SR-TSN

draft-stein-srtsn-00

DetNet

IETF-110

Yaakov (J) Stein
RAD
What is the problem I am solving?

SRTSN is a new mechanism for Time Sensitive packet flows

There are two good reasons not to use time scheduled gating (Qbv):

1. Qbv assigns packet flows to TDM-like timeslots negating much of the statmux advantage of PSNs

2. Qbv requires complex optimization and distribution which is a challenging exercise at scale (when not accomplished optimally efficiency is abysmal and/or delay requirements are not met)

Note that low latency is critical for URLLC xHauling yet, the TSN folks are not mandating their own mechanism for this kind of TS flow (802.1CM only includes preemption)

leaving only the alternative of physical traffic separation
What forwarder architecture is assumed?

Routers (forwarders) perform 2 distinct per-packet and per-router* functions:
- **forwarding**
  - to which output port
  - *where* to send
- **scheduling**
  - which packet to transmit
  - *when* to send

* They may also perform per-flow or per-router functions, which are already handled well enough

Note: I wrote output queues but conventional FIFO queues are not optimal data structures for SRTSN

with Segment Routing

with TSN/DetNet
What am I proposing?

There is an alternative to time scheduled gating that does not suffer from these drawbacks:

- it can be optimized even for relatively large networks
- its configuration can be easily and rapidly distributed
- time sensitive flows can be dynamically added or removed
- it lowers average latency as compared to standard queueing
- ratio of missed deadlines can be tuned

The approach requires adding a new extension header containing a *stack* (or equivalent) data structure.

One implementation of the proposal is merged with source routing or Segment Routing implementing a more complete form of network programming.
What else can be done?

There are several known ways to reduce end-to-end propagation delay, for example:

- **Longest In System**
  - insert the packet's birth time into the header
  - prioritize packets with earlier birth times
    
    *this is suboptimal since a LIS packet with a loose delay budget will be sent before a younger packet with a tight budget*

- **Earliest Deadline First**
  - insert packet’s deadline into the header
  - prioritize packets with earlier deadlines
    
    *this is suboptimal since an EDF packet already be close to its destination will be sent before a later packet far from*
So, what’s the stack-based approach?

The stack-based approach inserts into the packet *local* deadlines for each router along the path and each router prioritizes according to its own local deadline.

The router *may* perform EDF on local deadlines or maybe *Just In Time*, or any other method to ensure that the packet exits before its local deadline.

*In fact, one particularly convoluted method reproduces Qbv but without having to configure all the routers.*

Notes:

- the router needs something more complex than a FIFO queue but less complex than time scheduled gates
- there are several ways to compute the local deadlines (more on that later)
What is **SRTSN**?

If we are already using a stack why not reuse Segment Routing’s stack too?

With SRTSN each TS packet carries a stack with both

- forwarding (segment routing) instructions and
- scheduling (local deadline) instructions

in each stack entry

Like in SR, the stack is inserted by the ingress router which has its clock sync’ed to all the other routers so that the deadlines are directly comparable

There may additionally be non-TS packets with lower priority and there may be several priority levels of TS packets
Simple Example *

Total delay budget = 200 μsec
Minimal delay = link latencies + minimal residence times = 100 μsec
Fairly divide spare 100 μsec queueing time between forwarders

* This is just one way to set local deadlines
Won’t that take a lot of room?

If each SRTSN stack consists of
• a 128-bit IPv6 address
• a 64-bit timestamp
then each stack entry would consume 24 bytes!
But we needn’t be so wasteful!

Our deadline wraparound requirements are minimal
(they are unambiguous if wraparound is twice the maximum time path time)
and in a single network we need only specify router suffixes
(or even router indexes)

In fact, each entry need only be
\[
\text{cei}l\left( \log_2\left(N_{\text{routers}} \right) \right) + \text{cei}l\left( \log_2\left(2 \max\text{-path}\text{-time} / \text{resolution} \right) \right) + 1 \text{ bits}
\]

For small networks this is about 16 and for medium ones 32 bits
So, 4-8 hops only require about as much as a single IPv6 address!
What am I asking this WG?

Being a solution for Time Sensitive flows
this work seems to naturally fit the DetNet charter
– addresses bounds on latency
– focuses on the data plane aspects
– applicable to both L2 and L3 networks
  (no physical layer mechanism needed)
– mostly for networks under single administrative control

Is there interest here?

I request the DetNet and Spring chairs to coordinate
as to where this work should progress
Thanks for listening!

comments appreciated

Yaakov_S@rad.com