IPv6 Working Group Rahul Banerjee
Internet Draft Sumeshwar Paul Malhotra
Mahaveer M
BITS, Pilani (India)
February 2002

A Modified Specification for use of the IPv6 Flow Label for providing An efficient Quality of Service using a hybrid approach.

draft-banerjee-flowlabel-ipv6-qos-01.txt

Status of This Memo

Abstract

This memo suggests a modified specification for defining the 20-bit Flow Label field using a hybrid approach that includes options to provide IntServ as well as DiffServ based support for Quality of Service. It also compares various suggested approaches for defining the 20-bit Flow Label field in IPv6 Base Header based on RFC 2460 (December 1998) and draft-conta-ipv-flow-label-02.txt by Conta & Carpenter (July 2001). Addressing the IPv6-Multicast-QoS issues also Becomes possible as a consequence. The resultant mechanism is fully implementable and unambiguous as even the lower-level details have been worked out as may be required for real implementations.

Table of Contents

1. Introduction

IntServ and DiffServ present two alternative solutions of resolving QoS problems in the Internet. This paper talks about the design of Quality of Service (QoS) in IPv6 that provides support for IntServ and DiffServ Quality of Service. Though the IPv6 Base Header has a 20-bit Flow Label field for QoS implementation purposes, it has not yet been exploited. Few Internet drafts give various definitions of the 20-bit Flow Label in IPv6, each with its own advantages and disadvantages. This paper provides an analysis of these definitions and subsequently suggests a specification, which in view of the author can provide an efficient Quality of Service using a hybrid approach.

2. IPv6 Flows and Flow Label

As defined in [draft-conta-ipv6-flow-label-02.txt], a flow is a sequence of packets sent from a particular source, and a particular application running on the source host, using a particular host-to-host protocol for the transmission of data over the Internet, to a particular (unicast or multicast) destination, and particular application running on the destination host, or hosts, within a certain set of traffic, and QoS requirements.

The IPv6 Flow Label [RFC 2460] is defined as a 20-bit field in the IPv6 header which may be used by a source to label sequences of packets for which it requests special handling by the IPv6 routers, such as non-default quality of service or "real-time" service. According to RFC 2460, the nature of that special handling might be conveyed to the routers by a control protocol, such as RSVP, or by information within the flow's packets themselves, e.g., in a hop-by-hop option.

The characteristics of IPv6 flows and Flow Labels given in [RFC 2460] are rearranged as follows:

(a) A flow is uniquely identified by the combination of a source address and a non-zero Flow Label.

(b) Packets that do not belong to a flow carry a Flow Label of zero

(c) A Flow Label is assigned to a flow by the Flow's source node.

(d) New Flow Labels must be chosen (pseudo) randomly and uniformly from the range 1 to FFFFF hex. The purpose of the random allocation is to make any set of bits within the Flow Label field suitable for use as a hash key by routers, for looking up the state associated with the flow.

(e) All packets belonging to the same flow must be sent with the same source address, destination address, and Flow Label.

(f) If packets of flow include a Hop-by-Hop options header, then they all must be originated with the same Hop-by-Hop options header contents.

(g) If packets of a flow include a routing header, then they all must be originated with the same contents in all extension headers up to and including the routing header.

(h) The maximum's lifetime of any flow-handling state established along a flow's path must be specified as part of the description of the state-establishment mechanism, e.g., the resource reservation protocol or the flow-setup hop-by-hop option.

(i) The source must not reuse a Flow Label for a new flow within the maximum lifetime of any flow-handling state that might have been established for the prior use of that Flow Label.

3. Integrated Services Flows

The Integrated Services architecture [RFC 1633] defines a flow as an abstraction, which is a distinguishable stream of related datagrams that results from a single user activity and requires the same QoS.

The IntServ architecture supports services on per flow basis. The IntServ model uses Resource Reservation Protocol (RSVP) as the standard signaling protocol to provide QoS to application flows in the network. It offers three classes of service:

1. Best Effort Service (FCFS, meant for ordinary data: default).

2. Guaranteed Service (meant for Hard Real time requirements)
   - Known upper bound on delay.
   - Reliable (lossless) delivery for IP packets that conform to specification.
   - Guaranteed Bandwidth support.

3. Controlled Load service (meant for Soft Real time requirements)

As specified in [RFC 1633], the IntServ architecture defines a classifier:

> For the purpose of traffic control (and accounting), each incoming packet must be mapped into some class; all packets in the same class get the same treatment from the packet scheduler. This mapping is performed by the classifier. Choice of a class may be based upon the contents of the existing packet header(s) and/or some additional classification number added to each packet.
>
> A class might correspond to a broad category of flows, e.g., all video flows or all flows attributed to a particular organization. On the other hand, a class might hold only a single flow.

