ConEx:
A Research Tool for IPv6 Congestion Control

29 November 2010

John Leslie

Overview

Review IPv6 (many folks get it wrong!)

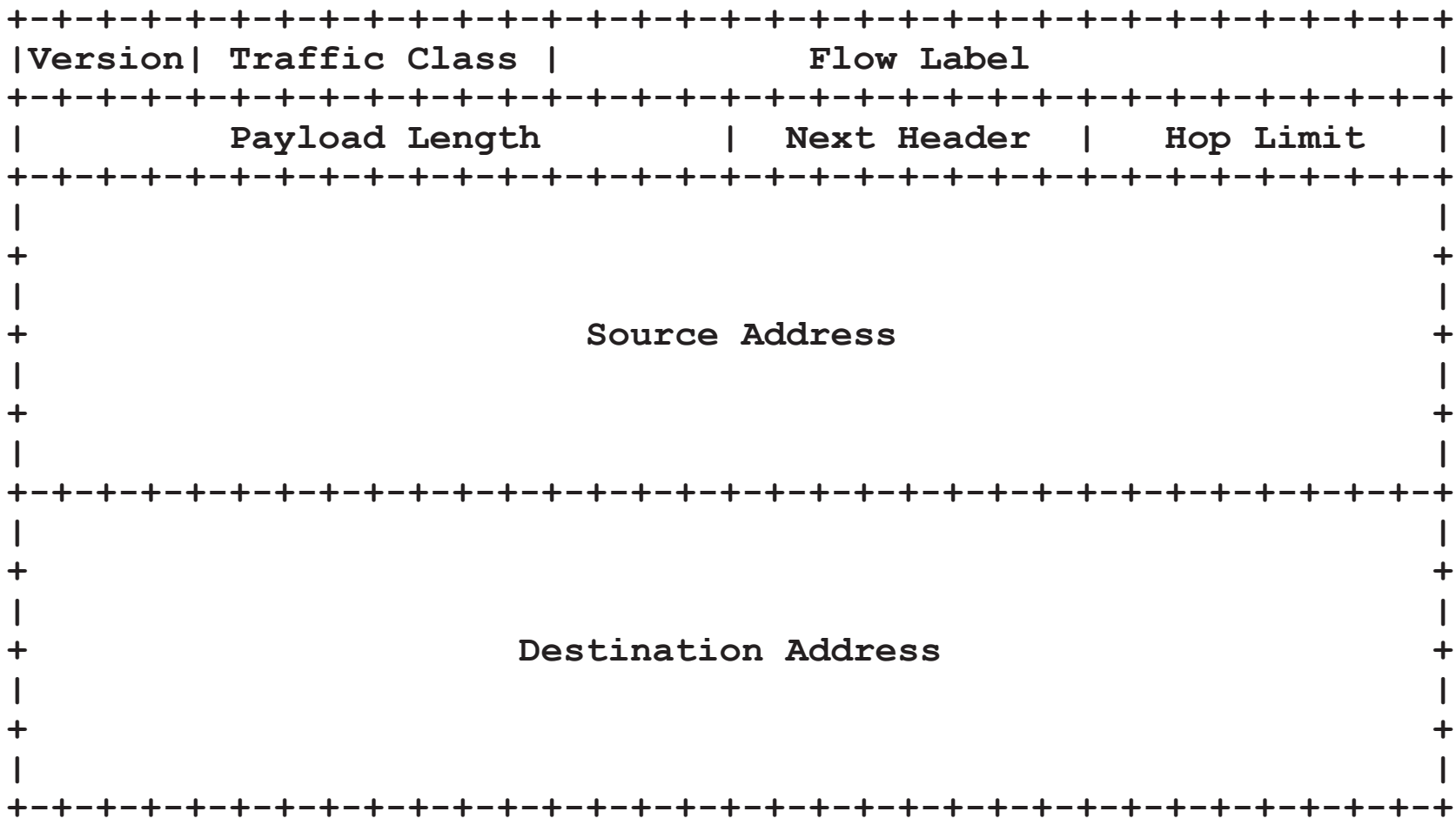
The “abstract” ConEx Mechanism

Fields Available in IPv6

What do we want from a Research Tool?

