TEAS Working Group
Internet Draft
Intended status: Standard Track
Expires: September 2018

Italo Busi (Ed.)
Huawei
Sergio Belotti (Ed.)
Nokia
Victor Lopez
Oscar Gonzalez de Dios
Telefonica
Anurag Sharma
Google
Yan Shi
China Unicom
Ricard Vilalta
CTTC
Karthik Sethuraman
NEC

March 5, 2018

Yang model for requesting Path Computation
draft-ietf-teas-yang-path-computation-01.txt

Status of this Memo

Copyright Notice

Abstract

There are scenarios, typically in a hierarchical SDN context, in which an orchestrator may not have detailed information to be able to perform an end-to-end path computation and would need to request lower layer/domain controllers to calculate some (partial) feasible paths.

Multiple protocol solutions can be used for communication between different controller hierarchical levels. This document assumes that the controllers are communicating using YANG-based protocols (e.g., NETCONF or RESTCONF).

Based on this assumption this document proposes a YANG model for a path computation request that an higher controller can exploit to retrieve the needed information, complementing his topology knowledge, to make his E2E path computation feasible.

The draft proposes a stateless RPC which complements the stateful solution defined in [TE-TUNNEL].

Moreover this document describes some use cases where a path computation request, via YANG-based protocols (e.g., NETCONF or RESTCONF), can be needed.

Table of Contents

1. Introduction

There are scenarios, typically in a hierarchical SDN context, in which an orchestrator may not have detailed information to be able to perform an end-to-end path computation and would need to request lower layer/domain controllers to calculate some (partial) feasible paths.

When we are thinking to this type of scenarios we have in mind specific level of interfaces on which this request can be applied.

We can reference ABNO Control Interface [RFC7491] in which an Application Service Coordinator can request ABNO controller to take in charge path calculation (see Figure 1 in the RFC) and/or ACTN [ACTN-frame],where controller hierarchy is defined, the need for path computation arises on both interfaces CMI (interface between Customer Network Controller(CNC) and Multi Domain Service Coordinator (MDSC)) and/or MPI (interface between MSDC-PNC).[ACTN-Info] describes an information model for the Path Computation request.

Multiple protocol solutions can be used for communication between different controller hierarchical levels. This document assumes that the controllers are communicating using YANG-based protocols (e.g., NETCONF or RESTCONF).

Path Computation Elements, Controllers and Orchestrators perform their operations based on Traffic Engineering Databases (TED). Such TEDs can be described, in a technology agnostic way, with the YANG Data Model for TE Topologies [TE-TOPO]. Furthermore, the technology specific details of the TED are modeled in the augmented TE topology models (e.g. [OTN-TOPO] for OTN ODU technologies).

The availability of such topology models allows providing the TED using YANG-based protocols (e.g., NETCONF or RESTCONF). Furthermore, it enables a PCE/Controller performing the necessary abstractions or modifications and offering this customized topology to another PCE/Controller or high level orchestrator.

Note: This document does not assume that an orchestrator/coordinator always implements a “PCE” functionality, as defined in [RFC4655].

The tunnels that can be provided over the networks described with the topology models can be also set-up, deleted and modified via YANG-based protocols (e.g., NETCONF or RESTCONF) using the TE-Tunnel Yang model [TE-TUNNEL].

This document proposes a YANG model for a path computation request defined as a stateless RPC, which complements the stateful solution defined in [TE-TUNNEL].

Moreover, this document describes some use cases where a path computation request, via YANG-based protocols (e.g., NETCONF or RESTCONF), can be needed.

1.1. Terminology

TED: The traffic engineering database is a collection of all TE information about all TE nodes and TE links in a given network.

PCE: A Path Computation Element (PCE) is an entity that is capable of computing a network path or route based on a network graph, and of applying computational constraints during the computation. The PCE entity is an application that can be located within a network node or component, on an out-of-network server, etc. For example, a PCE would be able to compute the path of a TE LSP by operating on the TED and considering bandwidth and other constraints applicable to the TE LSP service request. [RFC4655]

2. Use Cases

This section presents different use cases, where an orchestrator needs to request underlying SDN controllers for path computation.

The presented uses cases have been grouped, depending on the different underlying topologies: a) IP-Optical integration; b) Multi-domain Traffic Engineered (TE) Networks; and c) Data center interconnections.

2.1. Packet/Optical Integration

In this use case, an Optical network is used to provide connectivity to some nodes of a Packet network (see Figure 1).

A possible example could be the case where an Optical network provides connectivity to same IP routers of an IP network.

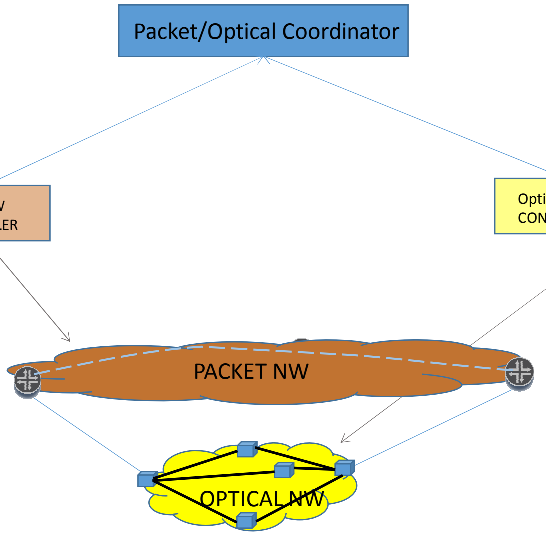

Figure 1 - Packet/Optical Integration Use Case

Figure 1 as well as Figure 2 below only show a partial view of the packet network connectivity, before additional packet connectivity is provided by the Optical network.

It is assumed that the Optical network controller provides to the packet/optical coordinator an abstracted view of the Optical network. A possible abstraction shall be representing the optical network as one “virtual node” with “virtual ports” connected to the access links.

It is also assumed that Packet network controller can provide the packet/optical coordinator the information it needs to setup connectivity between packet nodes through the Optical network (e.g., the access links).

The path computation request helps the coordinator to know the real connections that can be provided by the optical network.

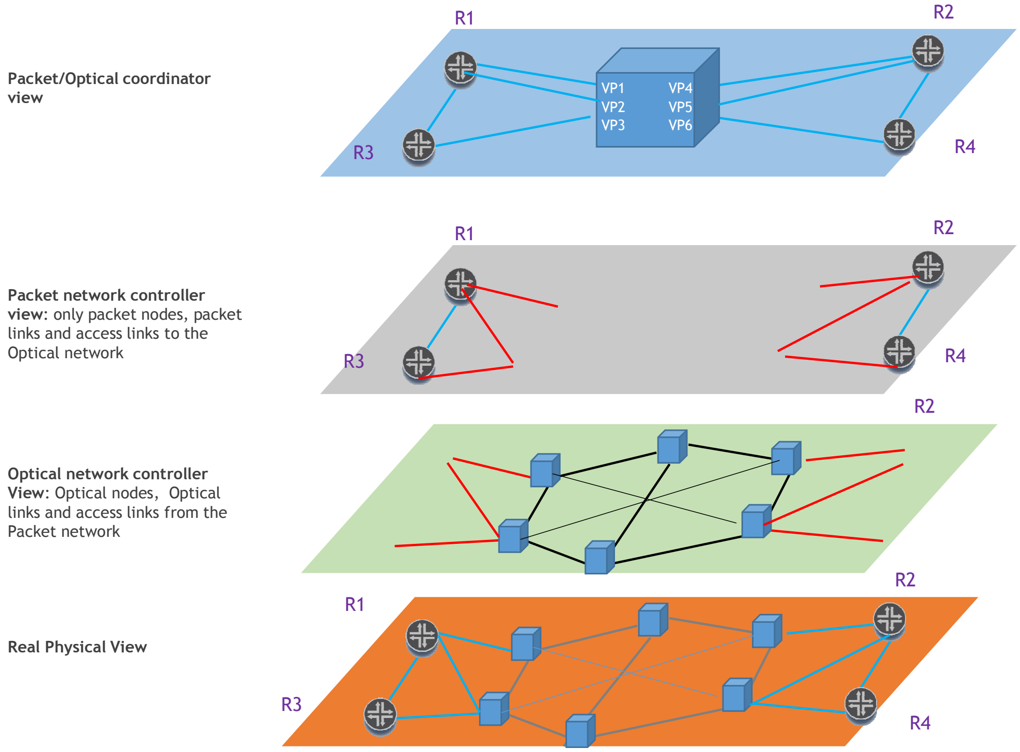
Packet/Optical coordinator view
R1
R2
R3
R4
VP1
VP2
VP3
VP4
VP5
VP6
Packet network controller view: only packet nodes, packet links and access links to the Optical network
R1
R2
R3
R4
Optical network controller View: Optical nodes, Optical links and access links from the Packet network
R2
Real Physical View
R1
R2
R3
R4

Figure 2 - Packet and Optical Topology Abstractions

In this use case, the coordinator needs to setup an optimal underlying path for an IP link between R1 and R2.

As depicted in Figure 2, the coordinator has only an "abstracted view" of the physical network, and it does not know the feasibility or the cost of the possible optical paths (e.g., VP1-VP4 and VP2-VP5), which depend from the current status of the physical resources within the optical network and on vendor-specific optical attributes.