4. Differentiated Services Flow

The Differentiated Services architecture [RFC 2475] defines a flow or microflow as a single instance of an application-to-application flow of packets, which is identified by the source address, source port, destination address, destination port and protocol id (fields in the IP and host-to-host protocol headers).

Unlike IntServ, which offers 'Per-Flow-based' QoS support, the DiffServ offers 'Aggregate-Flow-based' QoS support. It has the potential to complement the IntServ (rather than replacing it).

As specified in [RFC 2475], the DiffServ architecture defines a classifier:

> as a mechanism that selects packets in a traffic stream based on the content of some portions of the packet header. The MF (Multi-Field) classifier selects packets based on the value of a combination of one or more header fields, such as source address, destination address, DS field, protocol ID, source port and destination port numbers, and other information.

In order to support the Flow Label, a Differentiated services IPv6 classifier definition should be added. This classifier would be a multi field classifier that would include at least the Flow Label and the source address as the IPv6 specification suggests.

According to Differentiated Services architecture [RFC 2475], the classification fields have values according to the Service Level Agreements (SLA) and Traffic Conditioning Agreements (TCA), (Service Level Specifications - SLS and Traffic Conditioning

Specification - TCS) which are contractual agreements between clients and the network service providers. The Flow Label based DiffServ MF classifier would allow the same model, and would rely on the Flow Label that is a field with a value or a range of values on which or service providers would have to agree on. These values will be reflected in SLAs, TCAs, SLSs and TCSs.

The potential advantage of the DiffServ model is a substantial reduction in router state and a simplification in router design and implementation. The potential drawback to the DiffServ model is that all flows in the same service aggregate may receive the same level of service. This may force flows with very different QoS requirements into the same service class.

5. Issues related with IPv6 Flow Label

According to RFC 1809, the IPv6 specification originally left open a number of questions, of which the following are important.

5.1 What should a router do with Flow Labels for which it has no state?

What should be the default action of the router on receiving a datagram with a non-zero Flow Label for which it has no state information?

Unknown Flow Labels may also occur if a router crashes and loses its state.

The IPv6 specification gives the following possible solutions to the above-mentioned problem.

1. The routers can ignore the Flow Label.

2. IPv6 datagram may carry flow setup information in their options.

In any case, it is clear that treating this situation as an error and, say dropping the datagram and sending an ICMP message, is inappropriate. Indeed, it seems likely that in most cases, simply forwarding the datagram as one would forward a datagram with a zero Flow Label would give better service to the flow than dropping the datagram.

There may be situation in which routing the datagram as if it’s Flow Label were zero might cause the wrong result, but these situations can be treated as the exceptions rather than the rule. It is also reasonable to handle these situations using options that indicate that if the flow state is absent, the datagram needs special handling. (The options may be Hop-by-Hop or only handled at some routers, depending on the flow's needs).

Finally, [RFC 1809] and the author's view suggest that the default rule should be that if a router receives a datagram with an unknown Flow Label, it treats the datagram as if the Flow Label is zero. As part of forwarding, the router will examine any hop-by-hop options and learn if the datagram requires special handling. The options could include simply the information that the datagram is to be dropped if the Flow Label is unknown or could contain the flow state the router should have.

5.2 Flushing old Flow Labels

How does an internetwork flush old Flow Labels?

The flow mechanism assumes that state associated with a given Flow Label is somehow deposited in routers, so they know how to handle datagrams that carry the Flow Label. A serious problem is how to flush Flow Labels that are no longer being used (stale Flow Labels) from the routers.

Stale Flow Labels can happen in a number of ways, even if we assume that the source always sends a message deleting a Flow Label when the source finishes using a Flow.

1. The deletion message may be lost before reaching all routers.

2. Furthermore, the source may crash before it can send out a Flow Label deletion message.

The mechanism suggested by [RFC 1809] is to use a timer. Routers should discard Flow Labels whose state has not been refreshed within some period of time. At the same time, a source that crashes must observe a quiet time, during which it creates no flows, until it knows that all Flow Labels from its previous life must have expired. (Sources can avoid quiet time restrictions by keeping information about active Flow Labels in stable storage that survives crashes). According to [RFC 1809], there are two options for refreshing the Flow Label and its state:

1. The source could periodically send out a special refresh message to explicitly refresh the Flow Label and its state.

2. The router could treat every datagram that carries the Flow Label as an implicit refresh or sources could send explicit refresh options.

The choice is between periodically handling a special update message and doing an extra computation on each datagram (namely noting in the Flow Label's entry that the Flow Label has been refreshed).

