (TSpec) describing the level of traffic for which resources should be reserved, and, if needed by the selected QoS control service, an RSpec describing the level of service desired. These parameters are composed into an RSVP FLOWSPEC object and transmitted upstream by RSVP.

At each RSVP-aware point in the network, the SENDER_TSPECs arriving in PATH messages and the FLOWSPECs arriving in RESV messages are used to request an appropriate resource reservation from the desired QoS control service. State merging, message forwarding, and error handling proceed according to the rules of the RSVP protocol.

Finally, the merged FLOWSPEC object arriving at each of an RSVP session’s data senders is delivered to the application to inform each sender of the merged reservation request and properties of the data path.

2.2. RSVP support for multiple QoS control services

The design described in this note supports RSVP sessions in which the receivers choose a QoS control service at runtime.

To make this possible, a receiver must have all the information needed to choose a particular service before it makes the choice. This means that the RSVP SENDER_TSPEC and ADSPEC objects must provide the receivers with information for all services which might be chosen.

The Sender TSpec used by the two currently defined QoS control services is identical. This simplifies the RSVP SENDER_TSPEC object, which need carry only a single TSpec data structure in this shared format. This common SENDER_TSPEC can be used with either Guaranteed or Controlled-Load service.

The RSVP ADSPEC carries information needed by receivers to choose a service and determine the reservation parameters. This includes:

- Whether or not there is a non-RSVP hop along the path. If there is a non-RSVP hop, the application's traffic will receive reservationless best-effort service at at least one point on the path.

- Whether or not a specific QoS control service is implemented at every hop along the path. For example, a receiver might learn that at least one integrated-services aware hop along the path supports the Controlled-Load service but not the Guaranteed service.

- Default or global values for the general characterization parameters described in [RFC 2215]. These values describe properties of the path itself, irrespective of the selected QoS control service. A value reported in this section of the ADSPEC applies to all services unless a different, service-specific value is also present in the ADSPEC.

- A service-specific value for one or more general characterization parameters, if the service-specific value differs from the default value.

- Values of the per-service characterization parameters defined by each supported service.

Data in the ADSPEC is divided into blocks or fragments, each of which is associated with a specific service. This allows the adspec to carry information about multiple services, allows new services to be deployed in the future without immediately updating existing code, and allows an application which will never use a particular service to omit the ADSPEC data for that service. The structure of the ADSPEC is described in detail in Section 3.3.

A sender may indicate that a specific QoS control service should *not* be used by the receivers within an RSVP session. This is done by omitting all mention of that service from the ADSPEC, as described in Section 3.3. Upon arrival at a receiver, the complete absence of an ADSPEC fragment for a specific service indicates to receivers that the service should not be used.

NOTE: In RSVP Version 1, all receivers within a session are required to choose the same QoS control service. This restriction is imposed by the difficulty of merging reservations requesting different QoS control services, and the current lack of a general service replacement mechanism. The restriction may be eliminated in the future.

Considering this restriction, it may be useful to coordinate the receivers' selection of a QoS control service by having the sender(s) offer only one choice, using the ADSPEC mechanism mentioned above. All receivers must then select the same service. Alternatively, the coordination might be accomplished by using a higher-level session announcement and setup mechanism to inform the receivers of the QoS control service in use, by manual configuration of the receivers, or by an agreement protocol running among the session receivers themselves.

As with the ADSPEC, the FLOWSPEC and SENDER_TSPEC object formats described in Section 3 are capable of carrying TSpecs and RSpecs for more than one QoS control service in separate data fragments. Currently, use of a FLOWSPEC or SENDER_TSPEC containing fragments for more than one QoS control service is not supported. In the future, this capability may be used to implement a more flexible service request and replacement scheme, allowing applications to obtain useful end-to-end QoS control when not all intermediate nodes support the same set of QoS services. RSVP-application APIs should be designed to support passing SENDER_TSPEC, FLOWSPEC, and ADSPEC objects of variable size and containing information about multiple QoS control services between RSVP and its clients.

2.3. Use of ADSPEC Information

This section gives some details about setting reservation parameters and the use of information conveyed by the RSVP ADSPEC object.

2.3.1. Determining the availability of a QoS control service

The RSVP ADSPEC carries flag bits telling the application receivers whether or not a completely reservation-capable path exists between each sender and the receiver. These bits are called "break bits", because they indicate breaks in the QoS control along a network path. Break bits are carried within the header which begins each per-service data fragment of an RSVP ADSPEC.

Service number 1 is used within the ADSPEC to identify a fragment carrying information about global parameter values that apply to all services (see [RFC 2215] for more details). The break bit in service 1's per-service header is used to tell the receiver(s) whether all of the network elements along the path from sender to receiver support RSVP and integrated services. If a receiver finds this bit set, at least one network element along the data transmission path between the ADSPEC's sender and the receiver can not provide QoS control services at all. This bit corresponds to the global NON_IS_HOP characterization parameter defined in [RFC 2215].

NOTE: If this bit is set, the values of all other parameters in the ADSPEC are unreliable. The bit being set indicates that at least one node along the sender-receiver path did not fully process the ADSPEC.

Service-specific break bits tell the receiver(s) whether all of the network elements along the path from sender to receiver support a particular QoS control service. The break bit for each service is carried within the ADSPEC's per-service header for that service. If a bit is set at the receiver, at least one network element along the data transmission path supports RSVP but does not support the QoS control service corresponding to the per-service header. These bits correspond to the service-specific NON_IS_HOP characterization parameters defined in [RFC 2215].

Section 3 gives more information about break bits.

2.3.2. Determining Path MTU

Both Guaranteed and Controlled-Load QoS control services place an upper bound on packet size, and require that the application limit the maximum size of packets subject to resource reservation. For both services, the desired maximum packet size is a parameter of the reservation request, and the service will reject (with an admission control error) reservation requests specifying a packet size larger than that supported by the service.

Since RSVP reservation requests are made by receivers, this implies that the *receivers* in an RSVP session, as well as the senders, need to know the MTU supported by the QoS control services along a data path. Further, in some unusual cases the MTU supported by a QoS control service may differ from that supported by the same router when providing best effort service.

A scalable form of MTU negotiation is used to address these problems. MTU negotiation in an RSVP system works as follows:

- Each sending application joining an RSVP session fills in the M (maximum packet size) parameter in its generated Sender_TSpec (carried from senders to receivers in a SENDER_TSPEC object) with the maximum packet size it wishes to send covered by resource reservation.

- Each RSVP PATH message from a sending application also carries an ADSPEC object containing at least one PATH_MTU characterization parameter. When it arrives at the receiver, this parameter gives the minimum MTU at any point along the path from sender to receiver. Generally, only the "global" PATH_MTU parameter

(service 1, parameter 9) will be present, in which case its value is a legal MTU for all reservation requests. If a service specific PATH_MTU parameter is present, its value will be smaller than that of the global parameter, and should be used for reservation requests for that service.

- Each receiver takes the minimum of all the PATH_MTU values (for the desired QoS control service) arriving in ADSPEC messages from different senders and uses that value as the MTU in its reservation requests. This value is used to fill in the M parameter of the TSpec created at the receiver. In the case of a FF style reservation, a receiver may also choose to use the MTU derived from each sender's ADSPEC in the FLOWSPEC generated for that sender, if the receiver is concerned about obtaining the maximum MTU on each data path. To accomodate changes in the data path, the receiver may continue to watch the arriving ADSPECS, and modify the reservation if a newly arriving ADSPEC indicates a smaller MTU than is currently in use.

