TEAS Working Group   Italo Busi (Ed.)
Internet Draft   Huawei
Intended status: Standard Track   Sergio Belotti (Ed.)
Expires: May 2019   Nokia
                                            Victor Lopez
                                  Oscar Gonzalez de Dios
                                            Telefonica
                                          Anurag Sharma
                                                 Google
                                                Yan Shi
                                           China Unicom
                                         Ricard Vilalta
                                                   CTTC
                                     Karthik Sethuraman
                                                    NEC

                                       November 4, 2018

              Yang model for requesting Path Computation
              draft-ietf-teas-yang-path-computation-04.txt

Status of this Memo

This Internet-Draft will expire on May 4, 2019.

Copyright Notice

Abstract

There are scenarios, typically in a hierarchical SDN context, where the topology information provided by a TE network provider may not be sufficient for its client to perform end-to-end path computation. In these cases the client would need to request the provider to calculate some (partial) feasible paths.

This document defines a YANG data model for a stateless RPC to request path computation. This model 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, where
the topology information provided by a TE network provider may not
be sufficient for its client to perform end-to-end path computation.
In these cases the client would need to request the provider to
calculate some (partial) feasible paths, complementing his topology
knowledge, to make his end-to-end path computation feasible.

This type of scenarios can be applied to different interfaces in
different reference architectures:

o ABNO control interface [RFC7491], in which an Application Service
   Coordinator can request ABNO controller to take in charge path
   calculation (see Figure 1 in [RFC7491]).

o ACTN [RFC8453], where a 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). [RFC8454] 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 assumes that the client of the YANG data model
defined in this document may not implement 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 a client needs to
request underlying SDN controllers for path computation.