Based on the discussion mentioned above according to [RFC 1809], the authors of the document suggest the following approach as a solution to this problem:

1. The MRU (Most Recently Used) algorithm should be used for maintaining the Flow Labels. At any point of time, the most recently used Labels alone will be kept and the remaining should be flushed.

2. Before flushing a label, the router should send an ICMP message to the source saying that the particular label is going to be flushed. So the source should send a KEEPALIVE Message to the router saying not to flush the Flow Label in case the source requires the Flow Label to be used again. On the other hand, if the source agrees with the router to delete the Flow Label, it should send a GOAHEAD Message to the router. On receiving the GOAHEAD Message, the router immediately deletes the label for that particular source. These messages are also sent to all the intermediate routers, so that, those routers can as well flush the Flow Labels for that particular source.

3. In case, the router does not receive any consent from the source, it will re-send the ICMP message for at most two or three times. If the router does not receive any reply from the source, it can flush the particular Label assuming that the Flow Label was not important for the source or any other intermediate router. The intermediate routers will also delete that Flow Label as they didn't receive any message from the source. The policy of sending the ICMP message to the source two or three times ensures the proper behavior of the method of flushing Flow Labels in case of packet loss. This method assumes that the ICMP message would not be lost all the three times. Hence, if the router doesn't receive any reply from the source even after sending the ICMP message three times, it deletes the label.

5.3 Which datagrams should carry non-zero Flow Labels?

According to RFC 1809, following were some points of basic agreement.

1. Small exchanges of data should have a zero Flow Label since it is not worth creating a flow for a few datagrams.

2. Real-time flows must always have a Flow Label.

One option specified in [RFC 1809] is to use Flow Labels for all long-term TCP connections. The option is not feasible in the view of the authors as it will force all the applications on that particular connection to use the Flow Labels which in turn will force routing vendors to deal with cache explosion issue.

5.4 Mutable/Non-mutable IPv6 Flow Label

According to [draft-conta-ipv6-flow-label-02.txt], another issue is whether the Flow Label should be mutable or non-mutable, that is it should be read only for routers or not?

Agreeing with the suggestions of [draft-conta-ipv6-flow-label-02.txt], the Flow Labels should be non-mutable because of the following reasons:

1. Using mutable Flow Labels would require certain negotiation mechanism between neighboring routers, or a certain setup through router management or configuration, to make sure that the values or the changes made to the Flow Label are known to all the routers on the path of the packets, in which the Flow Label changes. On the other hand, the non-mutable Flow Labels certainly have the advantage of the simplicity implied by such a characteristic.

2. A mutable Flow Label characteristic goes against the IPv6 specification of the Flow Label explained in section 2 and the IPv6 Flow Label characteristics explained in the coming sections.

5.5 Using random numbers in setting the IPv6 Flow Label

The IPv6 specification specifies the requirement of pseudo-randomness in setting the value of a Flow Label as it can be used as hash key by routers for flow lookup.

However, a random value in the header introduces unpredictability of the field. Since predictability is a necessary condition for a deterministic behavior, network operators may require that packets of a flow always have the same IPv6 content. Random values in the IPv6 Flow Label certainly breaks this requirement. So supporting the arguments given in [draft-conta-ipv6-flow-label-02.txt], the authors of this document suggest the IPv6 specification of having a random number in the Flow Label field to be relaxed.

5.6 Filtering using Flow Label

If, at all, any filtering has to be done based on the Flow Label field in the IPv6 header, the expectation is that the IPv6 Flow Label field carries a predictable or well-determined value. This is not the case if the Flow Label has randomly chosen values.

Again, supporting the arguments given in [draft-conta-ipv6-flow-label-02.txt], the authors of this document suggest that the problem of not being able to configure load-filtering rules, which are based or are including the Flow Label, can be resolved by relaxing IPv6 specification of having a random number in the Flow Label field.

6. Various approaches in defining IPv6 Flow Label format

This section discusses the various already suggested approaches for defining the 20-bit Flow Label. It discusses the advantages and disadvantages of these approaches. Finally it tells about accepting or rejecting these approaches and includes the accepted approaches (with modifications wherever required) in the final definition of the Flow Label discussed in the next section.

6.1 First approach [draft-conta-ipv6-flow-label-02.txt]

Following format can be used for the Flow Label:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0 |   Pseudo - Random value                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 1 |   DiffServ IPv6 Flow Label              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The DiffServ IPv6 Flow Label is a number that is constructed based on the Differentiated services "Per Hop Behavior Identification Code".

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 1 |   Per Hop Behavior Ident. Code|  Res. |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The "Res" bits are reserved.

The PHB ID is either directly derived from a standard differentiated services code point, or it is an "IANA Assigned Value".

