An information model for Deterministic Data packets (and beyond ?)

draft-eckert-detnet-<tbd>

T. Eckert (Futurewei - tte@cs.fau.de)

DetNet interim meeting 12/2022
Goals

• Large number of deterministic “latency” proposals
  • Can/should we standardize all? How?

• Overview/comparison?

• AFAIK: Can not afford a separate header for each option
  • Eg: no hop-by-hop “routing headers in IPv6”
  • Forwarding HW would like to have as few headers as possible
  • Eg.: Not header for PREOF and another for queuing/latency
    • And then we forgot yet another function – third extension header?

• IETF process:
  • DetNet can define the function (proof: PREOF) and the information elements needed for it
  • Other WGs will likely have to define packetization
  • MPLS, 6MAN, ?none-for-IPv6?, ?BIER?

• Packet header thinking
  • If we need a new header, what other necessary/beneficial packet header fields would we want (latency or other DetNet functions).

• Writing up information model
  • Helps to separate packetization from functionality
  • Should hopefully be a good work item to make progress
  • Should be written so that other-WG folks may only need to read what is of interest to them
    • Size of information element, where inserted, where examined, read-only vs. read/write,...
Information elements (1) – non-latency

• IE: Sequence-number
  • Functions: IP PREOF, OAM (!), ? new/better MPLS extension header ?
  • Format: Is RFC8943 all we need (it was constrained by options of RFC385) ?

• IE: Flow-ID
  • Functions: Easier OAM/PREOF - avoids per-forwarding plane Flow-key (IP, MPLS, L2)
    • Without Flow-ID, OAM functions need to track label binding to know “Flow” (FEC)
  • Format: TBD. Example: (sender-id, sender-flow-id) – globally unique
Information elements (2) – end-to-end

• PlayOut Function (in egres DetNet router)
  • IE: Playout-TimeStamp
    • Set by ingres node based on known max-latency to end node and arrival time of packet.
    • Buffers packet on egres node until Playout-TimeStamp.
    • Assumes clock synchronization on ingres and egres node
    • Allows to use per-hop “jittery” (in-time) QoS (e.g.: rfc2212, TSN-ATS) and convert to “synchronous” (on-time) QoS on egress-node.
  • Format:
    • Size needs to be larger than maximum network end-to-end latency. 265 msec ?
    • Unit of granularity likely depending on network speed ?
    • What use-case has highest synchronicity requirement ?
      • Audio ? 1 usec ?
Information elements (3) – hop-by-hop

- PlayOut Function (in receiver)
  - IE: Accumulated Queuing Delay across path (eDelay)
    - Every router adds the latency the packet experienced on this router, from reception to sending
    - On wired networks, no significant external contributors to jitter
      - Link-propagation typically well jitter-free?! (some exceptions)
    - Allows network to operate without clock synchronization
      - Just require minimum per-router local frequency accuracy (easy to achieve for e.g.: “Ethernet accuracy”)
      - No buffering requirements in router – makes it most simple latency support function in routers?
    - Receiver can then use eDelay value for receiver-only playout buffering to
      - Buffering/delaying packets on receiver (software) much cheaper than in network?
  - Format:
    - Similar to prior slide. E.g.: usec accuracy, max-size < 1 sec -> 24...32 bits.
Information elements (4) hop-by-hop-latency

• IE: “delay” time (not timestamp!)
  • Damper Function – makes next node delay packet
  • Rewritten every hop
  • Size/accuracy depending on how “jitter-free” end-to-end path should be
    • And capabilities of forwaders for accurate timing of packets.
    • Max accuracy : time of sending 1 bit on fastest interface.
    • Size: maximum number of bits in a queue (from maximum latency on hop).

• IE: Cycle number
  • E.g.: TCQF - Determined cycle buffer on next hop
  • Rewritten on every hop
  • Similar in function, but much smaller than “delay” parameter (e.g.: 4 bit)
Information elements (5) per-hop-priority

• IE: list of per-hop priority
  • Small value, e.g.: 4 bit
  • But needs to be a sequence, one value per DetNet hop
  • Think of priority in UBS/TSN-ATS

• In DetNet with per-flow state, this could be attached to the state of the flow on each DetNet hop – but not with “per-flow stateless forwarding”
  • SR-MPLS, SRv6, BIER-TE

• Stateless forwarding already has packet header to indicate path (or tree)
  • Difficult to imagine that one would want a separate data-structure to indicate per-hop priorities
  • Likely: per-hop-priority needs to use/expand existing “steering header”
    • SR-MPLS: 16 label/SID per DetNet node (= 4 bits priority)
    • SRv6: 4-bit function parameter in SRH. 4 bits per steering hop in TBD CRH header
    • BIER/BIER-TE: ? Difficult?
Information elements (6) per-hop-deadline

• IE: list of per-hop deadlines
  • Proposed in e.g.: draft-stein-srtsn
  • Currently no deterministic calculus defined, but stochastic with high probabilities

• Similar considerations as per-hop priority
  • Except granularity
  • Possible with e.g.: 32 bits in SRH ?
  • Impossible within existing 20-bit SR-MPLS label space ?!

• Just example for:
  • A per-hop-sequence
Code points (1)

• Re-use DSCP idea
• Instead of defining Information elements with fixed semantic
• Define “Code-Point” style Information Elements
  • Semantic is assigned through configuration

• Example, for each DSCP:
  • Queue selection, queue-parameters, scheduler parameters for queue, drop parameters for packet (e.g.: RED, PIE,...)

• Using SRv6/SRH or SR-MPLS SID/programmability
  • is already a form of “code points”
  • So lets assume that hop-by-hop parameter (sequences) are using that.
(Thought) Experiment

• Assume we define a DetNet header with the following fields
  • If a router supports a particular queuing/latency mechanism:
    Would these fields be sufficient to support it (by configuring semantic of fields)

  • 32 bit latency mechanism parameter 1 (read/write)
    • Could be self descriptive, e.g.: start with 4-bit type
    • Mechanisms could structure it
  • 32 bit latency mechanism parameter 2 (read/write)
    • Could be self descriptive, e.g.: start with 4-bit type
    • Mechanisms could structure it
  • 32 bit for sequence number (read-only hop-by-hop)
  • 32 bit for flow-id (read-only hop-by-hop)

• Could we support all proposal (except for per-hop parameters) ?
  • Even if proposal is not standardized
  • We would need to allow how to configure in parallel standard and non-standard semantics
  • Or some algo might be standardized later.