SR-TSN

draft-stein-srtsn-00

Spring

IETF-110

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

We propose a new mechanism for minimizing latency for **Time Sensitive** packet flows

As compared to current TSN/DetNet mechanisms

- 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 solution inserts a stack data structure into the packets and hence can be seen as an extension of Segment Routing
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

with Segment Routing *with TSN/DetNet

* 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*
What can be done without a stack?

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 destination*
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 in TSN/DetNet networks 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
SRTSN and Network Programming

We can think of a progression of network programming types

1. packet containing a single destination address
headers dictate only the next-hop forwarding treatment
everything else is external to the packet

2. Segment Routing
headers dictate the forwarding treatment
but scheduling is still external to the packet

3. SRTSN
stack determines both forwarding and scheduling treatment

Note that behaviors that are specified
• per-flow but not per-router (e.g., DSCP, TC)
  need only be specified once in the packet header
• per-router but not per-flow (e.g., port rate limiting)
  need only be maintained by the router
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
What am I asking this WG?

Since SRTSN reuses Segment Routing concepts this work seems to naturally fit the Spring charter

SPRING WG serves as a forum to discuss SPRING networks operations, define new applications of, and specify extensions of Segment Routing technologies.

Is there interest here in progressing this work?

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

comments appreciated

Yaakov_S@rad.com