Advantages:

Preserves compatibility with the random number method of selecting a Flow Label value defined in IPv6 specification.

Captures the differentiated services treatment intended to be applied to the packet.

Unlike the value of the traffic class field, it is not locally mapped and hence suitable for use in an end-to-end header field.

Disadvantages:

It captures less information than the port number and protocol number normally used in multi field classifier.

6.2 Second Approach [draft-conta-ipv6-flow-label-02.txt]

DiffServ with multi field classifier can be used in a more efficient and practical manner as an alternative to IntServ and RSVP. The Flow Label classifier is basically a 3-element tuple - source and destination address and IPv6 Flow Label.

The classifier can be defined in any of the following two ways:

C = (SA, SAPrefix, DA, DAPrefix, Flow Label).

C` = (SA, SAPrefix, DA, DAPrefix, Flow Label min: Flow Label max).

Incoming packet header (SA, DA, Flow Label) is matched with classification rules table entry C or C`.

Advantages:

Helps the IPv6 Flow Label to achieve, as it is supposed, in a more efficient processing of packets in QoS engines in IPv6 forwarding devices.

Disadvantages:

When packets are transmitted, the end nodes have to force the correct Flow Label in the IPv6 headers of outgoing packets or the first hop routers have to do this job. To accomplish these rules, these routers will be configured with MF classifiers. This puts extra computations to be done by the routers.

6.3 Third approach [draft-conta-ipv6-flow-label-02.txt]

Includes the algorithmic mapping of the port numbers and protocol into the Flow Label. It reserves 12 bits for the port number and 8 bits for the protocol.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Server port number   | H-to-H protocol|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Advantages:

Classification rule is 5 or 6 element tuple format of a DiffServ MF classifier, containing the source and the destination address, the source and the destination ports, the host-to-host protocol. So no new classification rule format is needed.

Disadvantages:

It cannot differentiate among multiple instances of the same application running on the same two communication end nodes.

The reduced number of bits (12 out of 16) limits the value of ports. 12 bits can represent only the "IANA well-known ports", that is from 1 to 1023 and a subset of "IANA registered ports", that is from 1024 to 4095. Registered ports have values between 1024 and 65535.

6.4 Fourth approach [draft-conta-ipv6-Flow-label-02.txt]

The field occupied by host-to-host protocol could be reduced to 1, as TCP and UDP are the only well known protocols.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    TCP Server port number      |Res |0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    UDP Server port number      |Res |1|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The "Res" bits are reserved.

The "TCP Server Port Number" or "UDP Server Port Number" is the 16-bit port number assigned to the server side of the client/server application.

Advantages:

Again the classification field is a 5 or 6 element tuple. So no new classification rule is needed.

This approach keeps 16 bits for the port number so that all the "IANA well-known ports" and "IANA registered ports" can be accommodated in these 16 bits.

Disadvantages:

This approach, too, cannot differentiate among multiple instances of the same application running on the same two communication end nodes.

Reserving only 1 bit for the protocol field in the Flow Label restricts the use of any protocol other than TCP and UDP.

6.5 Fifth approach [draft-conta-ipv6-flow-label-02.txt]

Header length format:

Another possible solution is to store the length of IPv6 headers length that is the length of the IPv6 Base Headers and IPv6 extension headers preceding the host-to-host or transport header. The length of IPv6 headers in the Flow Label value would provide the information, which a DiffServ QoS engine classifier could use to locate and fetch the source and destination ports and apply those along with the source and destination address and host-to-host protocol from the Flow Label, to match the source and destination address, the source and destination ports and the protocol identifier elements of a DiffServ MF classifier.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Length of IPv6 headers| H-to-H protocol|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Advantages:

"Length of IPv6 headers" allows skipping the IPv6 headers to access directly the host-by-host header for other purposes. This format is useful for classifying packets that are not TCP or UDP, and have no source and destination ports.

Disadvantages:

IPv6 header does not include "Total Headers Length" field. So introducing this new field in the Flow Label puts extra computation to be done that may result in the processing delays.

Including "Length of IPv6 headers" in the Flow Label does not carry any significance in case ESP is used for IP Security.

This approach is discarded in this paper because of the reasons given above. Again, it does not carry any direct advantage in keeping the "Length of IPv6 headers" in the Flow Label.

7. A modified specification for the IPv6 Flow Label and related implementation mechanism: A hybrid approach suggested by this work

7.1 Overview

This section specifies a modified Flow Label for IPv6 for providing efficient Quality of Service that utilizes the results of some of the works referred above, extends some of the suggested mechanisms and finally presents an integrated hybrid approach.