The presented uses cases have been grouped, depending on the
different underlying topologies: a) Packet-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).

```
                        +----------------+
                        |                |
                        | Packet/Optical |
                        |  Coordinator   |
                        |                |
                        +---+------+-----+
                            |      |
                +-----------+      |
                |              +-----------+
          +------V-----+                  |
          |            |           +------V-----+
          | Packet     |           |            |
          | Network    |           | Optical    |
          | Controller |           | Network    |
          |            |           | Controller |
          +------+-----+           +-------+----+
                 |                         |
          .........V.........................           |
          :           Packet Network       :    |
     +----+                        +----+   |
     | R1 |= = = = = = = = = = = = =| R2 |   |
     +-+--+                        +--+-+   |
       | :                          : |    |
       | :........................... : |    |
       |                              |    |
       |             +-----+          |    |
       |   ..........| Opt |........   |    |
       |   :         |  C  |       :   |    |
       |   :         /+--+--+\      :   |    |
       |   :        /   |    \     :   |    |
       |   :       /    |     \    :   |    |
       |  +-----+ /   +--+--+   \ +-----+   |    |
       |  | Opt |/    | Opt |    \| Opt |   |    |
      +---|  A  |     |  D  |     |  B  |---+    |
         +-----+\    +--+--+    /+-----+        |
          :      \     |      /    :            |
          :       \    |     /     :            |
          :        \ +--+--+  / Optical<---------+
          :         \| Opt |/  Network:
          :..........|  E  |..........:
                     +-----+
```
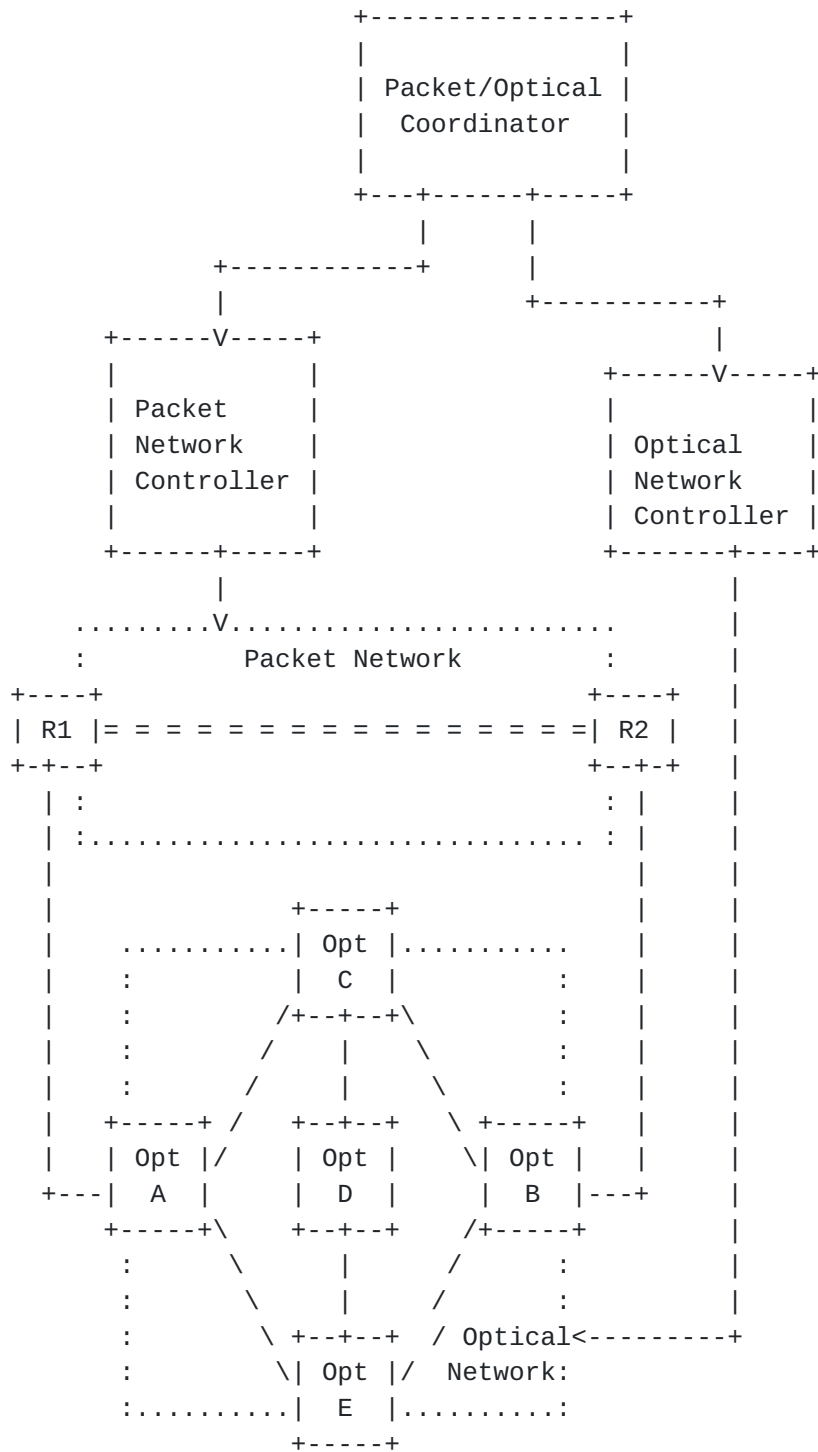
                Figure 1 - Packet/Optical Integration Use Case

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 could be to represent the whole
optical network as one "virtual node" with "virtual ports" connected
to the access links, as shown in Figure 2.