IPv6 Header

RFC 2460 Section 3. IPv6 Header Format



Header Fields

Version	4-bits	Internet Protocol version number = 6
Traffic Class	8-bits	traffic class field. See section 7
Flow Label	20-bits	flow label. See section 6
Payload Length	16-bits	unsigned integer: length of the IPv6 payload (rest of the packet following this IPv6 header) in octets
Next Header	8-bits	selector: Identifies the type of header immediately following the IPv6 header. *See IANA.org: Assigned Internet Protocol Numbers*
Hop Limit	8-bits	unsigned integer: decremented by 1 by each node that forwards the packet. The packet is discarded if Hop Limit is decremented to zero.
Source Address	128-bits	address of the originator of the packet
Destination Address	128-bits	address of the intended recipient of the packet (possibly not the ultimate recipient, if a Routing header is present)

“Next Header”

RFC 2460 Section 4. IPv6 Extension Headers:

Optional internet-layer information is encoded in separate headers placed between the IPv6 header and the upper- layer header. Each extension header is an integer multiple of 8 octets long, in order to retain 8-octet alignment for subsequent headers.

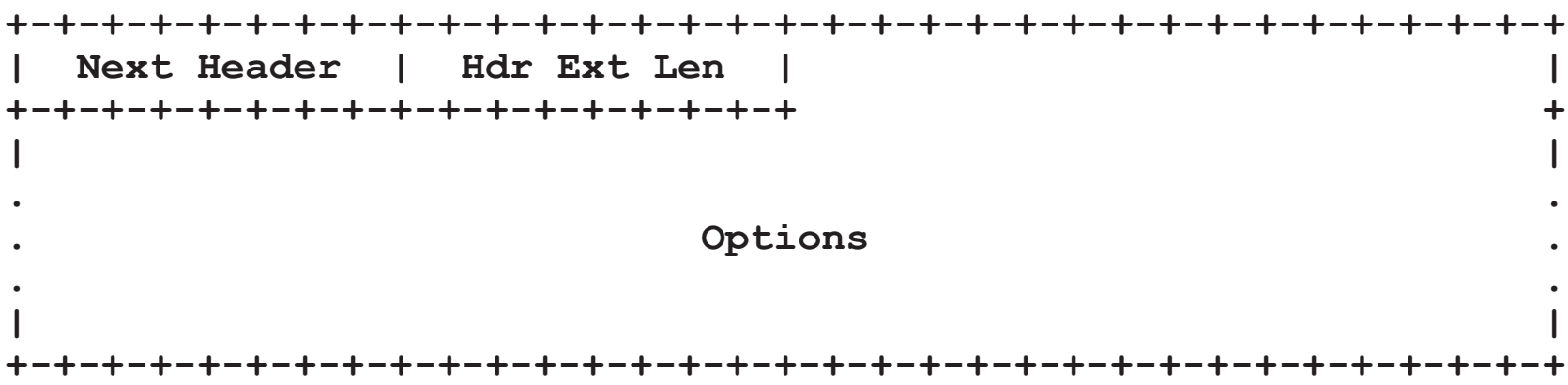
Except for Hop-by-Hop Headers, extension headers are not examined or processed by any node along a packet’s delivery path, until the packet reaches the node (or each of the set of nodes, in the case of multicast) identified in the Destination Address field of the IPv6 header.

Extension headers must be processed strictly in the order they appear in the packet; a receiver must not, for example, scan through a packet looking for a particular kind of extension header and process that header prior to processing all preceding ones.

If, as a result of processing a header, a node is required to proceed to the next header but the Next Header value in the current header is unrecognized by the node, it should discard the packet and send an ICMP Parameter Problem message to the source of the packet. The same action should be taken if a node encounters a Next Header value of zero (Hop-by-Hop) in any header other than an IPv6 header.

Hop-by-Hop Options Header

The Hop-by-Hop Options header is indicated by the value zero in the Next Header field of the IPv6 header. It must be examined and processed by every node along a packet's delivery path, including the source and destination nodes. It must be the first Extension Header in the IPv6 packet, and has the following format:



Next Header 8-bits selector: type of header immediately following this header

Hdr Ext Len 8-bits unsigned integer: length of this header in 8-octet units, not including the first 8 octets.