The coordinator can request the underlying Optical domain controller to compute a set of potential optimal paths, taking into account optical constraints. Then, based on its own constraints, policy and knowledge (e.g. cost of the access links), it can choose which one of these potential paths to use to setup the optimal e2e path crossing optical network.

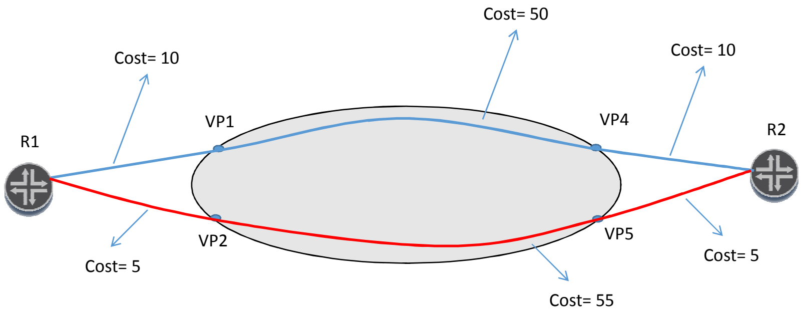

Figure 3 - Packet/Optical Path Computation Example

For example, in Figure 3, the Coordinator can request the Optical network controller to compute the paths between VP1-VP4 and VP2-VP5 and then decide to setup the optimal end-to-end path using the VP2-VP5 Optical path even this is not the optimal path from the Optical domain perspective.

Considering the dynamicity of the connectivity constraints of an Optical domain, it is possible that a path computed by the Optical network controller when requested by the Coordinator is no longer valid/available when the Coordinator requests it to be setup up.

It is worth noting that with the approach proposed in this document, the likelihood for this issue to happen can be quite small since the time window between the path computation request and the path setup request should be quite short (especially if compared with the time that would be needed to update the information of a very detailed abstract connectivity matrix).

If this risk is still not acceptable, the Orchestrator may also optionally request the Optical domain controller not only to compute the path but also to keep track of its resources (e.g., these resources can be reserved to avoid being used by any other connection). In this case, some mechanism (e.g., a timeout) needs to be defined to avoid having stranded resources within the Optical domain.

2.2. Multi-domain TE Networks

In this use case there are two TE domains which are interconnected together by multiple inter-domains links.

A possible example could be a multi-domain optical network.

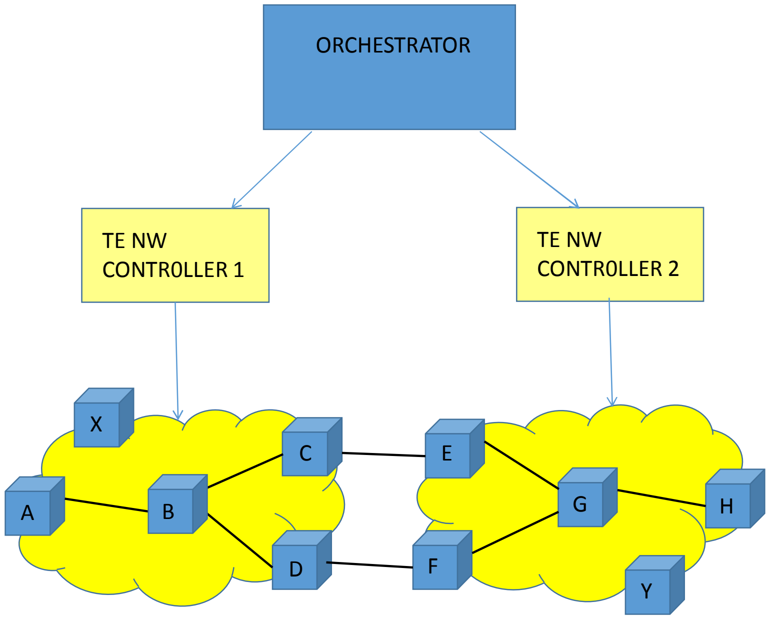

Figure 4 - Multi-domain multi-link interconnection

In order to setup an end-to-end multi-domain TE path (e.g., between nodes A and H), the orchestrator needs to know the feasibility or the cost of the possible TE paths within the two TE domains, which depend from the current status of the physical resources within each TE network. This is more challenging in case of optical networks because the optimal paths depend also on vendor-specific optical attributes (which may be different in the two domains if they are provided by different vendors).

In order to setup a multi-domain TE path (e.g., between nodes A and H), Orchestrator can request the TE domain controllers to compute a set of intra-domain optimal paths and take decisions based on the information received. For example:

o The Orchestrator asks TE domain controllers to provide set of paths between A-C, A-D, E-H and F-H

- o TE domain controllers return a set of feasible paths with the associated costs: the path A-C is not part of this set(in optical networks, it is typical to have some paths not being feasible due to optical constraints that are known only by the optical domain controller)

- o The Orchestrator will select the path A- D-F- H since it is the only feasible multi-domain path and then request the TE domain controllers to setup the A-D and F-H intra-domain paths

- o If there are multiple feasible paths, the Orchestrator can select the optimal path knowing the cost of the intra-domain paths (provided by the TE domain controllers) and the cost of the inter-domain links (known by the Orchestrator)

This approach may have some scalability issues when the number of TE domains is quite big (e.g. 20).

In this case, it would be worthwhile using the abstract TE topology information provided by the domain controllers to limit the number of potential optimal end-to-end paths and then request path computation to fewer domain controllers in order to decide what the optimal path within this limited set is.

For more details, see section 3.2.3.

2.3. Data center interconnections

In these use case, there is a TE domain which is used to provide connectivity between data centers which are connected with the TE domain using access links.

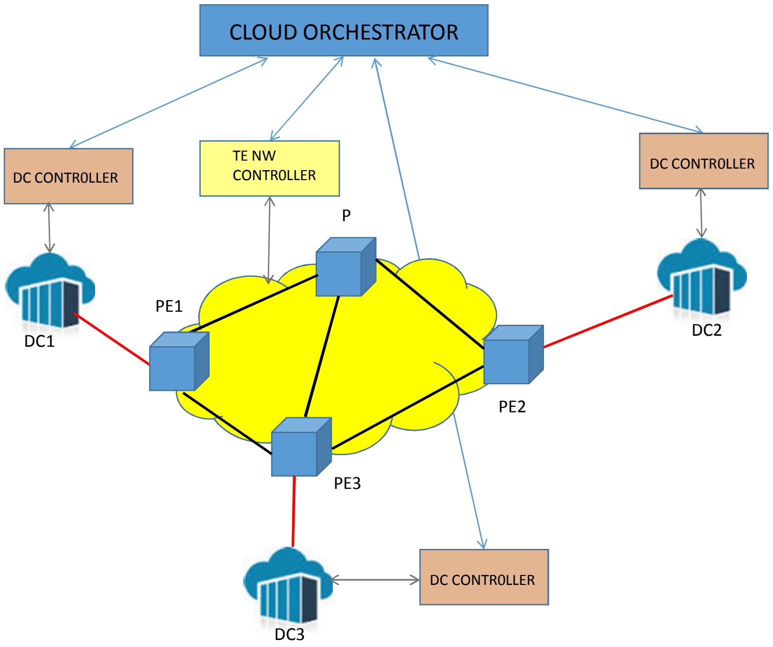

Figure 5 - Data Center Interconnection Use Case

In this use case, there is need to transfer data from Data Center 1 (DC1) to either DC2 or DC3 (e.g. workload migration).

The optimal decision depends both on the cost of the TE path (DC1-DC2 or DC1-DC3) and of the data center resources within DC2 or DC3.

The Cloud Orchestrator needs to make a decision for optimal connection based on TE Network constraints and data centers resources. It may not be able to make this decision because it has only an abstract view of the TE network (as in use case in 2.1).

The cloud orchestrator can request to the TE domain controller to compute the cost of the possible TE paths (e.g., DC1-DC2 and DC1-DC3) and to the DC controller to provide the information it needs about the required data center resources within DC2 and DC3 and then it can take the decision about the optimal solution based on this information and its policy.

3. Motivations

This section provides the motivation for the YANG model defined in this document.

Section 3.1 describes the motivation for a YANG model to request path computation.

Section 3.2 describes the motivation for a YANG model which complements the TE Topology YANG model defined in [TE-TOPO].

Section 3.3 describes the motivation for a stateless YANG RPC which complements the TE Tunnel YANG model defined in [TE-TUNNEL].

3.1. Motivation for a YANG Model

3.1.1. Benefits of common data models

Path computation requests are closely aligned with the YANG data models that provide (abstract) TE topology information, i.e., [TE-TOPO] as well as that are used to configure and manage TE Tunnels, i.e., [TE-TUNNEL]. Therefore, there is no need for an error-prone mapping or correlation of information. For instance, there is benefit in using the same endpoint identifiers in path computation requests and in the topology modeling. Also, the attributes used in path computation constraints use the same data models. As a result, there are many benefits in aligning path computation requests with YANG models for TE topology information and TE Tunnels configuration and management.

3.1.2. Benefits of a single interface

A typical use case for path computation requests is the interface between an orchestrator and a domain controller. The system integration effort is typically lower if a single, consistent interface is used between such systems, i.e., one data modeling language (i.e., YANG) and a common protocol (e.g., NETCONF or RESTCONF).

Practical benefits of using a single, consistent interface include:

1. Simple authentication and authorization: The interface between different components has to be secured. If different protocols have different security mechanisms, ensuring a common access control model may result in overhead. For instance, there may

be a need to deal with different security mechanisms, e.g., different credentials or keys. This can result in increased integration effort.