- As reservation requests (RESV messages) move from receivers to senders, reservation parameters are merged at intermediate nodes. As part of this merging, the smaller of two M parameters arriving at a merge point will be forwarded in the upstream RESV message.

- As reservation requests arrive at intermediate RSVPs, the minimum of the receivers' requested TSpec and the sum of the sender TSpecs is taken, and a reservation for the resulting TSpec is made. The reservation will use the smaller of the actual path MTU value computed by the receivers and the largest maximum packet size declared by any of the sender(s). (The TSpec sum() function result's M parameter is the max of the summed TSpec M parameters).

- When the completely merged RESV message arrives at each sender, the MTU value (M parameter) in the merged FLOWSPEC object will have been set to the smallest acceptable MTU of the data paths from that sender to any session receiver. This MTU should be used by the sending application to size its packets. Any packets larger than this MTU may be delivered as best-effort rather than being covered by the session's resource reservation.

Note that senders do *not* adjust the value of their Sender_TSpec's M field to match the actual packet size selected in this step. The value of M represents the largest packet the sender could send, not the largest packet the sender is currently sending.

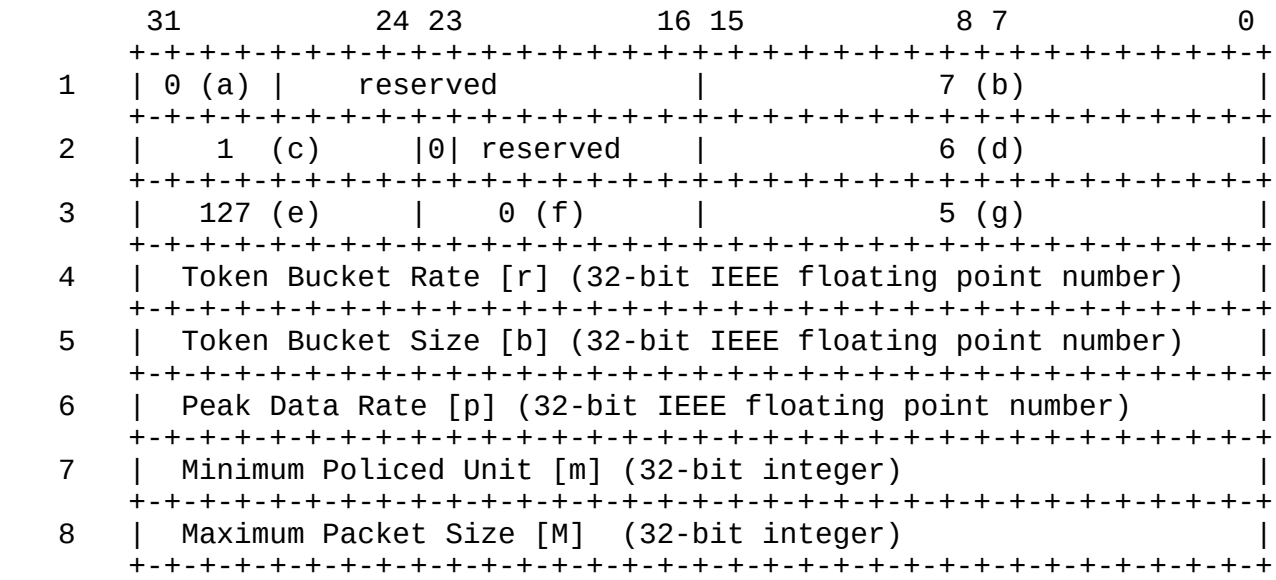
Note that the scheme above will allow each sender in a session to use the largest MTU appropriate for that sender, in cases where different data paths or receivers have different acceptable MTU's.

3. RSVP Object Formats

This section specifies the detailed contents and wire format of RSVP SENDER_TSPEC, ADSPEC, and FLOWSPEC objects for use with the Guaranteed and Controlled-Load QoS control services. The object formats specified here are based on the general message construction rules given in Appendix 1.

3.1. RSVP SENDER_TSPEC Object

The RSVP SENDER_TSPEC object carries information about a data source's generated traffic. The required RSVP SENDER_TSPEC object contains a global Token_Bucket_TSpec parameter (service_number 1, parameter 127, as defined in [RFC 2215]). This TSpec carries traffic information usable by either the Guaranteed or Controlled-Load QoS control services.



(a) - Message format version number (0)
(b) - Overall length (7 words not including header)
(c) - Service header, service number 1 (default/global information)
(d) - Length of service 1 data, 6 words not including header
(e) - Parameter ID, parameter 127 (Token_Bucket_TSpec)
(f) - Parameter 127 flags (none set)
(g) - Parameter 127 length, 5 words not including header

In this TSpec, the parameters [r] and [b] are set to reflect the sender's view of its generated traffic. The peak rate parameter [p] may be set to the sender's peak traffic generation rate (if known and controlled), the physical interface line rate (if known), or positive infinity (if no better value is available). Positive infinity is represented as an IEEE single-precision floating-point number with an exponent of all ones (255) and a sign and mantissa of all zeros. The format of IEEE floating-point numbers is further summarized in [RFC 1832].

The minimum policed unit parameter [m] should generally be set equal to the size of the smallest packet generated by the application. This packet size includes the application data and all protocol headers at or above the IP level (IP, TCP, UDP, RTP, etc.). The size given does not include any link-level headers, because these headers will change as the packet crosses different portions of the internetwork.

The [m] parameter is used by nodes within the network to compute the maximum bandwidth overhead needed to carry a flow's packets over the particular link-level technology, based on the ratio of [m] to the link-level header size. This allows the correct amount of bandwidth to be allocated to the flow at each point in the net. Note that smaller values of this parameter lead to increased overhead estimates, and thus increased likelyhood of a reservation request being rejected by the node. In some cases, an application transmitting a low percentage of very small packets may therefore choose to set the value of [m] larger than the actual minimum transmitted packet size. This will increase the likelyhood of the reservation succeeding, at the expense of policing packets of size less than [m] as if they were of size [m].

Note that the an [m] value of zero is illegal. A value of zero would indicate that no data or IP headers are present, and would give an infinite amount of link-level overhead.

The maximum packet size parameter [M] should be set to the size of the largest packet the application might wish to generate, as described in Section 2.3.2. This value must, by definition, be equal to or larger than the value of [m].