7.2 Definition of first three bits of the Flow Label

The hybrid approach suggested in this document includes various approaches already mentioned in the previous section. The 20-bits of the Flow Label should be defined in an appropriate manner so that various approaches can be included to produce a more efficient hybrid solution. Hence, for this purpose, the first three bits of the IPv6 Flow Label are used to define the approach used and the next 17 bits are used to define the format used in a particular approach.

Following is the bit pattern for the first 3 bits of Flow Label that defines the type of the approach used:

0 0 0 Default.

0 0 1 A random number is used to define the Flow Label.

0 1 0 The value given in the Hop-by-Hop extension header is used instead of the Flow Label.

0 1 1 Multi Field Classifier is used.

1 0 0 A format that includes the port number and the protocol in the Flow Label is used.

1 0 1 A new definition explained later in this section is used.

1 1 0 Reserved for future use.

1 1 1   Reserved for future use.

This definition of Flow Label includes IntServ and DiffServ and includes above mentioned approaches for defining the Flow Label. A further explanation of these options is provided in the remaining part of the document. The default value specifies that the datagram does not need any special Quality of Service.

7.3 Defining the remaining 17 bits of the IPv6 Flow Label

The remaining 17 bits of the IPv6 Flow Label are defined based on the approach defined in the first three bits of the Flow Label as mentioned in the previous section.

7.3.1 Random Number

As specified in IPv6 specification, a random number can be used to define the Flow Label. Here a 17-bit random number can be used. The random numbers can be generated in the range from 1 to 1FFFF. The advantages and disadvantages of using a random number are already discussed in the previous section. Keeping the IPv6 specifications in mind, the authors of this document believe that the random number can be used as one of the approaches. As other approaches are defined in the Flow Label, this random number approach may not be used whenever not feasible or efficient to do so.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 0 1|     Pseudo - Random value       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

7.3.2 Using Hop-by-Hop extension header

As defined in [draft-banerjee-ipv6-quality-service-01.txt], Hop-by-Hop extension header can be used for defining the Flow Label in case IntServ is used. In this case the value in the 20-bit Flow Label is ignored. The modified Hop-by-Hop extension has been suggested and defined in the reference [draft-banerjee-ipv6-quality-service-01.txt]. In that document, the Hop-by-Hop extension header has been defined to be used with IntServ. This mechanism applies to define for DiffServ as well.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 1 0|         Don't care              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

7.3.3 Using Multi Field Classifier

As mentioned in the previous section, DiffServ with MF classifier can be used. In that case the format of the Flow Label will be as shown below:

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 1 1|     DiffServ IPv6 Flow Label    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

As suggested in [draft-conta-ipv6-flow-label-02.txt], this Flow Label can be a PHB ID (Per Hop Behavior Identification Code). In this case, 16-bit PHB ID will be used and the remaining 1 bit is reserved for future use.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 1 1|  Per Hop Behavior Ident. Code |R|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

'R' is reserved.

Packets coming into the provider network can be policed based on the Flow Label. The provider, based on the SLAs, SLSs, TCAs, TCSs agreed with the client, configures MF classifiers. This document specifies the classifier which is little different from the one suggested in the [draft-conta-ipv6-flow-label-02.txt]. The classifier looks like:

C = (SA/SAPrefix, DA/DAPrefix, Flow-Label).

Or

C` = (SA/SAPrefix, DA/DAPrefix, Flow-Label-Min: Range).

The range here specifies the difference between the maximum and the minimum Flow Label. The significance of using the range instead of Maximum Flow Label is the reduced number of bits. Definitely the difference between the two values can be specified in a lesser number of bits as compared to the value itself.

Flow-Label-Classifier:

```
IPv6SourceAddressValue/Prefix:  10:11:12:13:14:15:16:17:18::1/128
IPv6DestAddressValue/Prefix:    1:2:3:4:5:6:7:8::2/128
IPv6 Flow Label:                50
```

Or

```
IPv6SourceAddressValue/Prefix:  10:11:12:13:14:15:16:17:18::1/128
IPv6DestAddressValue/Prefix:    1:2:3:4:5:6:7:8::2/128
IPv6 Flow Label:Range:          10:20
```

Incoming Packet header (SA, DA, Flow Label) is matched against classification rules table entry (C or C`).

7.3.4 Using the Port Number and the Protocol

This approach already discussed in this document in the previous section defines Flow Label by including the server port number and the host-to-host protocol. The "Server Port Number" is the port number assigned to the server side of the client/server applications. As specified in [draft-conta-ipv6-flow-label-02.txt], this approach reserves 16 bits for the port number and 1 bit for the protocol with the remaining bits reserved for the future use.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 0 0|    TCP Server port number     |0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 0 0|     UDP Server port number    |1|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

But this approach puts the restriction on the protocol to be used by any application.