2. Consistency: Keeping data consistent over multiple different interfaces or protocols is not trivial. For instance, the sequence of actions can matter in certain use cases, or transaction semantics could be desired. While ensuring consistency within one protocol can already be challenging, it is typically cumbersome to achieve that across different protocols.
3. Testing: System integration requires comprehensive testing, including corner cases. The more different technologies are involved, the more difficult it is to run comprehensive test cases and ensure proper integration.
4. Middle-box friendliness: Provider and consumer of path computation requests may be located in different networks, and middle-boxes such as firewalls, NATs, or load balancers may be deployed. In such environments it is simpler to deploy a single protocol. Also, it may be easier to debug connectivity problems.
5. Tooling reuse: Implementers may want to implement path computation requests with tools and libraries that already exist in controllers and/or orchestrators, e.g., leveraging the rapidly growing eco-system for YANG tooling.

3.1.3. Extensibility

Path computation is only a subset of the typical functionality of a controller. In many use cases, issuing path computation requests comes along with the need to access other functionality on the same system. In addition to obtaining TE topology, for instance also configuration of services (setup/modification/deletion) may be required, as well as:

1. Receiving notifications for topology changes as well as integration with fault management
2. Performance management such as retrieving monitoring and telemetry data
3. Service assurance, e.g., by triggering OAM functionality
4. Other fulfilment and provisioning actions beyond tunnels and services, such as changing QoS configurations

YANG is a very extensible and flexible data modeling language that can be used for all these use cases.

The YANG model for path computation requests seamlessly complements with [TE-TOPO] and [TE-TUNNEL] in the use cases where YANG-based protocols (e.g., NETCONF or RESTCONF) are used.

3.2. Interactions with TE Topology

The use cases described in section 2 have been described assuming that the topology view exported by each underlying SDN controller to the orchestrator is aggregated using the "virtual node model", defined in [RFC7926].

TE Topology information, e.g., as provided by [TE-TOPO], could in theory be used by an underlying SDN controllers to provide TE information to the orchestrator thus allowing a PCE available within the Orchestrator to perform multi-domain path computation by its own, without requesting path computations to the underlying SDN controllers.

In case the Orchestrator does not implement a PCE function, as discussed in section 1, it could not perform path computation based on TE Topology information and would instead need to request path computation to the underlying controllers to get the information it needs to compute the optimal end-to-end path.

This section analyzes the need for an orchestrator to request underlying SDN controllers for path computation even in case the Orchestrator implements a PCE functionality, as well as how the TE Topology information and the path computation can be complementary.

In nutshell, there is a scalability trade-off between providing all the TE information needed by PCE, when implemented by the Orchestrator, to take optimal path computation decisions by its own versus requesting the Orchestrator to ask to too many underlying SDN Domain Controllers a set of feasible optimal intra-domain TE paths.

3.2.1. TE Topology Aggregation

Using the TE Topology model, as defined in [TE-TOPO], the underlying SDN controller can export the whole TE domain as a single abstract TE node with a "detailed connectivity matrix", which extends the "connectivity matrix", defined in [RFC7446], with specific TE attributes (e.g., delay, SRLGs and summary TE metrics).

The information provided by the “detailed abstract connectivity matrix” would be equivalent to the information that should be provided by “virtual link model” as defined in [RFC7926].

For example, in the Packet/Optical integration use case, described in section 2.1, the Optical network controller can make the information shown in Figure 3 available to the Coordinator as part of the TE Topology information and the Coordinator could use this information to calculate by its own the optimal path between R1 and R2, without requesting any additional information to the Optical network Controller.

However, there is a tradeoff between accuracy (i.e., providing "all" the information that might be needed by the PCE available to Orchestrator) and scalability, to be considered when designing the amount of information to provide within the "detailed abstract connectivity matrix".

Figure 6 below shows another example, similar to Figure 3, where there are two possible Optical paths between VP1 and VP4 with different properties (e.g., available bandwidth and cost).

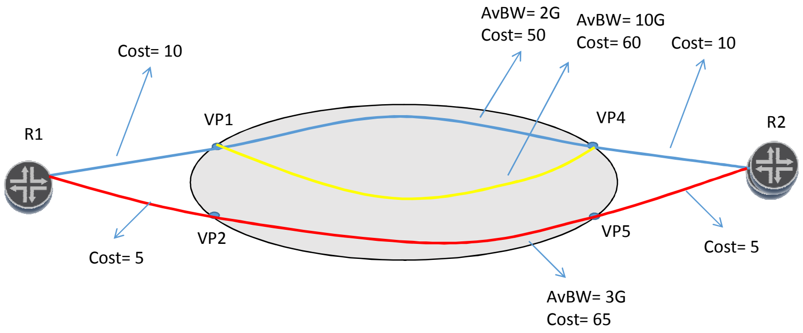

Figure 6 - Packet/Optical Path Computation Example with multiple choices

Reporting all the information, as in Figure 6, using the "detailed abstract connectivity matrix”, is quite challenging from a scalability perspective. The amount of this information is not just based on number of end points (which would scale as N-square), but also on many other parameters, including client rate, user constraints / policies for the service, e.g. max latency < N ms, max cost, etc., exclusion policies to route around busy links, min OSNR margin, max preFEC BER etc. All these constraints could be different based on connectivity requirements.

Examples of how the “detailed connectivity matrix” can be dimensioned are described in Appendix A.

It is also worth noting that the "connectivity matrix" has been originally defined in WSON, [RFC7446] to report the connectivity constrains of a physical node within the WDM network: the information it contains is pretty "static" and therefore, once taken and stored in the TE data base, it can be always being considered valid and up-to-date in path computation request.

Using the "connectivity matrix" with an abstract node to abstract the information regarding the connectivity constraints of an Optical domain, would make this information more "dynamic" since the connectivity constraints of an Optical domain can change over time because some optical paths that are feasible at a given time may become unfeasible at a later time when e.g., another optical path is established. The information in the "detailed abstract connectivity matrix" is even more dynamic since the establishment of another optical path may change some of the parameters (e.g., delay or available bandwidth) in the "detailed abstract connectivity matrix" while not changing the feasibility of the path.

“Connectivity matrix” is sometimes confused with optical reach table that contain multiple (e.g. k-shortest) regen-free reachable paths for every A-Z node combination in the network. Optical reach tables can be calculated offline, utilizing vendor optical design and planning tools, and periodically uploaded to the Controller: these optical path reach tables are fairly static. However, to get the connectivity matrix, between any two sites, either a regen free path can be used, if one is available, or multiple regen free paths are concatenated to get from src to dest, which can be a very large combination. Additionally, when the optical path within optical domain needs to be computed, it can result in different paths based on input objective, constraints, and network conditions. In summary, even though “optical reachability table” is fairly static, which regen free paths to build the connectivity matrix between any source and destination is very dynamic, and is done using very sophisticated routing algorithms.

There is therefore the need to keep the information in the "connectivity matrix" updated which means that there another tradeoff between the accuracy (i.e., providing "all" the information that might be needed by the Orchestrator’s PCE) and having up-to-date information. The more the information is provided and the longer it takes to keep it up-to-date which increases the likelihood that the Orchestrator’s PCE computes paths using not updated information.

It seems therefore quite challenging to have a "detailed abstract connectivity matrix" that provides accurate, scalable and updated information to allow the Orchestrator’s PCE to take optimal decisions by its own.

If the information in the "detailed abstract connectivity matrix” is not complete/accurate, we can have the following drawbacks considering for example the case in Figure 6:

o If only the VP1-VP4 path with available bandwidth of 2 Gb/s and cost 50 is reported, the Orchestrator’s PCE will fail to compute a 5 Gb/s path between routers R1 and R2, although this would be feasible;

o If only the VP1-VP4 path with available bandwidth of 10 Gb/s and cost 60 is reported, the Orchestrator’s PCE will compute, as optimal, the 1 Gb/s path between R1 and R2 going through the VP2-VP5 path within the Optical domain while the optimal path would actually be the one going thought the VP1-VP4 sub-path (with cost 50) within the Optical domain.

Instead, using the approach proposed in this document, the Orchestrator, when it needs to setup an end-to-end path, it can request the Optical domain controller to compute a set of optimal paths (e.g., for VP1-VP4 and VP2-VP5) and take decisions based on the information received:

o When setting up a 5 Gb/s path between routers R1 and R2, the Optical domain controller may report only the VP1-VP4 path as the only feasible path: the Orchestrator can successfully setup the end-to-end path passing though this Optical path;

o When setting up a 1 Gb/s path between routers R1 and R2, the Optical domain controller (knowing that the path requires only 1 Gb/s) can report both the VP1-VP4 path, with cost 50, and the VP2-VP5 path, with cost 65. The Orchestrator can then compute the optimal path which is passing thought the VP1-VP4 sub-path (with cost 50) within the Optical domain.

3.2.2. TE Topology Abstraction

Using the TE Topology model, as defined in [TE-TOPO], the underlying SDN controller can export an abstract TE Topology, composed by a set of TE nodes and TE links, which are abstracting the topology controlled by each domain controller.

Considering the example in Figure 4, the TE domain controller 1 can export a TE Topology encompassing the TE nodes A, B, C and D and the TE Link interconnecting them. In a similar way, TE domain controller 2 can export a TE Topology encompassing the TE nodes E, F, G and H and the TE Link interconnecting them.