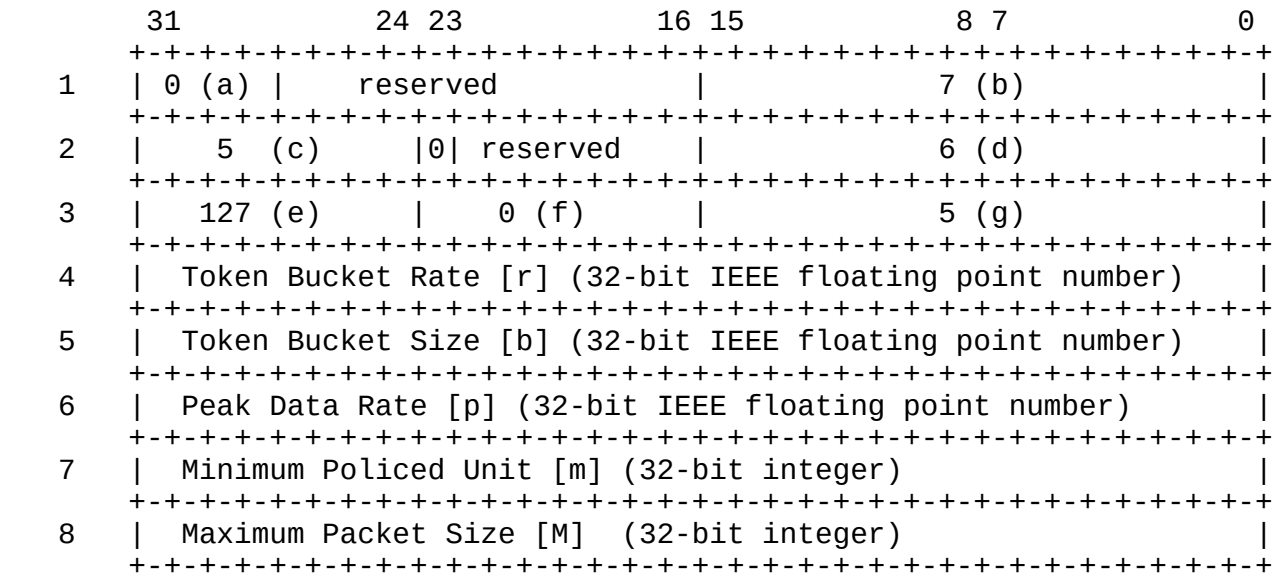
3.2. RSVP FLOWSPEC Object

The RSVP FLOWSPEC object carries information necessary to make reservation requests from the receiver(s) into the network. This includes an indication of which QoS control service is being requested, and the parameters needed for that service.

The QoS control service requested is indicated by the service_number in the FLOWSPEC's per-service header.

3.2.1 FLOWSPEC object when requesting Controlled-Load service

The format of an RSVP FLOWSPEC object originating at a receiver requesting Controlled-Load service is shown below. Each of the TSpec fields is represented using the preferred concrete representation specified in the 'Invocation Information' section of [RFC 2211]. The value of 5 in the per-service header (field (c), below) indicates that Controlled-Load service is being requested.



(a) - Message format version number (0)
(b) - Overall length (7 words not including header)
(c) - Service header, service number 5 (Controlled-Load)
(d) - Length of controlled-load data, 6 words not including per-service header
(e) - Parameter ID, parameter 127 (Token Bucket TSpec)
(f) - Parameter 127 flags (none set)
(g) - Parameter 127 length, 5 words not including per-service header

In this object, the TSpec parameters [r], [b], and [p] are set to reflect the traffic parameters of the receiver's desired reservation (the Reservation TSpec). The meaning of these fields is discussed fully in [RFC 2211]. Note that it is unlikely to make sense for the [p] term to be smaller than the [r] term.

The maximum packet size parameter [M] should be set to the value of the smallest path MTU, which the receiver learns from information in arriving RSVP ADSPEC objects. Alternatively, if the receiving application has built-in knowledge of the maximum packet size in use within the RSVP session, and this value is smaller than the smallest path MTU, [M] may be set to this value. Note that requesting a value of [M] larger than the service modules along the data path can support will cause the reservation to fail. See section 2.3.2 for further discussion of the MTU value.

The value of [m] can be chosen in several ways. Recall that when a resource reservation is installed at each intermediate node, the value used for [m] is the smaller of the receiver's request and the values in each sender's SENDER_TSPEC.

If the application has a fixed, known minimum packet size, than that value should be used for [m]. This is the most desirable case.

For a shared reservation style, the receiver may choose between two options, or pick some intermediate point between them.

- if the receiver chooses a large value for [m], then the reservation will allocate less overhead for link-level headers. However, if a new sender with a smaller SENDER_TSPEC [m] joins the session later, an already-installed reservation may fail at that time.

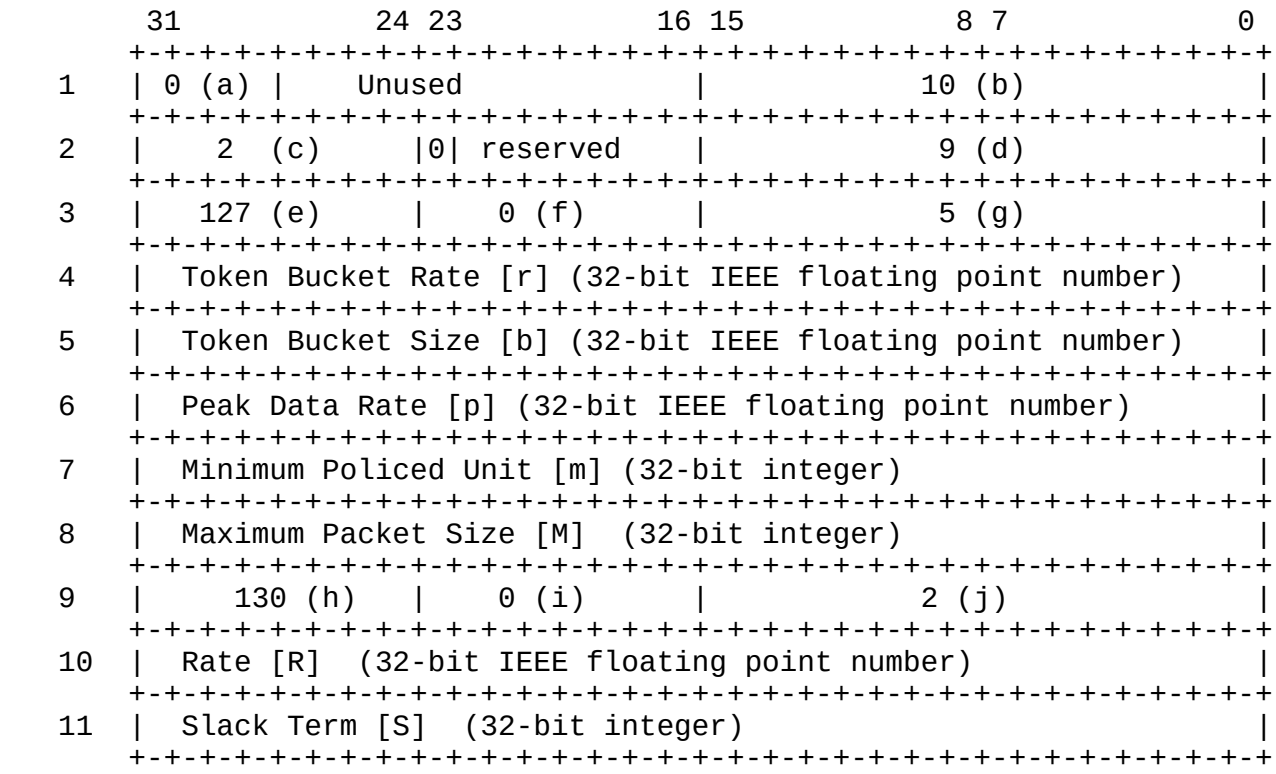
- if the receiver chooses a value of [m] equal to the smallest value which might be used by any sender, then the reservation will be forced to allocate more overhead for link-level headers. However it will not fail later if a new sender with a smaller SENDER_TSPEC [m] joins the session.

For a FF reservation style, if no application-specific value is known the receiver should simply use the value of [m] arriving in each sender's SENDER_TSPEC for its reservation request to that sender.

3.2.2. FLOWSPEC Object when Requesting Guaranteed Service

The format of an RSVP FLOWSPEC object originating at a receiver requesting Guaranteed service is shown below. The flowspec object used to request guaranteed service carries a TSpec and RSpec specifying the traffic parameters of the flow desired by the receiver.

Each of the TSpec and RSpec fields is represented using the preferred concrete representation specified in the 'Invocation Information' section of [RFC 2212]. The value of 2 for the service header identifier (field (c) in the picture below) indicates that Guaranteed service is being requested.



