Scheduling is not queuing

DetNet interim

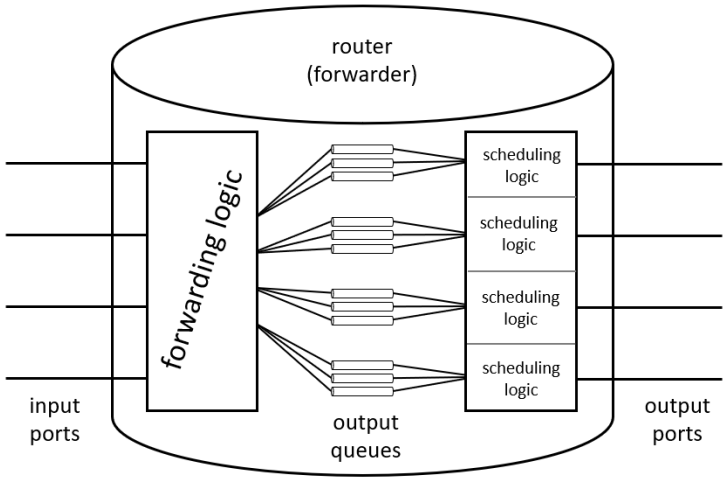
13 Sept 2021

Yaakov (J) Stein

yaakovjstein@gmail.com

RAD

What does a router (forwarder) do?



Forwarders perform 2 distinct per-packet **and** per-router* functions:

- **Forwarding** — selecting to which output port a packet should be sent
- **Scheduling** — selecting from which output queue the next packet to be forwarded should be selected

Scheduling is therefore often called called **queuing**

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

How has this changed?

When Segment Routing (or Traffic Engineering) is used
    forwarding is *more* than simply *selecting* the correct output port
It involves *deciding* ***where*** the packet should be forwarded

When TSN/DetNet is used (i.e., when the network elements are synchronized)
    scheduling is *more* than simply
        selecting from which queue to forward
It involves *deciding* ***when*** to send any given packet

Such generalized scheduling may not involve any queues at all
Instead, it may employ any mechanism
    that strives to send each packet at the *right time*

Where in the IETF?

The *right time* has both **control plane** and **user plane** aspects

- control (or management) plane
  calculate the right time per packet per router
- user (forwarding) plane
  schedule each packet, optimally before its right time

and is **hop-by-hop** and not **end-to-end**

Such scheduling is hence unambiguously work for the RTG area
and ***not*** the TSV area
even disregarding my proposal
for combining scheduling with Segment Routing

If not queues – then what?

A single queue is a **F**irst **I**n **F**irst **O**ut data structure

the packets in a queue are implicitly sorted by time of arrival

Multiple queues add a priority that is independent of time aspects

There are many alternatives to queue data structures, such as

- the stack – a **L**ast **I**n **F**irst **O**ut data structure
- the sorted list – a linear data structure ordered by some field
- the min (max) heap – a tree data structure in which every parent node is **<** (**>**) its child nodes

In particular, draft-stein-srtsn proposes using

- a stack of local deadlines in each packet
- a sorted list or min heap data structure in the router

rather than queues!

What can be done with a non-queue?

There are several *known* ways of using a sorted list or min heap
to reduce end-to-end propagation delay, for example :

- **L**ongest **I**n **S**ystem
  - insert the packet's birth time into the header
  - create sorted list or min heap based on birth times
  - forward packets with earlier birth times before packets with later times

    this is suboptimal since a longer in system packet
    with a loose delay budget
    will be sent before a younger packet with a tight budget

- **E**arliest **D**eadline **F**irst
  - insert packet's deadline into the header
  - create sorted list or min heap based on deadline
  - forward packets with earlier deadlines before packets with later deadlines

  this is suboptimal since an EDF packet
  already be near its destination
  will be sent before a later packet far from its destination

What is stack-based scheduling?

With the stack-based approach

- insert a stack of local deadlines
  (with an entry for each router along the path) into the header
- create sorted list or min heap based on local deadline
- forward earlier deadline packets before packets with later times

This method has multiple advantages

- the stack is inserted by the ingress router
  which has its clock sync'ed to all the other routers
  so that the local deadlines are directly comparable
- it can be optimized even for relatively large networks
- its configuration can be easily and rapidly distributed
- new flows can be dynamically added or removed
- it lowers average latency as compared to standard queueing
- ratio of missed deadlines can be tuned

What is SRTSN?

If we are already using a stack

why not reuse Segment Routing's stack too?

In such case a single joint optimization

- selects a path with appropriate immutable delay
- calculates delay offsets for each router along that path

With SRTSN each time sensitive packet carries a stack with both

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

in each stack entry

Summary

With synchronized forwarding elements

- scheduling means releasing packets at the right time
- queues are not optimal data structures for scheduling
  so scheduling is not queuing
- optimal scheduling involves
  - hop-by-hop and not end-to-end mechanisms
  - a sophisticated control or management plane

  and is thus unambiguously work for RTG not TSV
- inserting a stack of local deadlines into packet headers
  conveys multiple advantages
- combining deadline stacks with forwarding stacks
  is even better

Thanks for listening !

yaakovjstein@gmail.com