In this example, for simplicity reasons, each abstract TE node maps with each physical node, but this is not necessary.

In order to setup a multi-domain TE path (e.g., between nodes A and H), the Orchestrator can compute by its own an optimal end-to-end path based on the abstract TE topology information provided by the domain controllers. For example:

- Orchestrator's PCE, based on its own information, can compute the optimal multi-domain path being A-B-C-E-G-H, and then request the TE domain controllers to setup the A-B-C and E-G-H intra-domain paths

- But, during path setup, the domain controller may find out that A-B-C intra-domain path is not feasible (as discussed in section 2.2, in optical networks it is typical to have some paths not being feasible due to optical constraints that are known only by the optical domain controller), while only the path A-B-D is feasible

- So what the hierarchical controller computed is not good and need to re-start the path computation from scratch

As discussed in section 3.2.1, providing more extensive abstract information from the TE domain controllers to the multi-domain Orchestrator may lead to scalability problems.

In a sense this is similar to the problem of routing and wavelength assignment within an Optical domain. It is possible to do first routing (step 1) and then wavelength assignment (step 2), but the chances of ending up with a good path is low. Alternatively, it is possible to do combined routing and wavelength assignment, which is known to be a more optimal and effective way for Optical path setup. Similarly, it is possible to first compute an abstract end-to-end path within the multi-domain Orchestrator (step 1) and then compute an intra-domain path within each Optical domain (step 2), but there are more chances not to find a path or to get a suboptimal path that performing per-domain path computation and then stitch them.

3.2.3. Complementary use of TE topology and path computation

As discussed in section 2.2, there are some scalability issues with path computation requests in a multi-domain TE network with many TE domains, in terms of the number of requests to send to the TE domain controllers. It would therefore be worthwhile using the TE topology information provided by the domain controllers to limit the number of requests.

An example can be described considering the multi-domain abstract topology shown in Figure 7. In this example, an end-to-end TE path between domains A and F needs to be setup. The transit domain should be selected between domains B, C, D and E.

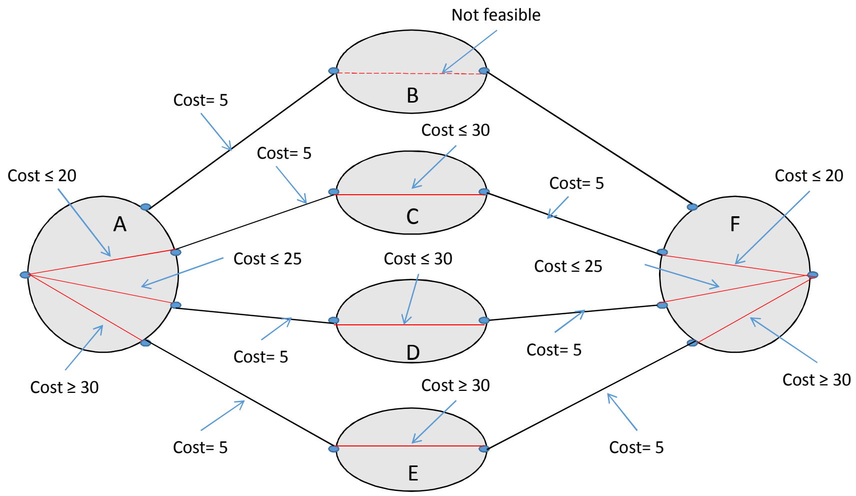

Figure 7 - Multi-domain with many domains (Topology information)

The actual cost of each intra-domain path is not known a priori from the abstract topology information. The Orchestrator only knows, from the TE topology provided by the underlying domain controllers, the feasibility of some intra-domain paths and some upper-bound and/or lower-bound cost information. With this information, together with the cost of inter-domain links, the Orchestrator can understand by its own that:

o Domain B cannot be selected as the path connecting domains A and E is not feasible;

o Domain E cannot be selected as a transit domain since it is know from the abstract topology information provided by domain controllers that the cost of the multi-domain path A-E-F (which is 100, in the best case) will be always be higher than the cost of the multi-domain paths A-D-F (which is 90, in the worst case) and A-E-F (which is 80, in the worst case)

Therefore, the Orchestrator can understand by its own that the optimal multi-domain path could be either A-D-F or A-E-F but it cannot known which one of the two possible option actually provides the optimal end-to-end path.

The Orchestrator can therefore request path computation only to the TE domain controllers A, D, E and F (and not to all the possible TE domain controllers).

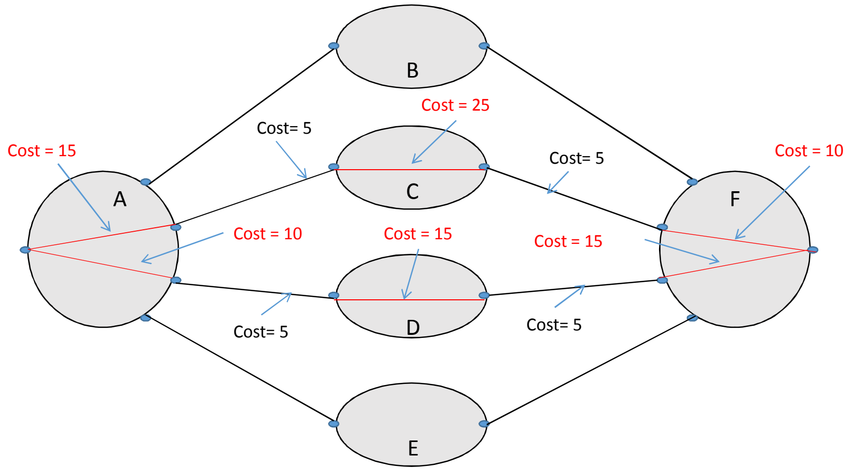

Figure 8 - Multi-domain with many domains (Path Computation information)

Based on these requests, the Orchestrator can know the actual cost of each intra-domain paths which belongs to potential optimal end-to-end paths, as shown in Figure 8, and then compute the optimal end-to-end path (e.g., A-D-F, having total cost of 50, instead of A-C-F having a total cost of 70).

3.3. Stateless and Stateful Path Computation

The TE Tunnel YANG model, defined in [TE-TUNNEL], can support the need to request path computation.

It is possible to request path computation by configuring a "compute-only" TE tunnel and retrieving the computed path(s) in the LSP(s) Record-Route Object (RRO) list as described in section 3.3.1 of [TE-TUNNEL].

This is a stateful solution since the state of each created "compute-only" TE tunnel needs to be maintained and updated, when underlying network conditions change.

It is very useful to provide options for both stateless and stateful path computation mechanisms. It is suggested to use stateless mechanisms as much as possible and to rely on stateful path computation when really needed.

Stateless RPC allows requesting path computation using a simple atomic operation and it is the natural option/choice, especially with stateless PCE.

Since the operation is stateless, there is no guarantee that the returned path would still be available when path setup is requested: this is not a major issue in case the time between path computation and path setup is short.

The RPC response must be provided synchronously and, if collaborative computations are time consuming, it may not be possible to immediate reply to client.

In this case, the client can define a maximum time it can wait for the reply, such that if the computation does not complete in time, the server will abort the path computation and reply to the client with an error. It may be possible that the server has tighter timing constraints than the client: in this case the path computation is aborted earlier than the time specified by the client.

Note - The RPC response issue (slow RPC server) is not specific to the path computation RPC case so, it may be worthwhile, evaluating whether a more generic solution applicable to any YANG RPC can be used instead.

In case the stateless solution is not sufficient, a stateful solution, based on "compute-only" TE tunnel, could be used to support asynchronous operations and/or to get notifications in case the computed path has been changed.

It is worth noting that also the stateful solution, although increasing the likelihood that the computed path is available at path setup, it does not guaranteed that because notifications may not be reliable or delivered on time.

The stateful path computation has also the following drawbacks:

- Several messages required for any path computation
- Requires persistent storage in the provider controller
- Need for garbage collection for stranded paths
- Process burden to detect changes on the computed paths in order to provide notifications update

4. Path Computation and Optimization for multiple paths

There are use cases, where it is advantageous to request path computation for a set of paths, through a network or through a network domain, using a single request [RFC5440].

This would reduce the protocol overhead to send multiple requests.

In the context of a typical multi-domain TE network, there could multiple choices for the ingress/egress points of a domain and the Orchestrator needs to request path computation between all the ingress/egress pairs to select the best pair. For example, in the example of section 2.2, the Orchestrator needs to request the TE network controller 1 to compute the A-C and the A-D paths and to the TE network controller 2 to compute the E-H and the F-H paths.

It is also possible that the Orchestrator receives a request to setup a group of multiple end to end connections. The orchestrator needs to request each TE domain controller to compute multiple paths, one (or more) for each end to end connection.

There are also scenarios where it can be needed to request path computation for a set of paths in a synchronized fashion.

One example could be computing multiple diverse paths. Computing a set of diverse paths in a not-synchronized fashion, leads to a high probability of not being able to satisfy all request. In this case, a sub-optimal primary path that could be protected by a diversely routed secondary path should be computed instead of an optimal primary path that could not be protected.

There are also scenarios where it is needed to request optimizing a set of paths using objective functions that apply to the whole set of paths, see [RFC5541], e.g. to minimize the sum of the costs of all the computed paths in the set.