(a) - Message format version number (0)
(b) - Overall length (9 words not including header)
(c) - Service header, service number 2 (Guaranteed)
(d) - Length of per-service data, 9 words not including per-service header
(e) - Parameter ID, parameter 127 (Token Bucket TSpec)
(f) - Parameter 127 flags (none set)
(g) - Parameter 127 length, 5 words not including parameter header
(h) - Parameter ID, parameter 130 (Guaranteed Service RSpec)
(i) - Parameter 130 flags (none set)
(j) - Parameter 130 length, 2 words not including parameter header

In this object, the TSpec parameters [r], [b], and [p] are set to reflect the traffic parameters of the receiver's desired reservation (the Reservation TSpec). The meaning of these fields is discussed fully in [RFC 2212]. Note that it is unlikely to make sense for the [p] term to be smaller than the [r] term.

The RSpec terms [R] and [S] are selected to obtain the desired bandwidth and delay guarantees. This selection is described in [RFC 2212].

The [m] and [M] parameters are set identically to those for the Controlled-Load service FLOWSPEC, described in the previous section.

3.3. RSVP ADSPEC Object

An RSVP ADSPEC object is constructed from data fragments contributed by each service which might be used by the application. The ADSPEC begins with an overall message header, followed by a fragment for the default general parameters, followed by fragments for every QoS control service which may be selected by application receivers. The size of the ADSPEC varies depending on the number and size of per-service data fragments present and the presence of non-default general parameters (described in Section 3.3.5).

The complete absence of a data fragment for a particular service means that the application sender does not know or care about that service, and is a signal to intermediate nodes not to add or update information about that service to the ADSPEC. It is also a signal to application receivers that they should not select that service when making reservations.

Each fragment present is identified by a per-service data header. Each header contains a field identifying the service, a break bit, and a length field.

The length field allows the ADSPEC information for a service to be skipped over by a network elements which does not recognize or implement the service. When an element does this, it sets the break bit, indicating that the service's ADSPEC data was not updated at at least one hop. Note that a service's break bit can be set without otherwise supporting the service in any way. In all cases, a network element encountering a per-service data header it does not understand simply sets bit 23 to report that the service is not supported, then skips over the rest of the fragment.

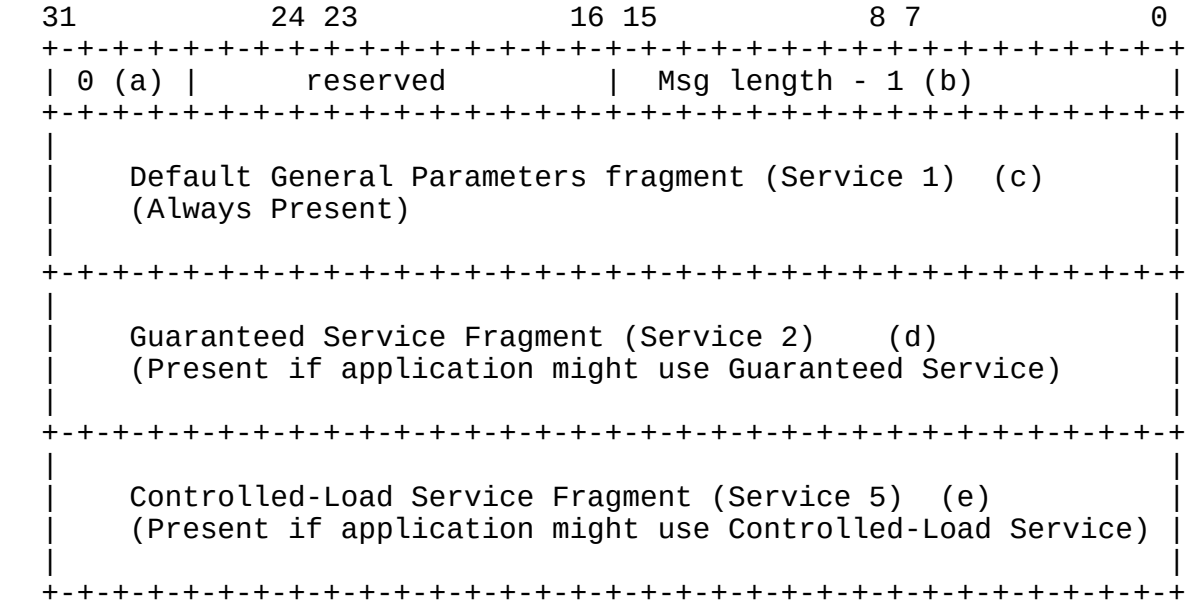
Data fragments must always appear in an ADSPEC in service_number order. In particular, the default general parameters fragment (service_number 1) always comes first.

Within a data fragment, the service-specific data must alway come first, followed by any non-default general parameters which may be present, ordered by parameter_number. The size and structure of the service-specific data is fixed by the service definition, and does not require run-time parsing. The remainder of the fragment, which carries non-default general parameters, varies in size and structure depending on which, if any, of these parameters are present. This part of the fragment must be parsed by examining the per-parameter headers.

Since the overall size of each data fragment is variable, it is always necessary to examine the length field to find the end of the fragment, rather than assuming a fixed-size structure.

3.3.1. RSVP ADSPEC format

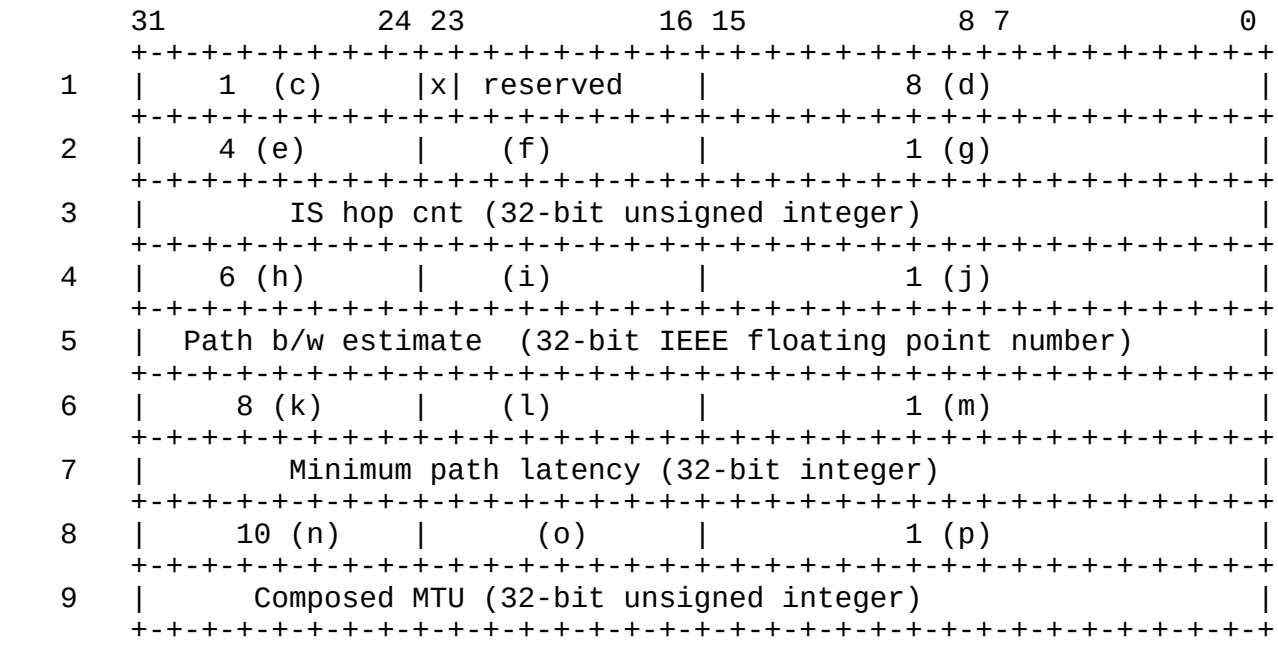
The basic ADSPEC format is shown below. The message header and the default general parameters fragment are always present. The fragments for Guaranteed or Controlled-Load service may be omitted if the service is not to be used by the RSVP session. Additional data fragments will be added if new services are defined.



