Architecture for Scheduled Use of Resources

draft-zhuang-teas-scheduled-resources-00

Yan Zhuang (zhuangyan.zhuang@huawei.com)
Qin Wu (bill.wu@huawei.com)
Adrian Farrel (adrian@olddog.co.uk)

IETF-94 : Yokohama : November 2015

What are the Services?

- The services we are considering are LSPs that reserve bandwidth
  - Any type of LSP
  - Bandwidth is basically “network resources”
- The value-add is that services can be booked for a time-slot in the future
  - “Guaranteed” to be provided
    - Unless something changes!
- Further option is to vary an existing or booked service during a time window in the future
  - Add or reduce bandwidth for a period
- Services are used
  - When there are limited physical resources
    - Booking wavelengths in optical networks
  - To make maximal use of resources in networks
    - Data centre inter-connect

Why This Draft?

- In Prague we noticed two drafts proposing solutions in this space
  - draft-zhuang-***pce***-stateful-pce-lsp-scheduling
  - draft-chen-***pce***-tts
- There is also some older work
  - draft-yong-***ccamp***-ason-gmpls-autobw-service
  - draft-zhang-***pce***-stateful-time-based-scheduling
- These drafts had different approaches and some unresolved issues
- It seems helpful to step back and look at the issues and architecture before working on solutions
  - TEAS is the right place to do this TE architecture work

What Does the Draft Do?

- Aims to get us all on the same page
  - Provide a reference for future work
- How?
  - Present a problem statement
    - When and why reserve resources?
    - What can go wrong if you don’t?
  - Describes the architectural concepts
    - Scheduling state: what and where
    - Discusses pros and cons of options
    - Recommends an architectural approach
  - Architectural overview
    - Figure
    - Service request processing
    - Initialisation and recover processing

Why do We Need the State?

- Want to maximise chance of service being delivered
  - Avoid contention for resources
    - Time arrives and resource is in use by someone else
    - Pre-emption is disruptive
    - Make-before-break re-optimization takes time
  - Which resources can we take out of service?
    - When is it safe?
    - When must they be returned?
    - Which planned services need to be re-planned or alerted?

What is the State We Need to Store?

- State applies to
  - The resources on a path through the network
  - The timing of reservation of those resources

{ link id;
resource id or reserved capacity;
start time;
end time }

- How much state is this?
  - How many start times do we need to support?
    - What is the arrival rate?
  - How long is a resource booked for?
    - What is the hold time?
- This question can be mitigated by:
  - Can we set a limited horizon?
    - How far into the future do we look?
  - Can we reduce the granularity?
    - Can we operate in one hour units, or 10 minutes, or 30 seconds, or one week?

Where do we Need the State?

- It depends on the architecture and who can request services
- Range of options…
  - Many applications can create “on-demand” services
  - Many applications can book resources
  - Centralised control of booking
  - Centralised control of all services
- These choices lead us to…
  - State is needed where time-based path computation is done
  - State is needed where on-demand path computation is done
  - State is needed where resources are reserved
- This is a philosophical question that substantially changes:
  - What function we can provide
  - What changes to protocols we make
  - What to implement

What do we mean by “distributed state”?

- We could mean
  - State is held where the resource exists
    - This prevents other services stealing the resource
    - But is doesn’t help other computation nodes
  - State is held “everywhere” in the network
    - Prevents stealing resources
    - But that is unlikely to be an issue because all path computation nodes can see what bookings exist
      - PCE servers
      - NMS / SDN controllers
      - Head-end LSRs requesting new services

How to Achieve Distributed State

- It all depends on the service architecture
- If we have centralised computation, but want to police reservations
  - State is in the PCE and stored in the network nodes
- If we have distributed computation and want to police reservations
  - State is in the network nodes and needs to be distributed to all points of computation

Why are People Concerned with Distributed State?

- It's a question of volume of data
  - If a node has 100 interfaces that can support 10,000 TE LSPs each, there is already some "interesting" challenges for state maintenance for a single point in time (i.e., now)
  - Suppose we allow booking in 15 minute intervals for a period of one month into the future
    - That is up to 4*24*30 = 2880 times as much state
  - In reality
    - One month may be too short
    - State can be considerably compressed
- If "future LSPs" are installed using RSVP-TE, then each such LSP also requires considerable RSVP protocol state

How Does Distributed State Persist?

- If state is installed by RSVP-TE we have to address the question of how the “future LSPs” survive network faults
  - We can use all of the soft-state/hard-state work
  - We can use RSVP-TE Recovery processing
  - But there is potentially a lot of processing to be done
- If state is installed some other way
  - That state needs to be resynched on recovery

Why are People Concerned by Distributing State?

- Suppose we want every node in the network to know about the scheduling state
  - This is no different from wanting every node to know about the other TE state
  - So we could use the IGP?
    - It is potentially a lot of information
    - The IGP has to refresh all information periodically (unless we change it)
    - The information must be advertised as new services are booked
    - Every node in the network has to hold all of the booking information for the whole network

Which leads to the alternative…

- Scheduling information is only held centrally
- This fits well with an active stateful PCE approach
  - Update the TED to show future reservations
  - Allow the LSPDB to hold future LSPs
- Can we integrate this with
  - Stateless PCE uses
    - Yes: easy
  - On-demand, non-PCE LSPs
    - Yes: no different from resource failures!

Details, details

- No changes to Signaling, IGPs, BGP-LS, information stored in the network
- Updates to PCEP to show LSP timing
- Synchronising databases between PCEs
  - It is messy, but no different from synching timeless LSPDBs
- Handling multiple PCEs for the same network
  - This is no different from today!
  - Two PCEs might both assign the same resources at the same time
    - At least with scheduling there is a chance to resolve this before the user notices
- Handling cooperating PCEs
  - We don’t think this changes
  - PCEs cooperate using PCReq
    - When one PCE responds to another, it “guarantees” a reservation
    - This might need to be released if not used

Warning to All Users!

- When a PCE agrees to a scheduled service, this is not a guarantee!
  - Network resources may fail
  - A more important user may come along
- The scheduling service is:
  - “We will try to deliver, possibly using re-routing, and let you know if the situation changes”

Next Steps for This Work

- Discuss to see whether we all agree
  - Early email exchanges suggest
    - Mainly agreement
    - Some desire to support distributed state
    - Concern to work through the details
- Decide whether this needs to be codified as an RFC by the TEAS WG
  - We could just discuss, agree, and move on