IPv6 Flow Label Specification

draft-ietf-ipv6-flow-label-02.txt

Jarno Rajahalme
Alex Conta
Brian E. Carpenter
Steve Deering

IETF #54, Yokohama

Changes since - 01

- Clarified definitions (section 1)
  - Contrasting the flow definition against RSVP and DiffServ definitions
- Changed text to not restrict flow definition to a “too fine granularity”
  - “Flow” is defined by the label and both addresses
  - Same “Flow state” can be shared by multiple “flows”
    - E.g. SCTP connections between nodes with multiple addresses
  - Addressing Brian Carpenter’s comment in the last meeting
- Re-wrote section 4 to give more practical guidance for flow labeling
  - Including SHOULDs for labeling application data streams and transport connections

Issues from the list

- Flow Label scope?
- Mandatory to implement?
- Flow caching?
- Load spreading?
- Flow state collisions in error situations
- Flow label value re-use
- Feedback from NSIS WG
- Security considerations

- WG last call

Issues

- Flow Label scope
  - Is the current text clear enough?
    "IPv6 nodes forwarding or receiving a labeled IPv6 packet can use the Flow Label and Source and Destination Address fields to classify the packet to a certain flow"
  - I.e. Flow Label values are meaningful in the context of the source AND destination addresses only.
  - Would adding "together" as below be better?
    "…can use the Flow Label and Source and Destination Address fields together to classify the packet to a certain flow"
- Flow Label mandatory to implement?
  - Not, but it is a SHOULD. Behavior for non-implementing nodes is included in section 3.

Issues (cont.)

- Should routers create flow state for unknown lows?
  - I.e. should routers cache forwarding or other information based on flow label and address values seen on packets being forwarded when there is no matching flow state?
  - No, that would invite nasty denial of service attacks
    - See RFC 1809
  - Specific hop-by-hop options might be an exception, but out of scope for the draft
- Load spreading
  - Example of an "algorithmic" flow state establishment method
    - But still new state entries are not created at the forwarding time
  - Just an example of potential use

Flow state collisions in error situations

- Currently the flow state timeout value (if any) depends on the flow state establishment method, no default value specified
    - But methods MUST provide means for flow state clean-up
- Problem: If the method does not clean up cleanly e.g. in a case of system crash, the same flow classifier may be accidentally re-used by a new flow after the reboot
    - Especially if there is no signaling for the new flow
    - Source may want to resume streaming multicast after a quick reboot...
- The risk is restricted to new flows between exactly the same source and destination addresses that had signaled state before the system crash
- Risk can be further minimized by choosing a random initial flow label value
    - This is what TCP does for the initial sequence number

Collisions (cont.)

- If the collision happens, packets may get wrong treatment until the old state is flushed
  - The timeout specified by the method that originally established the state
- Collisions can be 100% prevented by 100% reliable persistent storage, but this would be expensive
  - "Toasting flashes"
  - Even commit to disk may be too much for a normal IP stack to handle
- Is the residual threat significant enough to place any additional burden to the (default) implementations?
  - Specific signaling methods still MUST clean up the state
    - Or maybe even this should be a SHOULD, method could consider the probability and cost of collision low enough compared to the cost of providing 100% guarantee?
- Request on the list to NOT change the draft

Flow label value re-use

- “Re-use” happens only if the addresses are the same as in an already existing flow
- If an application indicates that a specific flow label value should be re-used, it MUST be re-used
  - Applications define which “streams” are parts of a same flow
    - e.g. two audio streams between the same addresses as a single flow
- An application SHOULD NOT pick a specific value for the 1st “component” of a flow
  - MAY specify re-using the same value for further components
  - However, app MAY have info stored over a reboot enabling resumption of video streaming, for example
- Applies also to transport protocols
  - May map multiple transport streams to the same flow
  - SCTP connections

NSIS comments by Robert Hancock

- 1. “Keeping packets together would be useful”
  - This could be an NSIS requirement, does not need to be a generic requirement
    - No change to the draft
- 2. “Precedence of the rules in section 4?”
  - Rules higher up in the list rule over the lower ones, (4) being the default if none of the above apply
  - This should be clearly stated in the draft
  - Would following suffice?

```
The assignment of a packet to a flow takes various forms, presented
below in the order of precedence:
(1) The source MAY take part in a signaling protocol that results in
assigning certain transport connection(s) or application data
stream(s) to specific flow(s).
(2) The source MAY be configured to assign certain transport
connection(s) or application data stream(s) to specific flow(s).
(3) The source SHOULD assign each new application data stream (e.g.
RTP streams) to a new flow.
(4) The source SHOULD assign each new transport connection (e.g.
TCP, SCTP) to a new flow.
```

NSIS (cont.)

- 3b. “End-systems to control state time out?”
  - Should probably be given some (constrained) control to end-systems
    - Would definition of a default value in this spec be counterproductive?
- 4. “Flow label agreement”?
  - An NSIS router could conclude that another flow state identity already “owns” the offered classifier
  - This would be a result of accidental (i.e. “unrelated”) reuse of the flow label value by applications either on the source and destination
  - Would be enough to fail the signaling in this case, offending app should pick another flow label and try again
- 5. Mobile IP home addresses in flow state establishment process
  - It is a SHOULD, but maybe should be reworded as (or removed):

    ```
    “Flow state establishment methods SHOULD avoid unnecessary flow
    state duplication in the IPv6 nodes on the path. For example, in
    the case of an mobile IPv6 node changing its Care-of Address this
    may require including the Mobile IP Home Addresses of the source
    and the destination in the state establishment process in addition
    to the Care-of Addresses being used by the actual classifiers.”
    ```

Security Considerations

- Flow label does not actually increase vulnerability for traffic analysis, since for analysis purposes, more detailed info (e.g. addresses, SPI, or ports) can be detected from the packet stream even when the flow label is not used
- Should remove the offending paragraph from the draft?

WG Last Call

- New draft addressing the decisions today to be issued next week, WG last call to start
- How much time for the WG last call?
- IETF last call after the WG comments have been addressed (late August?)

Default value for the flow state timeout?

- The minimum time a node w/o persistent storage must wait after it has lost flow state (e.g. crashed) before reusing any flow label values that may have been used before
  - Boot up time on embedded devices vs. personal computers?
  - Difficult to pick up one-size-fits-all value
- Also the minimum interval by which IPv6 nodes storing flow state must check flows liveliness
  - A different value can be defined by the flow state establishment method utilized to set up the old flow
    - Any value longer than the default will affect how the “crash reboot” case is to be handled
- Implicit flow state refresh based on flow accounting info
  - If no traffic, possible to refresh the state with “dummy” packet matching the flow classifier but with no payload
  - And/or explicit flow state refreshes
  - Check the flow specific counters (accounting info) when about to expire the flow state to see if the flow has been alive since it was last checked