(a) - Message format version number (0)
(b) - Overall message length not including header word
(c, d, e) - Data fragments

3.3.2. Default General Characterization Parameters ADSPEC data fragment

All RSVP ADSPECs carry the general characterization parameters defined in [RFC 2215]. Values for global or default general parameters (values which apply to the all services or the path itself) are carried in the per-service data fragment for service number 1, as shown in the picture above. This fragment is always present, and always first in the message.



(c) - Per-Service header, service number 1 (Default General Parameters)
(d) - Global Break bit ([RFC 2215], Parameter 2) (marked x) and length of General Parameters data block.
(e) - Parameter ID, parameter 4 (Number-of-IS-hops param from [RFC 2215])
(f) - Parameter 4 flag byte
(g) - Parameter 4 length, 1 word not including header
(h) - Parameter ID, parameter 6 (Path-BW param from [RFC 2215])
(i) - Parameter 6 flag byte
(j) - Parameter 6 length, 1 word not including header
(k) - Parameter ID, parameter 8 (minimum path latency from [RFC 2215])
(l) - Parameter 8 flag byte
(m) - Parameter 8 length, 1 word not including header
(n) - Parameter ID, parameter 10 (composed path MTU from [RFC 2215])
(o) - Parameter 10 flag byte
(p) - Parameter 10 length, 1 word not including header

Rules for composing general parameters appear in [RFC 2215].

In the above fragment, the global break bit (bit 23 of word 1, marked with (x) in the picture) is used to indicate the existence of a network element not supporting QoS control services somewhere in the data path. This bit is cleared when the ADSPEC is created, and set to one if a network element which does not support RSVP or integrated

services is encountered. An ADSPEC arriving at a receiver with this bit set indicates that all other parameters in the ADSPEC may be invalid, since not all network elements along the path support updating of the ADSPEC.

The general parameters are updated at every network node which supports RSVP:

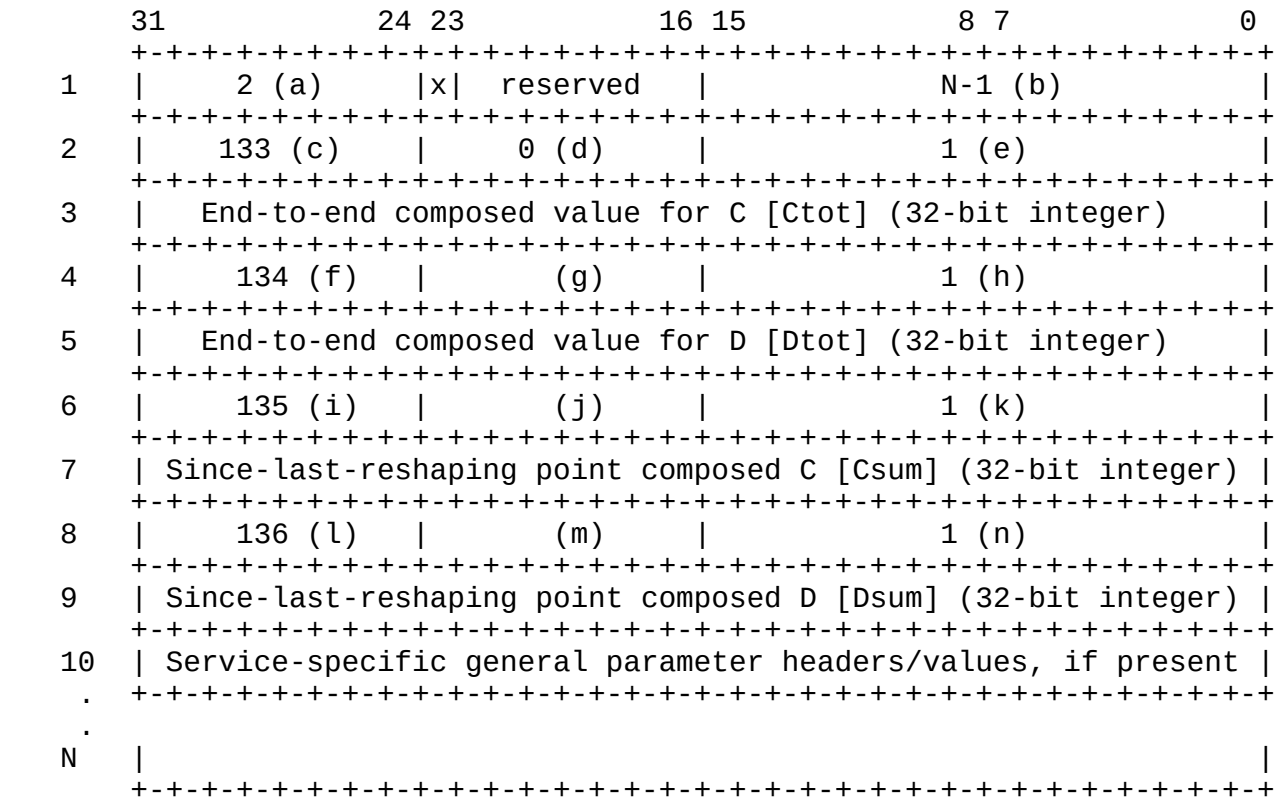
- When a PATH message ADSPEC encounters a network element implementing integrated services, the portion of the ADSPEC associated with service number 1 is passed to the module implementing general parameters. This module updates the global general parameters.

- When a PATH message ADSPEC encounters a network element that does *not* support RSVP or implement integrated services, the break bit in the general parameters service header must be set. In practice, this bit will usually be set by another network element which supports RSVP, but has been made aware of the gap in integrated services coverage.

- In either case, the ADSPEC is passed back to RSVP for delivery to the next hop along the path.

3.3.3. Guaranteed Service ADSPEC data fragment

The Guaranteed service uses the RSVP ADSPEC to carry data needed to compute the C and D terms passed from the network to the application. The minimum size of a non-empty guaranteed service data fragment is 8 32-bit words. The ADSPEC fragment for Guaranteed service has the following format:

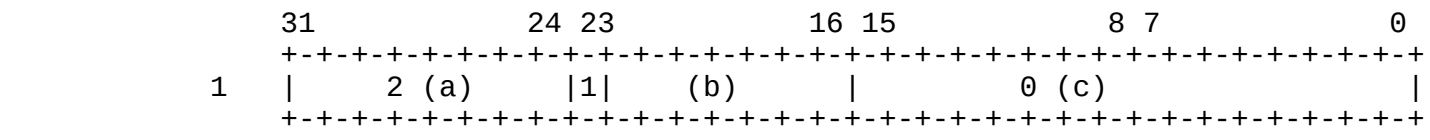


(a) - Per-Service header, service number 2 (Guaranteed)
(b) - Break bit and Length of per-service data in 32-bit words not including header word.
(c) - Parameter ID, parameter 133 (Composed Ctot)
(d) - Parameter 133 flag byte
(e) - Parameter 133 length, 1 word not including header
(f) - Parameter ID, parameter 134 (Composed Dtot)
(g) - Parameter 134 flag byte
(h) - Parameter 134 length, 1 word not including header
(i) - Parameter ID, parameter 135 (Composed Csum).
(j) - Parameter 135 flag byte
(k) - Parameter 135 length, 1 word not including header
(l) - Parameter ID, parameter 136 (Composed Dsum).
(m) - Parameter 136 flag byte
(n) - Parameter 136 length, 1 word not including header

When a node which actually implements guaranteed service creates the guaranteed service adspec fragment, the parameter values are set to the local values for each parameter. When an application or network

element which does not itself implement guaranteed service creates a guaranteed service adspec fragment, it should set the values of each parameter to zero, and set the break bit to indicate that the service is not actually implemented at the node.