Options (variable) Contains one or more TLV-encoded options, as described in section 4.2.

Options within Options Headers

RFC2460 section 4.2: Options:
The Hop-by-Hop Options header and the Destination Options header carry a variable number of type-length-value (TLV) encoded "options", of the following format:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+- - - - - - - - -
|  Option Type  |  Opt Data Len |  Option Data
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+- - - - - - - - -
```

Option Type 8-bits identifier of the type of option

Opt Data Len 8-bits unsigned integer: length of this Option Data, in octets

Option Data (variable) Option-Type-specific data

There are two padding options used when necessary to align subsequent options and to pad out the containing header to a multiple of 8 octets in length. Option Type 0 is reserved for a single octet of padding (with the length and data omitted); Option Type 1 is reserved for two or more octets of padding (with the length field used).

Options within a header must be processed strictly in the order they appear in the header; a receiver must not, for example, scan through the header looking for a particular kind of option and process that option prior to processing all preceding ones.

Option Type Encoding

Option Type identifiers are internally encoded with their highest-order two bits specifying the action that must be taken if the processing IPv6 node does not recognize the Option Type:

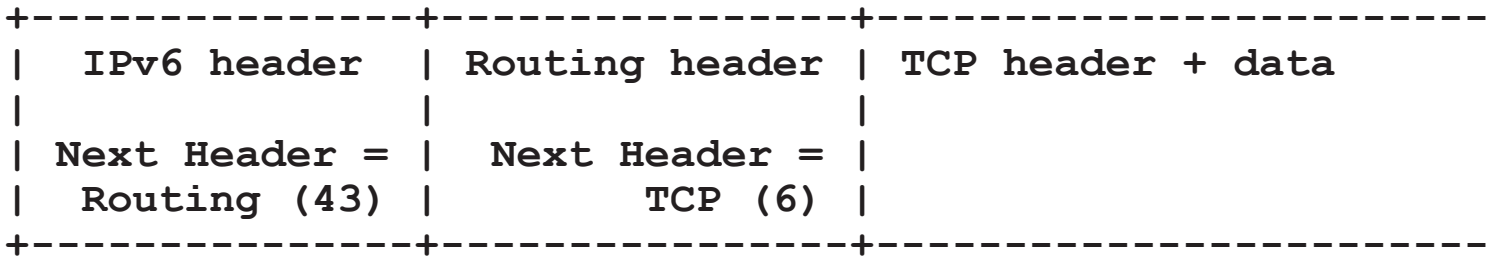
00 — skip over this option and continue processing the header.
01 — discard the packet.
10 — discard the packet and, regardless of whether or not the packet's Destination Address was a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.
11 — discard the packet and, only if the packet's Destination Address was not a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.

The third-highest-order bit of the Option Type specifies whether or not the Option Data of that option can change en-route to the packet's final destination.

0 — Option Data does not change en-route
1 — Option Data may change en-route

Appendix B of RFC2460 contains formatting guidelines for designing new options.

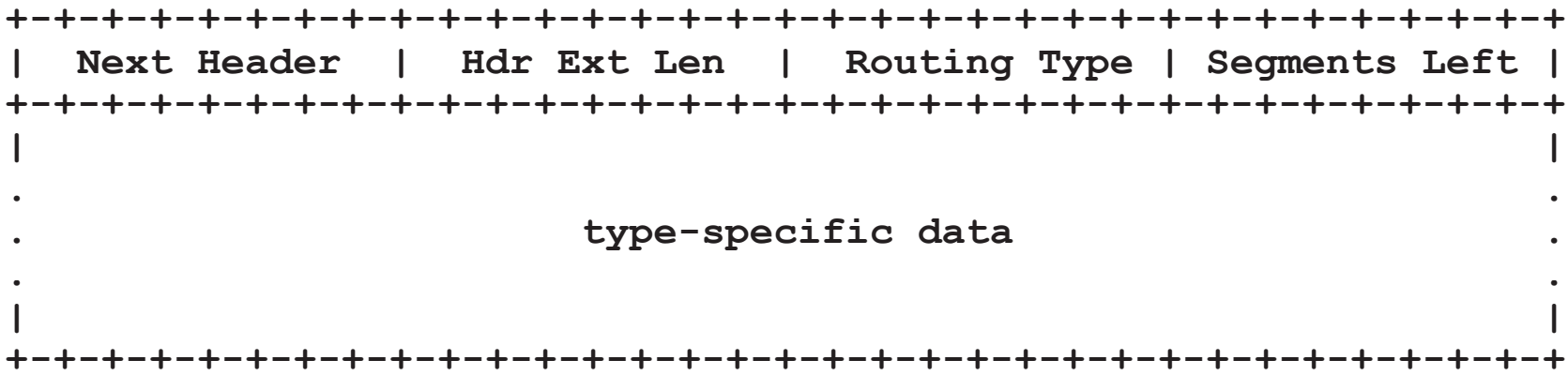
Routing Header



May be set by an IPv6 source to list intermediate nodes to be "visited" on the way to a packet's destination — similar to IPv4's Loose Source and Record Route option.

Not examined until it reaches the Destination Address of the IPv6 header. In that node, the Routing header module is invoked.

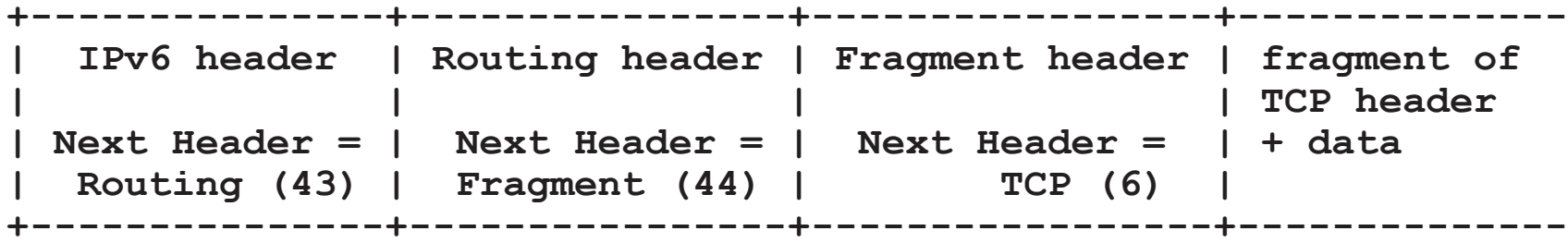
Routing Header Detail



Routing Type 8-bits identifier of a particular Routing header variant. Type zero is defined and deprecated

Segments Left 8-bits unsigned integer. Number of route segments remaining, i.e., number of explicitly listed intermediate nodes still to be visited before reaching the final destination.

Fragment Header



Only the Source may insert a Fragment Header. No router is allowed to fragment a packet too large for the MTU of the next hop: instead it must drop the packet.

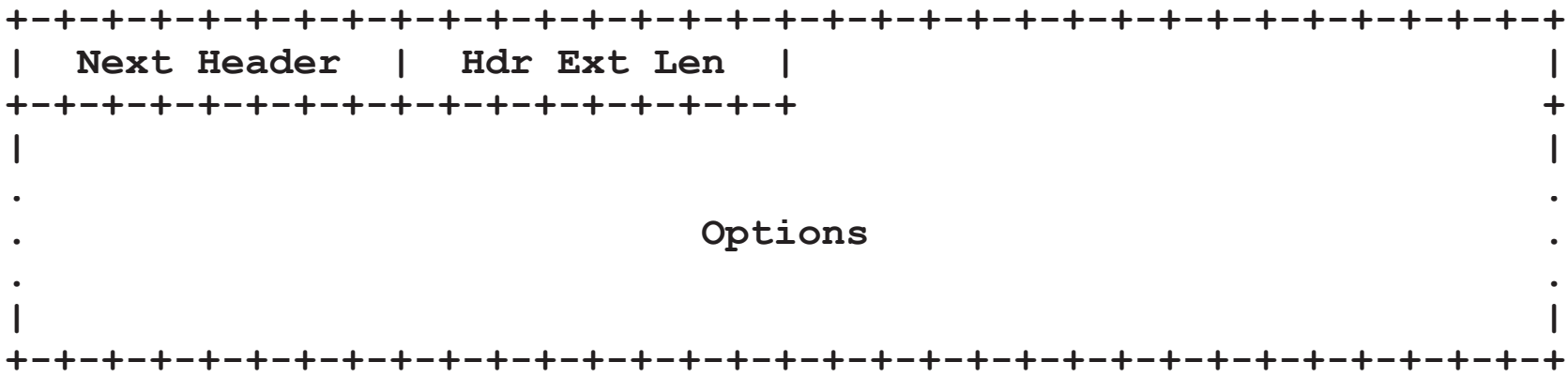
RFC2460 section 5. Packet Size Issues:

IPv6 requires that every link in the internet have an MTU of 1280 octets or greater. On any link that cannot convey a 1280-octet packet in one piece, link-specific fragmentation and reassembly must be provided at a layer below IPv6. It is strongly recommended that IPv6 nodes implement Path MTU Discovery [RFC-1981], in order to discover and take advantage of path MTUs greater than 1280 octets.

In order to send a packet larger than a path's MTU, a node may use the IPv6 Fragment header to fragment the packet at the source and have it reassembled at the destination(s).

Destination Options Header

The Destination Options header is used to carry optional information that need be examined only by a packet's destination node(s). The Destination Options header is identified by a Next Header value of 60 in the immediately preceding header, and has the following format:



Next Header 8-bits selector: type of header immediately following this header

Hdr Ext Len 8-bits unsigned integer: length of this header in 8-octet units, not including the first 8 octets.

Options (variable) Contains one or more TLV-encoded options, as described in section 4.2.

Note that there are two possible ways to encode optional destination information in an IPv6 packet: either as an option in the Destination Options header, or as a separate extension header.

Flow Labels

RFC2460 section 6. Flow Labels:

The 20-bit Flow Label field in the IPv6 header 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. This aspect of IPv6 is, at the time of writing, still experimental and subject to change as the requirements for flow support in the Internet become clearer. Hosts or routers that do not support the functions of the Flow Label field are required to set the field to zero when originating a packet, pass the field on unchanged when forwarding a packet, and ignore the field when receiving a packet.

Appendix A describes the current intended semantics and usage of the Flow Label field:

A flow is a sequence of packets sent from a particular source to a particular (unicast or multicast) destination for which the source desires special handling by the intervening routers. The details of such control protocols or options are beyond the scope of this document. There may be multiple active flows from a source to a destination, as well as traffic that is not associated with any flow.

A flow is uniquely identified by the combination of a source address and a non-zero flow label. Packets that do not belong to a flow carry a flow label of zero. A flow label is assigned to a flow by the flow's source node.

Rules for Packets within a Flow

from RFC2460 Appendix A:

- All packets belonging to the same flow must be sent with the same source address, destination address, and flow label.
- If any of those packets includes a Hop-by-Hop Options header, then they all must be originated with the same Hop-by-Hop Options header contents (excluding the Next Header field of the Hop-by-Hop Options header).
- If any of those packets includes a Routing header, then they all must be originated with the same contents in all extension headers up to and including the Routing header (excluding the Next Header field in the Routing header).
- maximum 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.
- A source must not re-use 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.
- If a node stops and restarts (e.g., as a result of a "crash"), it must be careful not to use a flow label that it might have used for an earlier flow whose lifetime may not have expired yet.