5. YANG Model for requesting Path Computation

This document define a YANG stateless RPC to request path computation as an “augmentation” of tunnel-rpc, defined in [TE-TUNNEL]. This model provides the RPC input attributes that are needed to request path computation and the RPC output attributes that are needed to report the computed paths.

```
augment /te:tunnels-rpc/te:input/te:tunnel-info:
  +---- path-request* [request-id]
  ...........

augment /te:tunnels-rpc/te:output/te:result:
  +--ro response* [response-id]
     +--ro response-id      uint32
     +--ro (response-type)?
        +--:(no-path-case)
        |  +--ro no-path!
        +--:(path-case)
           +--ro computed-path
              +--ro path-id?          yang-types:uuid
              +--ro path-properties
              ...........
```

This model extensively re-uses the grouping defined in [TE-TUNNEL] to ensure maximal syntax and semantics commonality.

5.1. Synchronization of multiple path computation requests

The YANG model permits to synchronize a set of multiple path requests (identified by specific request-id) all related to a “svec” container emulating the syntax of “SVEC” PCEP object [RFC 5440].

```
    +---- synchronization* [synchronization-id]
       +---- synchronization-id    uint32
       +---- svec
       |  +---- relaxable?            boolean
       |  +---- link-diverse?         boolean
       |  +---- node-diverse?         boolean
       |  +---- srlg-diverse?         boolean
       |  +---- request-id-number*   uint32
       +---- svec-constraints
       |  +---- path-metric-bound* [metric-type]
       |     +---- metric-type    identityref
       |     +---- upper-bound?   uint64
       +---- path-srlgs
       |  +---- usage?    identityref
       |  +---- values*   srlg
       +---- exclude-objects
       ...........
       +---- optimizations
          +---- (algorithm)?
             +--:(metric)
             |  +---- optimization-metric* [metric-type]
             |     +---- metric-type    identityref
             |     +---- weight?        uint8
             +--:(objective-function)
                +---- objective-function
                   +---- objective-function-type?   identityref
```

The model, in addition to the metric types, defined in [TE-TUNNEL], which can be applied to each individual path request, defines additional specific metrics types that apply to a set of synchronized requests, as referenced in [RFC5541].

```
  identity svec-metric-type {
    description
      "Base identity for svec metric type";
  }
```

```
     identity svec-metric-cumul-te {
       base svec-metric-type;
       description
         "TE cumulative path metric";
     }

     identity svec-metric-cumul-igp {
       base svec-metric-type;
       description
         "IGP cumulative path metric";
     }

     identity svec-metric-cumul-hop {
       base svec-metric-type;
       description
         "Hop cumulative path metric";
     }

     identity svec-metric-aggregate-bandwidth-consumption {
       base svec-metric-type;
       description
         "Cumulative bandwith consumption of the set of synchronized
   paths";
     }

     identity svec-metric-load-of-the-most-loaded-link {
       base svec-metric-type;
       description
         "Load of the most loaded link";
     }
```

5.2. Returned metric values

This YANG model provides a way to return the values of the metrics computed by the path computation in the output of RPC, together with other important information (e.g. srlg, affinities, explicit route), emulating the syntax of the “C” flag of the “METRIC” PCEP object [RFC 5440]:

```
     augment /te:tunnels-rpc/te:output/te:result:
       +--ro response* [response-id]
          +--ro response-id        uint32
          +--ro (response-type)?
             +--:(no-path-case)
             |  +--ro no-path!
             +--:(path-case)
                +--ro pathCompService
                   +--ro path-id?           yang-types:uuid
                   +--ro path-properties
                      +--ro path-metric* [metric-type]
                      |  +--ro metric-type           identityref
                      |  +--ro accumulative-value?   uint64
                      +--ro path-affinities
                      |  +--ro constraint* [usage]
                      |     +--ro usage    identityref
                      |     +--ro value?   admin-groups
                      +--ro path-srlgs
                      |  +--ro usage?    identityref
                      |  +--ro values*   srlg
                      +--ro path-route-objects
                      ...........
```

It also allows to request which metric should returned in the input of RPC:

```
     augment /te:tunnels-rpc/te:input/te:tunnel-info:
       +---- path-request* [request-id]
       |  +---- request-id                uint32
       ...........
       |  +---- requested-metrics* [metric-type]
       |     +---- metric-type    identityref
       ...........
```

This feature is essential for using a stateless path computation in a multi-domain TE network as described in section 2.2. In this case, the metrics returned by a path computation requested to a given TE network controller must be used by the Orchestrator to compute the best end-to-end path. If they are missing the Orchestrator cannot compare different paths calculated by the TE network controllers and choose the best one for the optimal e2e path.