An application or host RSVP which is creating a guaranteed service adspec fragment but does not itself implement the guaranteed service may create a truncated "empty" guaranteed adspec fragment consisting of only a header word:



(a) - Per-Service header, service number 2 (Guaranteed)
(b) - Break bit (set, service not implemented)
(c) - Length of per-service data in 32-bit words not including header word.

This might occur if the sending application or host does not do resource reservation iself, but still wants the network to do so. Note that in this case the break bit will always be set, since the creator of the adspec fragment does not itself implement guaranteed service.

When a PATH message ADSPEC containing a per-service header for Guaranteed service encounters a network element implementing Guaranteed service, the guaranteed service data fragment is updated:

- If the data block in the ADSPEC is an empty (header-only) block the header-only fragment must first be expanded into the complete data fragment described above, with initial values of Ctot, Dtot, Csum, and Dsum set to zero. An empty fragment can be recognized quickly by checking for a size field of zero. The value of the break bit in the header is preserved when the additional Guaranteed service data is added. The overall message length and the guaranteed-service data fragment size (field (b) in the pictures above) are changed to reflect the increased message length.

  The values of Ctot, Csum, Dtot, and Dsum in the ADSPEC data fragment are then composed with the local values exported by the network element according to the composition functions defined in [RFC 2212].

- When a PATH message ADSPEC with a Guaranteed service header encounters a network element that supports RSVP but does *not* implement Guaranteed service, the network element sets the break bit in the Guaranteed service header.

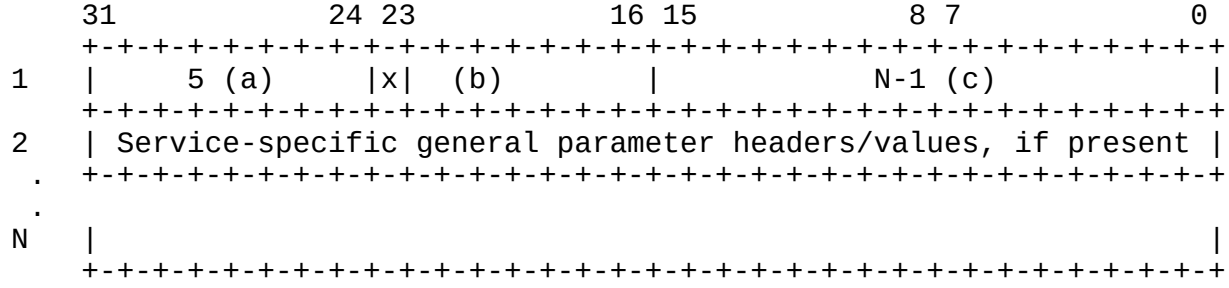
- The new values are placed in the correct fields of the ADSPEC, and the ADSPEC is passed back to RSVP for delivery to the next hop along the path.

When a PATH message ADSPEC containing a Guaranteed service data fragment encounters a network element that supports RSVP but does *not* implement Guaranteed service, the network element sets the break bit in the Guaranteed service header.

When a PATH message ADSPEC *without* a Guaranteed service header encounters a network element implementing Guaranteed service, the Guaranteed service module of the network element leaves the ADSPEC unchanged. The absence of a Guaranteed service per-service header in the ADSPEC indicates that the application does not care about Guaranteed service.

3.3.4. Controlled-Load Service ADSPEC data fragment

Unlike the Guaranteed service, the Controlled-Load service does not require extra ADSPEC data to function correctly. The only ADSPEC data specific to the Controlled-Load service is the Controlled-Load break bit. Therefore the usual Controlled-Load service data block contains no extra information. The minimum size of the controlled-load service data fragment is 1 32-bit word.



(a) - Per-Service header, service number 5 (Controlled-Load)
(b) - Break bit
(c) - Length of per-service data in 32 bit words not including header word.

The Controlled-Load portion of the ADSPEC is processed according to the following rules:

- When a PATH message ADSPEC with a Controlled-Load service header encounters a network element implementing Controlled-Load service, the network element makes no changes to the service header.

- When a PATH message ADSPEC with a Controlled-Load service header encounters a network element that supports RSVP but does *not* implement Controlled-Load service, the network element sets the break bit in the Controlled-Load service header.

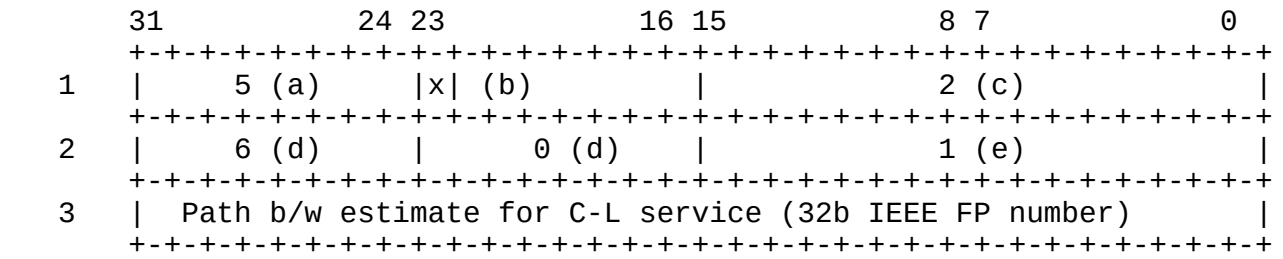
- In either case, the ADSPEC is passed back to RSVP for delivery to the next hop along the path.

3.3.5. Overriding Global ADSPEC Data with Service-Specific Information

In some cases, the default values for the general parameters are not correct for a particular service. For example, an implementation of Guaranteed service may accept only packets with a smaller maximum size than the link MTU, or the percentage of outgoing link bandwidth made available to the Controlled-Load service at a network element may be administratively limited to less than the overall bandwidth.

In these cases, a service-specific value, as well as the default value, is reported to the receiver receiving the ADSPEC. Service-specific information which overrides general information is carried by a parameter with the same name as the general parameter, placed within the data fragment of the QoS control service to which it applies. These service-specific values are referred to as override or service-specific general parameters.

For example, the following Controlled-Load ADSPEC fragment carries information overriding the global path bandwidth estimate with a different value:



(a) - Per-Service header, service number 5 (Controlled-Load)
(b) - Break bit
(c) - Length of per-service data, two words not including header
(c) - Parameter ID, parameter 6
(AVAILABLE_PATH_BANDWIDTH general parameter from [RFC 2215])
(d) - Parameter 6 flags (none set)
(e) - Parameter 6 length, one word not including header

The presence of override parameters in a data fragment can be quickly detected by examining the fragment's length field, which will be larger than the "standard" length for the fragment. Specific override parameters can be easily identified by examining the parameter headers, because they have parameter_number's from the general parameter portion of the number space (1-127), but are found in service-specific data blocks (those with service_numbers between 2 and 254 in the per_service header field).

The presence of override parameters in a data fragment is optional. A parameter header/value pair is added only when a particular application or QoS control service wishes to override the global value of a general parameter with a service-specific value.