Traffic Class

RFC2460 section 7. Traffic Classes:

The 8-bit Traffic Class field in the IPv6 header is available for use by originating nodes and/or forwarding routers to identify and distinguish between different classes or priorities of IPv6 packets. As of December 1998, there were a number of experiments underway in IPv4. The Traffic Class field in the IPv6 header is intended to allow similar functionality to be supported in IPv6. Detailed definitions of the syntax and semantics of all or some of the IPv6 Traffic Class bits, whether experimental or intended for eventual standardization, are to be provided in separate documents.

The following general requirements apply to the Traffic Class field:

- The service interface to the IPv6 service within a node must provide a means for an upper-layer protocol to supply the value of the Traffic Class bits in packets originated by that upper-layer protocol. The default value must be zero for all 8 bits.
- Nodes that support a specific (experimental or eventual standard) use of Traffic Class bits are permitted to change the value of those bits in packets that they originate, forward, or receive, as required for that specific use. Nodes should ignore and leave unchanged any bits of the Traffic Class field for which they do not support a specific use.
- An upper-layer protocol must not assume that the value of the Traffic Class bits in a received packet are the same as the value sent by the packet's source.

DSCP + ECN

The upper six bits of Traffic Class are allocated to Differentiated Services Code Point (DSCP) in accordance with RFC 3260. The IANA.org Differentiated Services Field Codepoints registry shows the current assignments:

8: assigned by RFC 2474 (initial assignments, obsoleting RFC 1349)
12: assigned by RFC 2597 (assured forwarding, Per-Hop Behavior)
1: assigned by RFC 3246 (expedited forwarding, Per-Hop Behavior)
1: assigned by RFC 5865 (Capacity-Admitted Traffic)
10: to be Assigned by Standards Action
16: Available for expansion by Standards Action
16: Reserved for Experimental or Local Use

The lower two bits of Traffic Class are allocated to Explicit Congestion Notification (ECN) in accordance with RFC 3168. The values are:

00: Non ECN-Capable Transport - Non-ECT
10: ECN Capable Transport - ECT(0)
01: ECN Capable Transport - ECT(1)
11: Congestion Encountered - CE

Headers Already Defined

A full implementation of IPv6 includes implementation of the these extension headers:

- Hop-by-Hop Options
- Routing (Type 0 defined but deprecated)
- Fragment
- Destination Options
- Authentication (intended for IPsec)
- Encapsulating Security Payload (intended for (IPsec)

Authentication and Security headers are specified in [RFC-2402] and [RFC-2406], respectively.

The value 59 is reserved for “No Next Header” to indicate the packet contains no upper-layer header.

Order of the Headers

RFC2460 section 4.1 Extension Header Order:

It is recommended that those headers appear in the following order:

- IPv6 header
- Hop-by-Hop Options header
- Destination Options header (for intermediate Routing destinations)
- Routing header
- Fragment header
- Authentication header (see also RFC2406)
- Encapsulating Security Payload header (see also RFC2406)
- Destination Options header (for the final destination)
- upper-layer header