6. YANG model for stateless TE path computation

6.1. YANG Tree

Figure 9 below shows the tree diagram of the YANG model defined in module ietf-te-path-computation.yang.

```
module: ietf-te-path-computation
    +--rw paths
       +--ro path* [path-id]
          +--ro path-id            yang-types:uuid
          +--ro path-properties
             +--ro path-metric* [metric-type]
             |  +--ro metric-type           identityref
             |  +--ro accumulative-value?   uint64
             +--ro path-affinities
             |  +--ro constraint* [usage]
             |     +--ro usage    identityref
             |     +--ro value?   admin-groups
             +--ro path-srlgs
             |  +--ro usage?    identityref
             |  +--ro values*   srlg
             +--ro path-route-objects
                +--ro path-route-object* [index]
                   +--ro index             uint32
                   +--ro (type)?
                      +--:(numbered)
                      |  +--ro numbered-hop
                      |     +--ro address?     te-types:te-tp-id
                      |     +--ro hop-type?    te-hop-type
                      |     +--ro direction?   te-link-direction
                      +--:(as-number)
                      |  +--ro as-number-hop
                      |     +--ro as-number?   binary
                      |     +--ro hop-type?    te-hop-type
                      +--:(unnumbered)
                      |  +--ro unnumbered-hop
                      |     +--ro node-id?      te-types:te-node-id
                      |     +--ro link-tp-id?   te-types:te-tp-id
                      |     +--ro hop-type?     te-hop-type
```

```
                        |     +--ro direction?    te-link-direction
                        +--:(label)
                           +--ro label-hop
                              +--ro te-label
                                 +--ro (technology)?
                                 |  +--:(generic)
                                 |     +--ro generic?     rt-
   types:generalized-label
                                 +--ro direction?   te-label-direction
     augment /te:tunnels-rpc/te:input/te:tunnel-info:
       +---- path-request* [request-id]
       |  +---- request-id                uint32
       |  +---- source?                   inet:ip-address
       |  +---- destination?              inet:ip-address
       |  +---- src-tp-id?                binary
       |  +---- dst-tp-id?                binary
       |  +---- bidirectional
       |  |  +---- association
       |  |     +---- id?              uint16
       |  |     +---- source?          inet:ip-address
       |  |     +---- global-source?   inet:ip-address
       |  |     +---- type?            identityref
       |  |     +---- provisioning?    identityref
       |  +---- explicit-route-objects
       |  |  +---- route-object-exclude-always* [index]
       |  |  |  +---- index             uint32
       |  |  |  +---- (type)?
       |  |  |     +--:(numbered)
       |  |  |     |  +---- numbered-hop
       |  |  |     |     +---- address?     te-types:te-tp-id
       |  |  |     |     +---- hop-type?    te-hop-type
       |  |  |     |     +---- direction?   te-link-direction
       |  |  |     +--:(as-number)
       |  |  |     |  +---- as-number-hop
       |  |  |     |     +---- as-number?   binary
       |  |  |     |     +---- hop-type?    te-hop-type
       |  |  |     +--:(unnumbered)
       |  |  |     |  +---- unnumbered-hop
       |  |  |     |     +---- node-id?      te-types:te-node-id
       |  |  |     |     +---- link-tp-id?   te-types:te-tp-id
       |  |  |     |     +---- hop-type?     te-hop-type
       |  |  |     |     +---- direction?    te-link-direction
       |  |  |     +--:(label)
       |  |  |        +---- label-hop
       |  |  |           +---- te-label
       |  |  |              +---- (technology)?
       |  |  |              |  +--:(generic)
       |  |  |              |     +---- generic?     rt-
   types:generalized-label
       |  |  |              +---- direction?   te-label-direction
       |  |  +---- route-object-include-exclude* [index]
       |  |     +---- explicit-route-usage?   identityref
       |  |     +---- index                   uint32
       |  |     +---- (type)?
       |  |        +--:(numbered)
       |  |        |  +---- numbered-hop
       |  |        |     +---- address?     te-types:te-tp-id
       |  |        |     +---- hop-type?    te-hop-type
       |  |        |     +---- direction?   te-link-direction
       |  |        +--:(as-number)
       |  |        |  +---- as-number-hop
       |  |        |     +---- as-number?   binary
       |  |        |     +---- hop-type?    te-hop-type
       |  |        +--:(unnumbered)
       |  |        |  +---- unnumbered-hop
       |  |        |     +---- node-id?      te-types:te-node-id
       |  |        |     +---- link-tp-id?   te-types:te-tp-id
       |  |        |     +---- hop-type?     te-hop-type
       |  |        |     +---- direction?    te-link-direction
       |  |        +--:(label)
       |  |           +---- label-hop
       |  |              +---- te-label
       |  |                 +---- (technology)?
       |  |                 |  +--:(generic)
       |  |                 |     +---- generic?     rt-
   types:generalized-label
       |  |                 +---- direction?   te-label-direction
       |  +---- path-constraints
       |  |  +---- te-bandwidth
       |  |  |  +---- (technology)?
       |  |  |     +--:(generic)
       |  |  |        +---- generic?   te-bandwidth
       |  |  +---- setup-priority?       uint8
       |  |  +---- hold-priority?        uint8
       |  |  +---- signaling-type?       identityref
       |  |  +---- disjointness?         te-types:te-path-disjointness
       |  |  +---- path-metric-bounds
       |  |  |  +---- path-metric-bound* [metric-type]
       |  |  |     +---- metric-type    identityref
       |  |  |     +---- upper-bound?   uint64
       |  |  +---- path-affinities
       |  |  |  +---- constraint* [usage]
       |  |  |     +---- usage    identityref
       |  |  |     +---- value?   admin-groups
       |  |  +---- path-srlgs
       |  |     +---- usage?    identityref
       |  |     +---- values*   srlg
       |  +---- optimizations
       |  |  +---- (algorithm)?
       |  |     +--:(metric) {path-optimization-metric}?
       |  |     |  +---- optimization-metric* [metric-type]
       |  |     |  |  +---- metric-type
   identityref
       |  |     |  |  +---- weight?                           uint8
       |  |     |  |  +---- explicit-route-exclude-objects
       |  |     |  |  |  +---- route-object-exclude-object* [index]
       |  |     |  |  |     +---- index             uint32
       |  |     |  |  |     +---- (type)?
       |  |     |  |  |        +--:(numbered)
       |  |     |  |  |        |  +---- numbered-hop
       |  |     |  |  |        |     +---- address?     te-types:te-tp-
   id
       |  |     |  |  |        |     +---- hop-type?    te-hop-type
       |  |     |  |  |        |     +---- direction?   te-link-
   direction
       |  |     |  |  |        +--:(as-number)
       |  |     |  |  |        |  +---- as-number-hop
       |  |     |  |  |        |     +---- as-number?   binary
       |  |     |  |  |        |     +---- hop-type?    te-hop-type
       |  |     |  |  |        +--:(unnumbered)
       |  |     |  |  |        |  +---- unnumbered-hop
       |  |     |  |  |        |     +---- node-id?      te-types:te-
   node-id
       |  |     |  |  |        |     +---- link-tp-id?   te-types:te-
   tp-id
       |  |     |  |  |        |     +---- hop-type?     te-hop-type
       |  |     |  |  |        |     +---- direction?    te-link-
   direction
       |  |     |  |  |        +--:(label)
       |  |     |  |  |           +---- label-hop
       |  |     |  |  |              +---- te-label
       |  |     |  |  |                 +---- (technology)?
       |  |     |  |  |                 |  +--:(generic)
       |  |     |  |  |                 |     +---- generic?     rt-
   types:generalized-label
       |  |     |  |  |                 +---- direction?   te-label-
   direction
       |  |     |  |  +---- explicit-route-include-objects
       |  |     |  |     +---- route-object-include-object* [index]
       |  |     |  |        +---- index             uint32
       |  |     |  |        +---- (type)?
       |  |     |  |           +--:(numbered)
       |  |     |  |           |  +---- numbered-hop
       |  |     |  |           |     +---- address?     te-types:te-tp-
   id
       |  |     |  |           |     +---- hop-type?    te-hop-type
       |  |     |  |           |     +---- direction?   te-link-
   direction
       |  |     |  |           +--:(as-number)
       |  |     |  |           |  +---- as-number-hop
       |  |     |  |           |     +---- as-number?   binary
       |  |     |  |           |     +---- hop-type?    te-hop-type
       |  |     |  |           +--:(unnumbered)
       |  |     |  |           |  +---- unnumbered-hop
       |  |     |  |           |     +---- node-id?      te-types:te-
   node-id
```

```
       |  |     |  |           |     +---- link-tp-id?   te-types:te-
   tp-id
       |  |     |  |           |     +---- hop-type?     te-hop-type
       |  |     |  |           |     +---- direction?    te-link-
   direction
       |  |     |  |           +--:(label)
       |  |     |  |              +---- label-hop
       |  |     |  |                 +---- te-label
       |  |     |  |                    +---- (technology)?
       |  |     |  |                    |  +--:(generic)
       |  |     |  |                    |     +---- generic?     rt-
   types:generalized-label
       |  |     |  |                    +---- direction?   te-label-
   direction
       |  |     |  +---- tiebreakers
       |  |     |     +---- tiebreaker* [tiebreaker-type]
       |  |     |        +---- tiebreaker-type    identityref
       |  |     +--:(objective-function) {path-optimization-objective-
   function}?
       |  |        +---- objective-function
       |  |           +---- objective-function-type?   identityref
       |  +---- requested-metrics* [metric-type]
       |     +---- metric-type    identityref
       +---- synchronization* [synchronization-id]
          +---- synchronization-id    uint32
          +---- svec
          |  +---- relaxable?           boolean
          |  +---- link-diverse?        boolean
          |  +---- node-diverse?        boolean
          |  +---- srlg-diverse?        boolean
          |  +---- request-id-number*   uint32
          +---- svec-constraints
          |  +---- path-metric-bound* [metric-type]
          |     +---- metric-type    identityref
          |     +---- upper-bound?   uint64
          +---- path-srlgs
          |  +---- usage?    identityref
          |  +---- values*   srlg
          +---- exclude-objects
```

```
        |  +---- excludes* [index]
        |     +---- index             uint32
        |     +---- (type)?
        |        +--:(numbered)
        |        |  +---- numbered-hop
        |        |     +---- address?     te-types:te-tp-id
        |        |     +---- hop-type?    te-hop-type
        |        |     +---- direction?   te-link-direction
        |        +--:(as-number)
        |        |  +---- as-number-hop
        |        |     +---- as-number?   binary
        |        |     +---- hop-type?    te-hop-type
        |        +--:(unnumbered)
        |        |  +---- unnumbered-hop
        |        |     +---- node-id?      te-types:te-node-id
        |        |     +---- link-tp-id?   te-types:te-tp-id
        |        |     +---- hop-type?     te-hop-type
        |        |     +---- direction?    te-link-direction
        |        +--:(label)
        |           +---- label-hop
        |              +---- te-label
        |                 +---- (technology)?
        |                 |  +--:(generic)
        |                 |     +---- generic?     rt-
   types:generalized-label
        |                 +---- direction?   te-label-direction
        +---- optimizations
           +---- (algorithm)?
              +--:(metric)
              |  +---- optimization-metric* [metric-type]
              |     +---- metric-type    identityref
              |     +---- weight?        uint8
              +--:(objective-function)
                 +---- objective-function
                    +---- objective-function-type?   identityref
     augment /te:tunnels-rpc/te:output/te:result:
       +--ro response* [response-id]
          +--ro response-id      uint32
          +--ro (response-type)?
```

```
             +--:(no-path-case)
             |  +--ro no-path!
             +--:(path-case)
                +--ro computed-path
                   +--ro path-id?           yang-types:uuid
                   +--ro path-properties
                      +--ro path-metric* [metric-type]
                      |  +--ro metric-type           identityref
                      |  +--ro accumulative-value?   uint64
                      +--ro path-affinities
                      |  +--ro constraint* [usage]
                      |     +--ro usage    identityref
                      |     +--ro value?   admin-groups
                      +--ro path-srlgs
                      |  +--ro usage?    identityref
                      |  +--ro values*   srlg
                      +--ro path-route-objects
                         +--ro path-route-object* [index]
                            +--ro index             uint32
                            +--ro (type)?
                               +--:(numbered)
                               |  +--ro numbered-hop
                               |     +--ro address?     te-types:te-tp-
   id
                               |     +--ro hop-type?    te-hop-type
                               |     +--ro direction?   te-link-
   direction
                               +--:(as-number)
                               |  +--ro as-number-hop
                               |     +--ro as-number?   binary
                               |     +--ro hop-type?    te-hop-type
                               +--:(unnumbered)
                               |  +--ro unnumbered-hop
                               |     +--ro node-id?      te-types:te-
   node-id
                               |     +--ro link-tp-id?   te-types:te-
   tp-id
                               |     +--ro hop-type?     te-hop-type
```

```
                              |     +--ro direction?    te-link-
   direction
                              +--:(label)
                                 +--ro label-hop
                                    +--ro te-label
                                       +--ro (technology)?
                                       |  +--:(generic)
                                       |     +--ro generic?     rt-
   types:generalized-label
                                       +--ro direction?   te-label-
   direction
```