As with IP options, it is only the use of these override parameters that is optional. All implementations must be prepared to receive and process override parameters.

The basic principle for handling override parameters is to use the override value (local or adspec) if it exists, and to use the default value otherwise. If a local node exports an override value for a general parameter, but there is no override value in the arriving adspec, the local node adds it. The following pseudo-code fragment gives more detail:

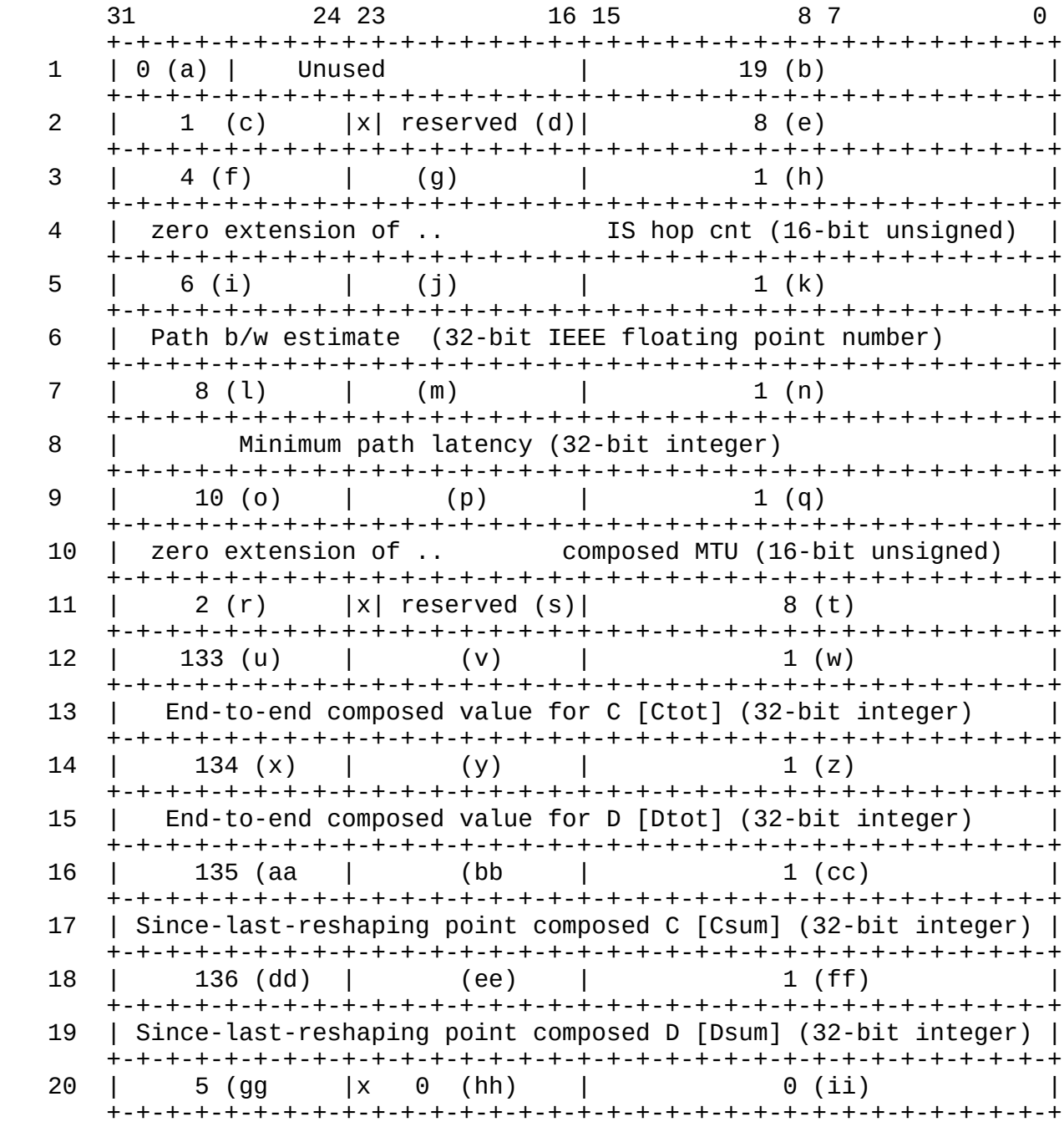
```
/* Adspec parameter processing rules *

<get arriving ADSPEC from RSVP>

for ( <each service number N with a fragment in the ADSPEC> ) {
  if ( <the local node does not support the service> ) {
    <set the break bit in the service header>
  } else {
    for ( <each parameter in the data fragment for service N> ) {
      if ( < the local service N supplies a value for the parameter> ) {
         <compose the arriving and values and update the adspec>
      } else {
         /* Must be a general parameter, or service N would have
          * supplied a value..
          */
         <compose the arriving value with the local default value
          and update the adspec>
      }
    }
    for ( <any parameters supplied by the local service N
          implementation but not found in the adspec> ) {
         /*
          * Must be an override value for a general parameter,
          * or the adspec would have contained a value..
          */
         <compose the local override value with the arriving default
          value (from the service 1 data fragment) and add the parameter
          to the adspec's service N fragment in parameter_number order>
    }
  }
}

<pass updated ADSPEC back to RSVP>
```

In practice, the two 'for' loops can be combined. Since override parameters within a service's fragment are transmitted in numerical order, it is possible to determine whether a parameter is present without scanning the entire fragment. Also, because the data fragments are ordered by service_number, the default values for general parameters will always be read before they might be needed to update local override values in the second for loop.

3.3.6. Example

The picture below shows the complete adspec for an application which can use either controlled-load or guaranteed service. In the example,

data fragments are present for general parameters, guaranteed, and controlled-load services. All fragments are of standard size, and there are no override parameters present.



Word 1: Message Header:
(a) - Message header and version number
(b) - Message length - 19 words not including header

Words 2-7: Default general characterization parameters
(c) - Per-Service header, service number 1
(Default General Parameters)
(d) - Global Break bit (NON_IS_HOP general parameter 2) (marked x)
(e) - Length of General Parameters data block (8 words)
(f) - Parameter ID, parameter 4 (NUMBER_OF_IS_HOPS
general parameter)
(g) - Parameter 4 flag byte
(h) - Parameter 4 length, 1 word not including header
(i) - Parameter ID, parameter 6 (AVAILABLE_PATH_BANDWIDTH
general parameter)
(j) - Parameter 6 flag byte
(k) - Parameter 6 length, 1 word not including header
(l) - Parameter ID, parameter 8 (MINIMUM_PATH_LATENCY
general parameter)
(m) - Parameter 8 flag byte
(n) - Parameter 8 length, 1 word not including header
(o) - Parameter ID, parameter 10 (PATH_MTU general parameter)
(p) - Parameter 10 flag byte
(q) - Parameter 10 length, 1 word not including header

Words 11-19: Guaranteed service parameters
(r) - Per-Service header, service number 2 (Guaranteed)
(s) - Break bit
(t) - Length of per-service data, 8 words not including header
(u) - Parameter ID, parameter 133 (Composed Ctot)
(v) - Composed Ctot flag byte
(w) - Composed Ctot length, 1 word not including header
(x) - Parameter ID, parameter 134 (Composed Dtot)
(y) - Composed Dtot flag byte
(z) - Composed Dtot length, 1 word not including header
(aa)- Parameter ID, parameter 135 (Composed Csum).
(bb)- Composed Csum flag byte
(cc)- Composed Csum length, 1 word not including header
(dd)- Parameter ID, parameter 136 (Composed Dsum).
(ee)- Composed Dsum flag byte
(ff)- Composed Dsum length, 1 word not including header

Word 20: Controlled-Load parameters
(gg - Per-Service header, service number 5 (Controlled-Load)
(hh)- Break bit
(ii)- Length of controlled-load data, 0 words not including header

4. Security Considerations

The message formatting and usage rules described in this note raise no security issues. The overall use of these rules to implement multiple qualities of service using RSVP and integrated services scheduling modules introduces a new security requirement; the need to control and authenticate access to enhanced qualities of service. This requirement is discussed further in [RFC 2205], [RFC 2212], and [RFC 2211]. [RFCRSVPMD5] describes the mechanism used to protect the integrity of RSVP messages carrying the information described here.