Each of these should occur at most once. The upper-layer header may be another IPv6 header in the case of IPv6 tunneled over IPv6.

If other extension headers are defined, ordering constraints relative to the above headers must be specified.

IPv6 nodes must accept and attempt to process extension headers in any order and occurring any number of times in the same packet, except for the Hop-by-Hop Options header which is restricted to appear immediately after an IPv6 header only. Nonetheless, it is strongly advised that sources of IPv6 packets adhere to the above recommended order until and unless subsequent specifications revise that recommendation.

ConEx Mechanism

Receiver signals sender to back off sending rate due to congestion

Sender relays congestion feedback at IPv6 layer

Policy Monitors/Enforcers verify congestion is being managed

Sender May Mark

- packet lost
- ECN-marked packet reached Receiver
- credit against future congestion (during RTT)
- ConEx aware (but none of the above)

Sender Mark should be immutable by any intermediate router.

Likely ConEx Proposal

- sender marks next packet sent
- any observer aggregates to get rate(s)
- monitor/enforcer counts against allowance and charges/disincents flow which exceeds it
- ECN used to reduce latency of detection

Loss-based (instead of ECN) being considered, near-sender and/or at predominant bottleneck

Fields Available in IPv6

Traffic Class

- DiffServe codepoints could be added
- ECN Nonce could be reclaimed

Flow Label

- could distinguish flows
- maybe could extract a bit or two

Other Fields Available in IPv6

Hop-by-Hop Options

- anything could be added
- but would suffer slow-path in backbone

Destination Options

- anything could be added
- would not be observed by routers

What do we want from Research Tool?

How important is aggregate-by-flow?

How do we measure success of LBE algorithms?

Can we measure DDoS?

Does anything here need authentication?

et cetera...

(supplemental slides follow)

Revised Rules for Flow Labels

from RFC3697 Section 2:

The Flow Label value set by the source MUST be delivered unchanged to the destination node(s).

IPv6 nodes MUST NOT assume any mathematical or other properties of the Flow Label values assigned by source nodes. Router performance SHOULD NOT be dependent on the distribution of the Flow Label values. Especially, the Flow Label bits alone make poor material for a hash key.

Nodes keeping dynamic flow state MUST NOT assume packets arriving 120 seconds or more after the previous packet of a flow still belong to the same flow, unless a flow state establishment method in use defines a longer flow state lifetime or the flow state has been explicitly refreshed within the lifetime duration.

If an IPv6 node is not providing flow-specific treatment, it MUST ignore the field when receiving or forwarding a packet.

from RFC3697 Section 3:

To avoid accidental Flow Label value reuse, the source node SHOULD select new Flow Label values in a well-defined sequence (e.g., sequential or pseudo-random) and use an initial value that avoids reuse of recently used Flow Label values each time the system restarts. The initial value SHOULD be derived from a previous value stored in non-volatile memory, or in the absence of such history, a randomly generated initial value using techniques that produce good randomness properties [RND] SHOULD be used.