Figure 9 - TE path computation YANG tree

6.2. YANG Module

```
   <CODE BEGINS>file "ietf-te-path-computation@2018-03-02.yang"
   module ietf-te-path-computation {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-te-path-computation";
     // replace with IANA namespace when assigned

     prefix "tepc";

     import ietf-inet-types {
       prefix "inet";
     }

     import ietf-yang-types {
       prefix "yang-types";
     }

     import ietf-te {
       prefix "te";
     }

     import ietf-te-types {
       prefix "te-types";
     }
     organization
       "Traffic Engineering Architecture and Signaling (TEAS)
        Working Group";

     contact
       "WG Web:   <http://tools.ietf.org/wg/teas/>
        WG List:  <mailto:teas@ietf.org>

        WG Chair: Lou Berger
                  <mailto:lberger@labn.net>

        WG Chair: Vishnu Pavan Beeram
                  <mailto:vbeeram@juniper.net>

      ";

     description "YANG model for stateless TE path computation";

     revision "2018-03-02" {
       description "Revision to fix issues #22, 29, 33 and 39";
       reference "YANG model for stateless TE path computation";
     }

     /*
      * Features
      */

     feature stateless-path-computation {
       description
         "This feature indicates that the system supports
          stateless path computation.";
     }


     /*
      * Groupings
      */

     grouping path-info {
       leaf path-id {
         type yang-types:uuid;
         config false;
         description "path-id ref.";
       }
       uses te-types:generic-path-properties;
       description "Path computation output information";
     }

     grouping end-points {
       leaf source {
         type inet:ip-address;
         description "TE tunnel source address.";
       }
       leaf destination {
         type inet:ip-address;
         description "P2P tunnel destination address";
       }
       leaf src-tp-id {
         type binary;
         description "TE tunnel source termination point identifier.";
       }
       leaf dst-tp-id {
         type binary;
         description "TE tunnel destination termination point
   identifier.";
       }
       description "Path Computation End Points grouping.";
     }

     grouping requested-metrics-info {
       description "requested metric";
       list requested-metrics {
         key 'metric-type';
         description "list of requested metrics";
         leaf metric-type {
           type identityref {
             base te-types:path-metric-type;
           }
           description "the requested metric";
         }
       }
     }

     identity svec-metric-type {
       description
         "Base identity for svec metric type";
     }

     identity svec-metric-cumul-te {
       base svec-metric-type;
       description
         "TE cumulative path metric";
     }

     identity svec-metric-cumul-igp {
       base svec-metric-type;
       description
         "IGP cumulative path metric";
     }

     identity svec-metric-cumul-hop {
       base svec-metric-type;
       description
         "Hop cumulative path metric";
     }

     identity svec-metric-aggregate-bandwidth-consumption {
       base svec-metric-type;
       description
         "Cumulative bandwith consumption of the set of synchronized
   paths";
     }

     identity svec-metric-load-of-the-most-loaded-link {
       base svec-metric-type;
       description
         "Load of the most loaded link";
     }

     grouping svec-metrics-bounds_config {
       description "TE path metric bounds grouping for computing a set
   of
         synchronized requests";
       leaf metric-type {
         type identityref {
           base svec-metric-type;
         }
         description "TE path metric type usable for computing a set of
            synchronized requests";
       }
       leaf upper-bound {
         type uint64;
         description "Upper bound on end-to-end svec path metric";
       }
     }

     grouping svec-metrics-optimization_config {
       description "TE path metric bounds grouping for computing a set
   of
         synchronized requests";
       leaf metric-type {
         type identityref {
           base svec-metric-type;
         }
         description "TE path metric type usable for computing a set of
            synchronized requests";
       }
       leaf weight {
         type uint8;
         description "Metric normalization weight";
       }
     }

     grouping svec-exclude {
       description "List of resources to be excluded by all the paths
         in the SVEC";
       container exclude-objects {
         description "resources to be excluded";
         list excludes {
           key index;
           description
             "List of explicit route objects to always exclude
              from synchronized path computation";
           uses te-types:explicit-route-hop;
         }
       }
     }

     grouping synchronization-constraints {
       description "Global constraints applicable to synchronized
         path computation";
       container svec-constraints {
         description "global svec constraints";
         list path-metric-bound {
           key metric-type;
           description "list of bound metrics";
           uses svec-metrics-bounds_config;
         }
       }
       uses te-types:generic-path-srlgs;
       uses svec-exclude;
     }

     grouping synchronization-optimization {
         description "Synchronized request optimization";
       container optimizations {
         description
           "The objective function container that includes
            attributes to impose when computing a synchronized set of
   paths";

         choice algorithm {
           description "Optimizations algorithm.";
           case metric {
             list optimization-metric {
               key "metric-type";
               description "svec path metric type";
               uses svec-metrics-optimization_config;
             }
           }
           case objective-function {
             container objective-function {
               description
                 "The objective function container that includes
                  attributes to impose when computing a TE path";
               uses te-types:path-objective-function_config;
             }
           }
         }
       }
     }

     grouping synchronization-info {
       description "Information for sync";
       list synchronization {
         key "synchronization-id";
         description "sync list";
         leaf synchronization-id {
           type uint32;
           description "index";
         }
         container svec {
           description
            "Synchronization VECtor";
           leaf relaxable {
             type boolean;
             default true;
             description
              "If this leaf is true, path computation process is free
   to ignore svec content.
               otherwise it must take into account this svec.";
           }
           leaf link-diverse {
             type boolean;
             default false;
             description "link-diverse";
           }
           leaf node-diverse {
             type boolean;
             default false;
             description "node-diverse";
           }
           leaf srlg-diverse {
             type boolean;
             default false;
             description "srlg-diverse";
           }
           leaf-list request-id-number {
             type uint32;
             description "This list reports the set of M path
   computation
               requests that must be synchronized.";
           }
         }
         uses synchronization-constraints;
         uses synchronization-optimization;
       }
     }

     grouping no-path-info {
       description "no-path-info";
       container no-path {
         presence "Response without path information, due to failure
           performing the path computation";
         description "if path computation cannot identify a path,
           rpc returns no path.";
       }
     }

     /*
      * Root container
      */
     container paths {
       list path {
         key "path-id";
         config false;
         uses path-info;
         description "List of previous computed paths.";
       }
       description "Root container for path-computation";
     }

     /**
      * AUGMENTS TO TE RPC
      */

     augment "/te:tunnels-rpc/te:input/te:tunnel-info" {
       description "statelessComputeP2PPath input";
       list path-request {
         key "request-id";
         description "request-list";
         leaf request-id {
           type uint32;
           mandatory true;
           description "Each path computation request is uniquely
   identified by the request-id-number.
             It must be present also in rpcs.";
         }
         uses end-points;
         uses te:bidir-assoc-properties;
         uses te-types:path-route-objects;
         uses te-types:generic-path-constraints;
         uses te-types:generic-path-optimization;
         uses requested-metrics-info;
       }
       uses synchronization-info;
     }

     augment "/te:tunnels-rpc/te:output/te:result" {
       description "statelessComputeP2PPath output";
       list response {
         key response-id;
         config false;
         description "response";
         leaf response-id {
           type uint32;
           description
             "The list key that has to reuse request-id-number.";
         }
         choice response-type {
           config false;
           description "response-type";
           case no-path-case {
             uses no-path-info;
           }
           case path-case {
             container computed-path {
               uses path-info;
               description "Path computation service.";
             }
           }
         }
       }
     }
   }
   <CODE ENDS>
```

Figure 10 - TE path computation YANG module

7. Security Considerations

This document describes use cases of requesting Path Computation using YANG models, which could be used at the ABNO Control Interface [RFC7491] and/or between controllers in ACTN [ACTN-frame]. As such, it does not introduce any new security considerations compared to the ones related to YANG specification, ABNO specification and ACTN Framework defined in [RFC6020], [RFC7950], [RFC7491] and [ACTN-frame].

This document also defines common data types using the YANG data modeling language. The definitions themselves have no security impact on the Internet, but the usage of these definitions in concrete YANG modules might have. The security considerations spelled out in the YANG specification [RFC6020] apply for this document as well.

8. IANA Considerations

This section is for further study: to be completed when the YANG model is more stable.

9. References

9.1. Normative References

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC7139] Zhang, F. et al., "GMPLS Signaling Extensions for Control of Evolving G.709 Optical Transport Networks", RFC 7139, March 2014.

[RFC7491] Farrel, A., King, D., "A PCE-Based Architecture for Application-Based Network Operations", RFC 7491, March 2015.

[RFC7926] Farrel, A. et al., "Problem Statement and Architecture for Information Exchange Between Interconnected Traffic Engineered Networks", RFC 7926, July 2016.

[RFC7950] Bjorklund, M., "The YANG 1.1 Data Modeling Language", RFC 7950, August 2016.

[TE-TOPO] Liu, X. et al., "YANG Data Model for TE Topologies", draft-ietf-teas-yang-te-topo, work in progress.

[TE-TUNNEL] Saad, T. et al., "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", draft-ietf-teas-yang-te, work in progress.

[ACTN-Frame] Ceccarelli, D., Lee, Y. et al., "Framework for Abstraction and Control of Traffic Engineered Networks" draft-ietf-actn-framework, work in progress.

[ITU-T G.709-2016] ITU-T Recommendation G.709 (06/16), "Interface for the optical transport network", June 2016

9.2. Informative References

[RFC4655] Farrel, A. et al., "A Path Computation Element (PCE)-Based Architecture", RFC 4655, August 2006.

[RFC5541] Le Roux, JL. et al., " Encoding of Objective Functions in the Path Computation Element Communication Protocol (PCEP)", RFC 5541, June 2009.

[RFC7446] Lee, Y. et al., "Routing and Wavelength Assignment Information Model for Wavelength Switched Optical Networks", RFC 7446, February 2015.

[OTN-TOPO] Zheng, H. et al., "A YANG Data Model for Optical Transport Network Topology", draft-ietf-ccamp-otn-topo-yang, work in progress.

[ACTN-Info] Lee, Y., Belotti, S., Dhody, D., Ceccarelli, D., “Information Model for Abstraction and Control of Transport Networks”, draft-leebelotti-actn-info, work in progress.

[PCEP-Service-Aware] Dhody, D. et al., "Extensions to the Path Computation Element Communication Protocol (PCEP) to compute service aware Label Switched Path (LSP)", draft-ietf-pce-pcep-service-aware, work in progress.

10. Acknowledgments

The authors would like to thank Igor Bryskin and Xian Zhang for participating in discussions and providing valuable insights.