Appendix 1: Message construction rules

This section gives the rule used to generate the object formats of Section 3. It is a general wire format for encoding integrated services data objects within setup and management protocol messages. The format has a three-level structure:

- An overall message header carries a version number and message length. Providing this header in a standard format allows the same code library to handle data objects carried by multiple setup protocols.

- Per-service fragments carry information about a specific QoS control service, such as guaranteed [RFC 2212] or controlled load [RFC 2211]. Each per-service fragment carries one or more parameters. The set of parameters present in a fragment is determined by the needs of the protocol in use. Examples are given in Section 2.

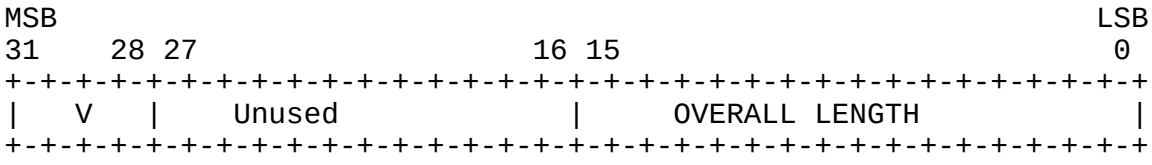
- Parameters are the actual data used to control or monitor a service. A parameter may be a single quantity such as an integer, or a composite data structure such as a TSpec. The parameters specific to a service are defined by the service specification. The available general parameters, with definitions shared by many services, are defined by [RFC 2215].

A1.1. Message Header

The 32-bit message header specifies the message format version number and total length of the message. The overall message must be aligned to a 32-bit boundary within the transport protocol's data packet. The message length is measured in 32-bit words *not including the word containing the header*. This is to lower the probability of an accidentally cleared word resulting in an infinite loop in the message parser.

The Message Header is represented by a 32-bit bitfield laid out as shown below and then encoded as an XDR unsigned integer. Encoding as an XDR unsigned integer is equivalent to converting the bitfield from the machine's native format to big-endian network byte order.

Message Header



V - Message format version; currently 0
OVERALL LENGTH - Message length in 32-bit words not including header

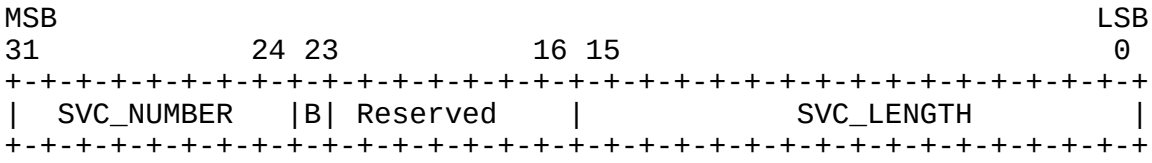
A1.2. Per-Service Data Header

The message header is followed by one or more service-specific data blocks, each containing the data associated with a specific QoS control service. Each service-specific data block begins with an identifying header. This 32-bit header contains the service number, a one-bit flag (the "break bit", because it indicates a break in the QoS control path) and a length field. The length field specifies the number of 32-bit words used to hold data specific to this service as a count of 32-bit words *not including the word containing the header*.

The break bit, if set, indicates that the service specified by the header was unsupported or unrecognized at some point in the message's path through the network. This bit corresponds to the general parameter NON_IS_HOP defined in [RFC 2215]. It is cleared when a message is first generated, and set whenever the message passes through an element that does not recognize the service_number in the per-service header.

The Per-Service Data Header is represented by a 32-bit bitfield laid out as shown below and then encoded as an XDR unsigned integer.

Per-Service Data Header



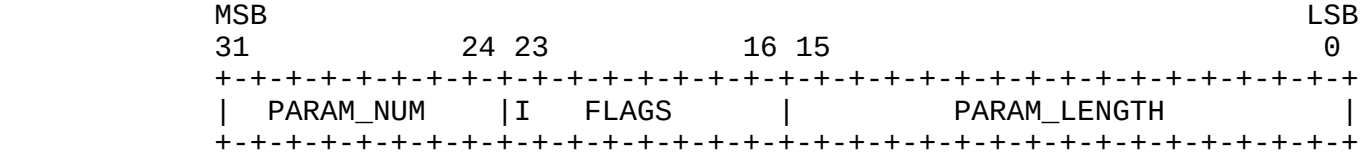
SVC_NUMBER - Service ID number (defined in service specification).
B - Break bit - service unsupported/break in path.
SVC_LENGTH - Service-specific data length in 32-bit words, not including header.

A1.3. Parameter Header

The per-service header is followed by one or more service parameter blocks, each identified by a Parameter Header. This header contains the parameter identifier (parameter number), the length of the data carrying the parameter's value, and a flag field. The data field(s) of the parameter follow. The parameter number, as well as the meaning and format of the data words following the header, are given by the specification which defines the parameter.

The Parameter Header is represented by a 32-bit bitfield laid out as shown below and then encoded as an XDR unsigned integer.

Parameter Header



PARAM_NUM - Parameter number (defined in service specification)
FLAGS - Per-parameter flags
PARAM_LENGTH - Length of per-parameter data in 32-bit words, not including the header word.

The following flags are currently defined in the FLAGS field:

I (bit 23) - INVALID

This flag indicates that the parameter value was not correctly processed at one or more network elements along a data path. It is intended for use in a possible future service composition scheme.

Other bits in the FLAGS field of the parameter header are currently reserved, and should be set to zero.

A1.4. Parameter Data

Following the Parameter Header is the actual data representing the parameter value. Parameter values are encoded into one or more 32-bit words using the XDR external data representation described in [RFC 1832], and the resulting words are placed in the message.

The document defining a parameter should provide an XDR description of the parameter's data fields. If it does not, a description should be provided in this note.

References

[RFC 2205] Braden, B., Ed., et. al., "Resource Reservation Protocol (RSVP) - Version 1 Functional Specification", RFC 2205, September 1997.

[RFC 2216] Shenker, S., and J. Wroclawski. "Network Element QoS Control Service Specification Template", RFC 2216, September 1997.

[RFC 2212] Shenker, S., Partridge, C., and R Guerin, "Specification of Guaranteed Quality of Service", RFC 2212, September 1997.

[RFC 2211] Wroclawski, J., "Specification of the Controlled Load Quality of Service", RFC 2211, September 1997.

[RFC 2215] Shenker, S., and J. Wroclawski, "General Characterization Parameters for Integrated Service Network Elements", RFC 2215, September 1997.

[RFCRSVPMD5] Baker, F., "RSVP Cryptographic Authentication", Work in Progress.

[RFC 1832] Srinivansan, R., "XDR: External Data Representation Standard", RFC 1832, August 1995.

Author's Address

John Wroclawski
MIT Laboratory for Computer Science
545 Technology Sq.
Cambridge, MA 02139

Phone: 617-253-7885
Fax: 617-253-2673 (FAX)
EMail: jtw@lcs.mit.edu