from RFC3697 Section 4:

Flow state needs to be established on all or a subset of the IPv6 nodes on the path from the source to the destination(s). The methods for the state establishment, as well as the models for flow-specific treatment will be defined in separate specifications.

To enable co-existence of different methods in IPv6 nodes, the methods MUST meet the following basic requirements:

(1) The method MUST provide the means for flow state clean-up from the IPv6 nodes providing the flow-specific treatment. Signaling based methods where the source node is involved are free to specify flow state lifetimes longer than the default 120 seconds.

(2) Flow state establishment methods MUST be able to recover from the case where the requested flow state cannot be supported.

(other proposals exist in Internet-Draft form.)

Operation of ECN with TCP

TCP supports ECN congestion signaling from Receiver to Sender, using two flags in the TCP header (not the IP header). These are the ECN-Echo (ECE) and Congestion Window Reduced (CWR) bits. Use of ECN on a TCP connection is optional; for ECN to be used, it must be negotiated at connection establishment by including suitable options in the SYN and SYN-ACK segments.

When ECN has been negotiated on a TCP connection, the sender indicates that IP packets that carry TCP segments of that connection are carrying traffic from an ECN Capable Transport by marking them with an ECT codepoint. This allows intermediate routers that support ECN to mark those IP packets with the CE codepoint instead of dropping them in order to signal impending congestion.

Upon receiving an IP packet with the Congestion Experienced codepoint, the TCP receiver echoes back this congestion indication using the ECE flag in the TCP header. When a sender receives a TCP segment with the ECE bit, it reduces its congestion window as for a packet drop. It then acknowledges the congestion indication by sending a segment with the CWR bit set.

A Receiver keeps transmitting TCP segments with the ECE bit set until it receives a segment with the CWR bit set.

TCP Congestion Window

For each connection, TCP maintains a Congestion Window, limiting the total number of unacknowledged packets that may be in transit end-to-end. This is somewhat analogous to TCP's sliding window used for flow control.

Signalling from Receiver to Sender is TCP ACKnowledgments for packets received. The Sender adjusts its Congestion Window in various ways, depending on TCP version.

"Slow-Start" applies when a connection is initialized, or after a timeout. It starts with a window of two times the maximum segment size (MSS). The rate of increase isn't actually "slow:" for every packet ACKnowledged, the congestion window increases by 1 MSS so that the congestion window effectively doubles for every round trip time (RTT). When the congestion window exceeds a threshold "ssthresh," the algorithm enters a new state, called "Congestion Avoidance."

For TCP-Reno, if three duplicate ACKs are received (i.e., four ACKs acknowledging the same packet, which are not piggybacked on data, and do not change the receiver's advertised window), Reno will halve the Congestion Window, perform a "fast retransmit", and enter a phase called Fast Recovery. If an ACK times out, slow start is entered.