The authors would like to thank the authors of the TE Tunnel YANG model [TE-TUNNEL], in particular Igor Bryskin, Vishnu Pavan Beeram, Tarek Saad and Xufeng Liu, for their inputs to the discussions and support in having consistency between the Path Computation and TE Tunnel YANG models.

This document was prepared using 2-Word-v2.0.template.dot.

Appendix A. Examples of dimensioning the “detailed connectivity matrix”

In the following table, a list of the possible constraints, associated with their potential cardinality, is reported.

The maximum number of potential connections to be computed and reported is, in first approximation, the multiplication of all of them.

Constraint Cardinality

End points N(N-1)/2 if connections are bidirectional (OTN and WDM), N(N-1) for unidirectional connections.

Bandwidth In WDM networks, bandwidth values are expressed in GHz.

On fixed-grid WDM networks, the central frequencies are on a 50GHz grid and the channel width of the transmitters are typically 50GHz such that each central frequency can be used, i.e., adjacent channels can be placed next to each other in terms of central frequencies.

On flex-grid WDM networks, the central frequencies are on a 6.25GHz grid and the channel width of the transmitters can be multiples of 12.5GHz.

For fixed-grid WDM networks typically there is only one possible bandwidth value (i.e., 50GHz) while for flex-grid WDM networks typically there are 4 possible bandwidth values (e.g., 37.5GHz, 50GHz, 62.5GHz, 75GHz).

In OTN (ODU) networks, bandwidth values are expressed as pairs of ODU type and, in case of ODUflex, ODU rate in bytes/sec as described in section 5 of [RFC7139].

For "fixed" ODUk types, 6 possible bandwidth values are possible (i.e., ODU0, ODU1, ODU2, ODU2e, ODU3, ODU4).

For ODUflex(GFP), up to 80 different bandwidth values can be specified, as defined in Table 7-8 of [ITU-T G.709-2016].

For other ODUflex types, like ODUflex(CBR), the number of possible bandwidth values depends on the rates of the clients that could be mapped over these ODUflex types, as shown in Table 7.2 of [ITU-T G.709-2016], which in theory could be a countinuum of values. However, since different ODUflex bandwidths that use the same number of TSs on each link along the path are equivalent for path computation purposes, up to 120 different bandwidth ranges can be specified.

Ideas to reduce the number of ODUflex bandwidth values in the detailed connectivity matrix, to less than 100, are for further study.

Bandwidth specification for ODUCn is currently for further study but it is expected that other bandwidth values can be specified as integer multiples of 100Gb/s.

In IP we have bandwidth values in bytes/sec. In principle, this is a countinuum of values, but in practice we can identify a set of bandwidth ranges, where any bandwidth value inside the same range produces the same path.
The number of such ranges is the cardinality, which depends on the topology, available bandwidth and status of the network. Simulations (Note: reference paper submitted for publication) show that values for medium size topologies (around 50-150 nodes) are in the range 4-7 (5 on average) for each end points couple.

Metrics IGP, TE and hop number are the basic objective metrics defined so far. There are also the 2 objective functions defined in [RFC5541]: Minimum Load Path (MLP) and Maximum Residual Bandwidth Path (MBP). Assuming that one only metric or objective function can be optimized at once, the total cardinality here is 5.

With [PCEP-Service-Aware], a number of additional metrics are defined, including Path Delay metric, Path Delay Variation metric and Path Loss metric, both for point-to-point and point-to-multipoint paths. This increases the cardinality to 8.

Bounds Each metric can be associated with a bound in order to find a path having a total value of that metric lower than the given bound. This has a potentially very high cardinality (as any value for the bound is allowed). In practice there is a maximum value of the bound (the one with the maximum value of the associated metric) which results always in the same path, and a range approach like for bandwidth in IP should produce also in this case the cardinality. Assuming to have a cardinality similar to the one of the bandwidth (let say 5 on average) we should have 6 (IGP, TE, hop, path delay, path delay variation and path loss; we don't consider here the two objective functions of [RFC5541] as they are conceived only for optimization)*5 = 30 cardinality.

Technology constraints For further study

Priority   We have 8 values for setup priority, which is used in path computation to route a path using free resources and, where no free resources are available, resources used by LSPs having a lower holding priority.

Local prot   It's possible to ask for a local protected service, where all the links used by the path are protected with fast reroute (this is only for IP networks, but line protection schemas are available on the other technologies as well). This adds an alternative path computation, so the cardinality of this constraint is 2.

Administrative Colors   Administrative colors (aka affinities) are typically assigned to links but when topology abstraction is used affinity information can also appear in the detailed connectivity matrix.

There are 32 bits available for the affinities. Links can be tagged with any combination of these bits, and path computation can be constrained to include or exclude any or all of them. The relevant cardinality is 3 (include-any, exclude-any, include-all) times 2^32 possible values. However, the number of possible values used in real networks is quite small.

Included Resources

A path computation request can be associated to an ordered set of network resources (links, nodes) to be included along the computed path. This constraint would have a huge cardinality as in principle any combination of network resources is possible. However, as far as the Orchestrator doesn't know details about the internal topology of the domain, it shouldn't include this type of constraint at all (see more details below).

Excluded Resources

A path computation request can be associated to a set of network resources (links, nodes, SRLGs) to be excluded from the computed path. Like for included resources, this constraint has a potentially very high cardinality, but, once again, it can't be actually used by the Orchestrator, if it's not aware of the domain topology (see more details below).

As discussed above, the Orchestrator can specify include or exclude resources depending on the abstract topology information that the domain controller exposes:

o In case the domain controller exposes the entire domain as a single abstract TE node with his own external terminations and connectivity matrix (whose size we are estimating), no other topological details are available, therefore the size of the connectivity matrix only depends on the combination of the constraints that the Orchestrator can use in a path computation request to the domain controller. These constraints cannot refer to any details of the internal topology of the domain, as those details are not known to the Orchestrator and so they do not impact size of connectivity matrix exported.

o Instead in case the domain controller exposes a topology including more than one abstract TE nodes and TE links, and their attributes (e.g. SRLGs, affinities for the links), the Orchestrator knows these details and therefore could compute a path across the domain referring to them in the constraints. The connectivity matrixes to be estimated here are the ones relevant to the abstract TE nodes exported to the Orchestrator. These connectivity matrixes and therefore theirs sizes, while cannot depend on the other abstract TE nodes and TE links, which are external to the given abstract node, could depend to SRLGs (and other attributes, like affinities) which could be present also in the portion of the topology represented by the abstract nodes, and therefore contribute to the size of the related connectivity matrix.

We also don't consider here the possibility to ask for more than one path in diversity or for point-to-multi-point paths, which are for further study.

Considering for example an IP domain without considering SRLG and affinities, we have an estimated number of paths depending on these estimated cardinalities:

Endpoints = N*(N-1), Bandwidth = 5, Metrics = 6, Bounds = 20, Priority = 8, Local prot = 2

The number of paths to be pre-computed by each IP domain is therefore 24960 * N(N-1) where N is the number of domain access points.

This means that with just 4 access points we have nearly 300000 paths to compute, advertise and maintain (if a change happens in the domain, due to a fault, or just the deployment of new traffic, a substantial number of paths need to be recomputed and the relevant changes advertised to the upper controller).

This seems quite challenging. In fact, if we assume a mean length of 1K for the json describing a path (a quite conservative estimate), reporting 300000 paths means transferring and then parsing more than 300 Mbytes for each domain. If we assume that 20% (to be checked) of this paths change when a new deployment of traffic occurs, we have 60 Mbytes of transfer for each domain traversed by a new end-to-end path. If a network has, let say, 20 domains (we want to estimate the load for a non-trivial domain setup) in the beginning a total initial transfer of 6Gigs is needed, and eventually, assuming 4-5 domains are involved in mean during a path deployment we could have 240-300 Mbytes of changes advertised to the higher order controller.

Further bare-bone solutions can be investigated, removing some more options, if this is considered not acceptable; in conclusion, it seems that an approach based only on connectivity matrix is hardly feasible, and could be applicable only to small networks with a limited meshing degree between domains and renouncing to a number of path computation features.

Contributors

Dieter Beller
Nokia
Email: dieter.beller@nokia.com

Gianmarco Bruno
Ericsson
Email: gianmarco.bruno@ericsson.com

Francesco Lazzeri
Ericsson
Email: francesco.lazzeri@ericsson.com

Young Lee
Huawei
Email: leeyoung@huawei.com

Carlo Perocchio
Ericsson
Email: carlo.perocchio@ericsson.com

Authors’ Addresses

Italo Busi (Editor)
Huawei
Email: italo.busi@huawei.com

Sergio Belotti (Editor)
Nokia
Email: sergio.belotti@nokia.com

Victor Lopez
Telefonica
Email: victor.lopezalvarez@telefonica.com

Oscar Gonzalez de Dios
Telefonica
Email: oscar.gonzalezdedios@telefonica.com

Anurag Sharma
Google
Email: ansha@google.com

Yan Shi
China Unicom
Email: shiyan49@chinaunicom.cn

Ricard Vilalta
CTTC
Email: ricard.vilalta@cttc.es

Karthik Sethuraman
NEC
Email: karthik.sethuraman@necam.com

Michael Scharf
Nokia
Email: michael.scharf@nokia.com

Daniele Ceccarelli
Ericsson
Email: daniele.ceccarelli@ericsson.com