As most of the application seeking Real-time service use TCP or UDP as the transport layer protocol, this approach would work fine in most of the cases. In case the application requires to use any other host-to-host protocol, the other methods for specifying the Flow Label, discussed in this section can be used. Anyhow, this method for specifying the port number and the protocol can be exploited further in the future to remove any limitations.

7.3.5 A new structure and mechanism for the use of the Flow Label

This section describes an innovative approach to define the 20-bit Flow Label field in IPv6 header. By the optimal use of the bits in the Flow Label, this approach includes various Quality of Service parameters in the IPv6 Flow Label that may be requested by any application. The various Quality of Service parameters are:

1. Bandwidth
2. Delay or Latency
3. Jitter

4. Packet Loss
5. Buffer Requirements

As packet loss and the jitter are often desired to be of minimum value by any application, these two parameters may not be defined in the Flow Label field itself. Instead, if needed, the Hop-by-Hop EH space can be effectively used to specify these parameters. Bits thus saved in the FL can be effectively used for more demanding purposes.The Quality of Service parameters that are to be included in the Flow Label are:

1. Bandwidth (to be expressed in multiples of kbps).
2. Delay (to be expressed in nanoseconds).
3. Buffer requirements (to be expressed in bytes).

As there are only 17 bits left, the optimal use of the bits is very important so as to obtain the maximum information out of those 17 bits. The first bit out of these 17 bits is used to differentiate between the hard real time and soft real time applications. This bit is set to 0 for soft real time applications and it is set to 1 for hard real time applications.

Soft Real time applications:

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 0 1|0|      Flow Label                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

This service is meant for RTT (Real Time Tolerant) or soft real time applications, which have an average bandwidth requirement and an intermediate end-to-end delay for an arbitrary packet. Even if the minimum or maximum values specified in the Flow Label are not exactly met, the application can afford to manage with the QoS provided. These RTI applications demand weak bounds on the maximum delay over the network.

Hard Real time applications:

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 0 1|1|      Flow Label                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

This service is meant for RTI (Real Time Intolerant) or hard Real Time applications, which demand minimal latency and jitter. For example, a multicast real time application (videoconferencing). Delay is unacceptable and ends should be brought as close as possible.

For this videoconference (DTVC) case, the required resource reservations are

a. Constant bandwidth for the application traffic.
b. Deterministic Minimum delay that can be tolerated.

These types of applications can decrease delay by increasing demands for bandwidth. The minimum or maximum values specified in the Flow Label have to be exactly met for these kind of applications.

After keeping one bit for Hard/Soft real time applications, we are left with 16 bits for defining the Flow Label. The remaining part of this section discusses how to represent the values of bandwidth, delay and buffer requirements.

1. Bandwidth

This definition specifies 6 bits out of the 16 bits to be used for specifying the bandwidth value. The application can demand for a minimum or a maximum value of bandwidth. So one bit out of these 6 bits is used for specifying whether the application is asking for a minimum value of bandwidth or a maximum.

0 - minimum expected value is specified.
1 - maximum expected value is specified.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 0 1|0|      Flow Label                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

In the above bit sequence, the application uses this new definition for defining the Flow Label, as described by the first 3 bits. The application is a hard real time, as evident by the 4th bit. It asks for a minimum bandwidth of value that will be described in the next few bits.

The 5 bits for the bandwidth can be exploited in two ways as shown below:

Approach 1:

This approach uses a simple formula to calculate the bandwidth from the five bits. The following values of bandwidths can be obtained for various bit-sequences.

00000 - 32 kbps
00001 - 64 kbps
.
.
.
00111 - 4 mbps
.
.
.
01111 - 1 gbps
.
.
.
11111 - 64 tbps

The formula used here to calculate the bandwidth in decimal from the bit pattern is:

Bandwidth (in decimal) = 2^B * 32.

Where B is the decimal equivalent of the bandwidth specified in 5 bits.

Approach 2:

This approach uses a hashed lookup table that maps the value mentioned in the bandwidth field of the Flow Label to the value already defined in the lookup table. These values have to be universally accepted and uniformly defined in all the routers and end-nodes.

In the opinion of the authors, using first approach will result in saving the time for lookup in providing the quality of service. In event of the requirement of certain intermediate values, the second approach could be used. However whichever be the case, it is to be recommended explicitly in this specification-field. This option may be indicated by an all-zero bit-string that in turn might indicate that the Look-up Table is embedded in the Hop-by-Hop EH Payload area as appended to the standard defined payload. [draft-banerjee-ipv6-quality-service-01.txt]

2. Buffer Requirements

This definition specifies 6 bits out of the 16 bits to be used for specifying the buffer value.

00000 - 512 bytes
00001 - 1 kbytes
.
.
.

00111 - 64 kbytes
.
.
.
01111 - 16 mbytes
.
.
.
11111 - 1 tbytes

The formula used here to calculate the buffer in decimal from the bit pattern is:

Buffer (in decimal) = 2^B * 512.

Where B is the decimal equivalent of the buffer specified in 5 bits.

3. Delay

This definition specifies 5 bits out of the 16 bits to be used for specifying the delay value. The application can tolerate a specified value of delay. So the five bits left for the delay value can be used in the following manner:

00000 - 4 nanoseconds
00001 - 8 nanoseconds
.
.
.
01000 - 1 microseconds
.
.
.
11111 - 8 seconds

The formula used here to calculate the buffer in decimal from the bit pattern is:

Delay (in decimal) = 2^B * 4 nanoseconds.

Where B is the decimal equivalent of the delay specified in 5 bits.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 0 1|1|0|0 0 0 0 1|0 0 0 0 1|0 0 0 0 1|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The above bit pattern tells that the application is a hard real time application. It asks for a minimum bandwidth of 64 Kbps at any time, a buffer requirement of 1 kilobyte and can tolerate a minimum delay of 8 nanoseconds.

8. Possible data structures required for the implementation of the above design (at the router).

Any router that tries to implement QoS maintains a QoS routing table and keeps track of the QoS available to each destination through the required number of hops. [RFC 2676]. Apart from this table, the router needs to keep track of the allotted QoS to each and every flow. This table is the ALLOTTED_QOS_TABLE.

1. Defining the different approaches.

```
enum MODEL_ID {
  RANDNUM=1,        // the random number method
  HOPTBYHOP=2,      // the hop-by-hop extension header method
  MFCLASS=3,        // the multi-field classifier
  PORT_PROT=4,      // port/protocol method
  HYBRID=5          // the hybrid approach
};
```

2. Defining the different Resource Identifiers.

```
enum RES_ID {
  BANDWIDTH=0,      // bandwidth requirement
  DELAY=1,          // delay requirement
  BUFFER=2,         // buffer requirement
};
```

3. Defining the value of the resource.

```
typedef unsigned int RES_VAL;

struct RESOURCE {
  RES_ID res_identifier; //the identifier of the resource
  RES_VAL res_value;     //the 32 bit value of the resource
};
```

4. Defining the Quality of Service.

```
struct QOS_INFO {
  MODEL model_id;
  RESOURCE resource;
};
```

5. Defining the port/protocol and the flow label.

```
struct port_protocol {
  unsigned port;
  unsigned protocol;
};

union format {
    unsigned flowlabel;
    struct port_protocol port_prot;
};
```

6. Defining the packet information.

```
 struct PACKET_INFO {
    struct sockaddr_in6 src_addr;
    struct sockaddr_in6 dest_addr;
    union format format_value;
  };
```

7. Defining the Alloted QoS table.

```
struct ALLOTED_QOS_TABLE {
  struct PACKET_INFO packet;
  struct QOS_INFO qos;
};
```

9. Overview of the whole design.

This section describes the whole process by taking an example. Consider any application (like Videoconferencing or Video/Audio on Demand) that needs some specified QoS.

9.1 Function of the Source

The application specifies the desired QoS and the Flow Label Field in the IPv6 header is filled based on the QoS asked by the application. The application has the flexibility of specifying which format it wants to use for getting the desired QoS. It can specify any of the formats described earlier in this document. The packet is then put on the network and it reaches the intermed- -iate routers.

9.2 Function of each relevant intermediate router

9.2.1 Initial Processing (Checks for default service)

It gets the format used by the packet by reading the first three bits of the Flow Label. In case the first three bits are 000 or 110 or 111, it represents the default service. No specific treatment required for this particular packet. In this case no further processing of the packet is required and the default QoS is provided to the packet. If the value given in the first three bits is 010, no further processing is done and the router knows that the required QoS is specified in the hop-by-hop extension header.

9.2.2 Searching for the entry (In case of non-default service)

1. The ALLOTTED_QOS table is searched based on the source address.
2. If an entry is found, then for that particular source, a search is made based on the PACKET_INFO structure defined above. If all the information stored exactly matches with the information contained in the incoming packet, the IPv6 packet is processed so that the reserved QoS is met.

9.2.3 New Entry

1. If an entry is not found, a new entry is made in the ALLOTTED_QOS table for the source and further processing of this new entry is done as follows.
2. All the relevant structures defined above are filled based on the information contained in the packet. Information about the packet is stored in the PACKET_INFO structure.
3. It reads the desired QoS from the packet’s header. If the format specifies that a random number is used in the Flow Label field, it reads the RANDOM_NUMBER table and searches for the given random number in the packet. It reads the specified QoS from the table and maintains that in the QOS_INFO structure after updating the RESOURCE structure. It then moves onto step 6.
4. If the format specifies that PHB ID is used in the Flow Label field, it reads the PHB_ID_TABLE and searches for the given ID. It reads the specified QoS from the table and maintains that in the QOS_INFO structure after updating the RESOURCE structure. It then moves onto step 6.
5. If the value in the Flow Label field specifies that the PORT/PROTOCOL field is used in defining the QoS required by the packet, it fills the RESOURCE structure and the QOS_INFO structure and moves onto step 6.
6. It then checks with the QoS Routing table, to find out if the desired QoS is possible to be provided to the packet. If yes, it updates the new entry in the ALLOTTED_QOS table in the memory or else this entry is removed.
7. If any relevant router en-route is not able to guarantee the requested QoS, an ICMPv6 message is sent to the source and the other routers (that had guaranteed the QoS) are also notified of the same so that they delete the corresponding entry from their QoS tables.

This process happens at all the intermediate routers between the source and the destination.

10. Security Considerations

The specifications of this draft don't raise any new security issues as the Flow Label field in the IPv6 header cannot be encrypted because of the known reasons. If encrypted, each in between router has to decrypt the header for providing the required QoS to the packet. As the QoS specification requires minimum delay for the packet, decrypting each packet's header at each router will not be a good idea because of the time required in processing the packet.

11. Conclusion

This report has dealt extensively with all the suggested formats for defining the 20-bit IPv6 Flow Label and finally has suggested a hybrid approach for efficiently defining the 20-bit IPv6 Flow Label. The emphasis of this work is to result into a practically acceptable specification that could be effectively used for a reasonably long period of time for implementing IPv6 Quality of Service that so far has been elusive in absence of a clear, verifiable and complete specification. A separate ID is under preparation specifically building upon these specifications so as to explicitly address the scalabilty issues related to the IPv6-Multicast-QoS.

Acknowledgements

Authors acknowledge technical inputs and support from the members of the "Project IPv6@BITS" at the Birla Institute of Technology and Science, Pilani, India, Dr. Latif Ladid of Ericsson Telebit, (Luxembourg); Dr. Torstern Braun of University of Bern (Switzerland); Dr. Pascal Lorenz of I.U.T. at the University of Haute Alsace, Colmar (France); Dr. S. Rao of Telscom A.G. (Switzerland); Dr. Bernardo Martinez of Versaware Inc. (Spain); Juan Quemada of UPM, Madrid (Spain) and Dr. Merce and Dr. Paulo at the EC and Dr. Zoubir Mammeri of IRIT (France).

Authors gratefully acknowledge the works of many dedicated brains at the IETF and elsewhere, sections or extracts of which have helped us to shape this document.

References

[RFC 2460]   S. Deering and Bob Hinden, “The Internet Protocol Specification”, RFC 2460, Internet Protocol version 6 Specification.

[RFC 1809]   C. Partridge, RFC 1809, "Using the Flow Label Field in IPv6".

[RFC 2676]   RFC 2676, QoS Routing Mechanisms and OSPF Extensions.

[RFC 1633]   RFC 1633, Integrated Services in the Internet Architecture: an overview.

[RFC 2475]   RFC 2475, An Architecture for Differentiated Services.

[RFC 2676]   RFC 2676, QoS Routing Mechanisms and OSPF Extensions.

References to the work in progress

[draft-banerjee-ipv6-quality-service-01.txt]   Rahul Banerjee, N. Preethi, M. Sethuraman, "Design and Implementation of the Quality-of-Service in IPv6 using the modified Hop-by-Hop Extension header - A Practicable Mechanism".

[draft-conta-ipv6-flow-label-02.txt]   A. Conta, B. Carpenter, "A proposal for the IPv6 Flow Label".

[NGNI-MMI-QoS: D1 and D2] Rahul Banerjee (BITS), Juan Quemda (UPM), P. Lorenz (UHA), Torsten Braun (UoB), Bernardo Martinez (Versaware): "Use of Various Parameters for Attaining QoS in IPv6-based Multimedia Internetworks", Nov. 15, 2001 and Feb. 2002 readily available at the URL: http://ipv6.bits-pilani.ac.in/ngni/.

Disclaimer

The views and specification here are those of the authors and are not necessarily those of their employers. The authors and their employers specifically disclaim responsibility for any problems arising from correct or incorrect implementation or use of this specification.

Author Information

Rahul Banerjee
3256, Centre for Software Development
BITS, Pilani - 333031
Rajasthan, India.

Phone: +91-159-7645073 Ext. 335
Email: rahul@bits-pilani.ac.in

Full Copyright Statement