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          ,
          ,                                   +----+        , .
         ,                                    |    |          ,
        ,                                     | R2 |      ,     .
       ,    +----+  +----------- +       /+----+     ,
      ,     |    |  | |          |/-----/ / /         ,        .
      ,     | R1 |--O VP1     VP4 O          / /        ,
     ,      |    |\ |           | /----/ /         ,        .
      ,     +----+ \|           |/       /         ,
     ,       /      O VP2     VP5 O       /         ,         .
      ,     /       |           | +----+       ,
     ,     /        |           | |    |   ,              .
     ,     /        O VP3     VP6 O--| R4 |  ,
    ,   +----+ /-----/|_____|  +----+ ,              .
     ,   |    |/      +----------- +        ,
    ,    | R3 |                             ,            .
     ,   +----+                       '''''''''''''''
   '''''''''''''''''''''''''''''''''''''''''''''''           '.
   . Packet Network Controller view           +----+        ,
     only packet nodes and packet links       |    |      , .
   . with access links to the optical network | R2 |       ,
            ,   +----+                         /+----+     ,    .
   .             ,  |    |            /-----/ / /      ,
             ,      | R1 |---             / /        ,       .
   .          ,     +----+\            /----/ /     ,
             ,       /    \          /      /        ,        .
   .       ,        /                     /        ,
         ,         /                +----+     ,          .
   .     ,        /                 |    |   ,
        ,        /               ---| R4 |  ,            .
   .   ,     +----+ /-----/          +----+ ,
      ,     |    |/                          ,          .
   . ,      | R3 |                          ,
      ,     +----+                    '''''''''''''''.
   .''''''''''''''''''''''''''''''''''''''''''''          ,
      Optical Network Controller view                    , .
   . only optical nodes,         +--+                    ,
     optical links and         /|OF|                    ,    .
   . access links from the  +--++--+              /       ,
     packet network         |OA|     \      /-----/ /      ,      .
   .           ,          ---+--+--\  +--+/       /       ,
             ,              \  |  \  \-|OE|-------/       ,       .
   .         ,               \  |   \ /-+--+
           ,                  \+--+  X     |             ,       .
```
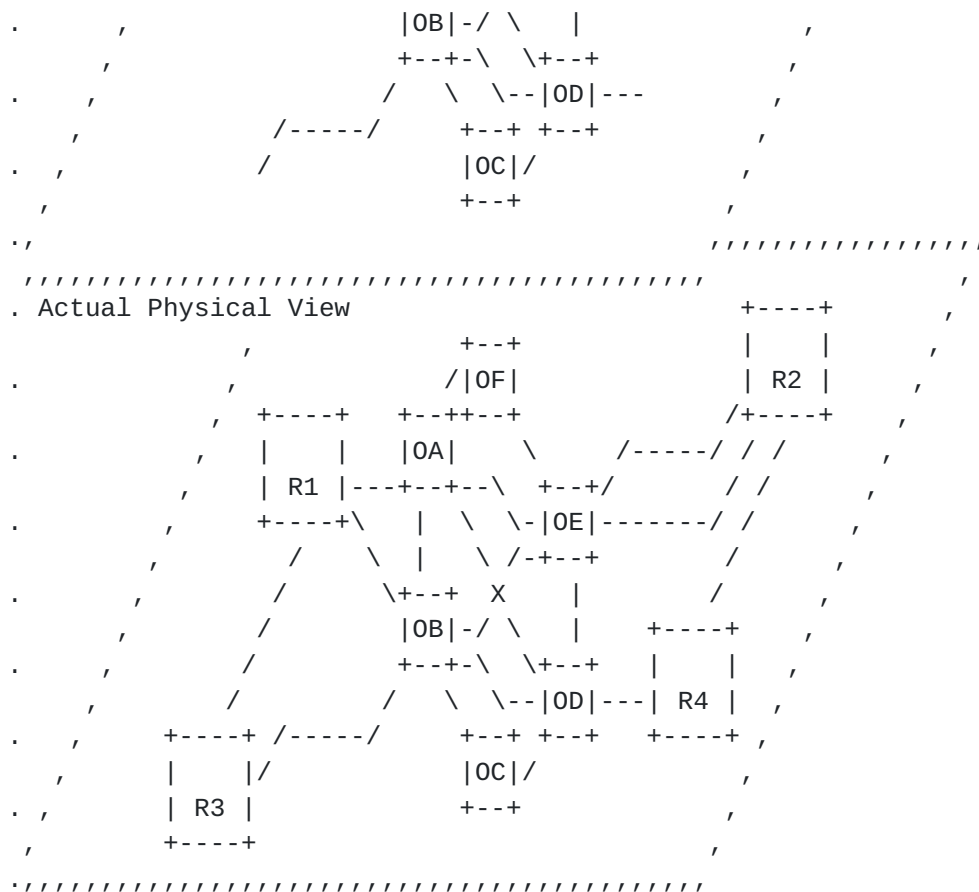
```
.      ,                |OB|-/ \    |                ,
    ,                  +--+-\  \+--+          ,           .
.    ,                /   \  \--|OD|---          ,
    ,          /-----/      +--+ +--+             ,        .
.  ,          /              |OC|/              ,
   ,                         +--+              ,          .
 .,                                           ///////////////
  ///////////////////////////////////////////           ,
 . Actual Physical View                         +----+      ,
              ,                +--+              |    |      ,
.              ,             /|OF|              | R2 |     ,
          ,  +----+    +--++--+              /+----+      ,
.          ,  |    |   |OA|       \     /-----/ / /        ,
          ,   | R1 |---+--+--\  +--+/         / /        ,
.          ,   +----+\   |   \  \-|OE|-------/ /         ,
          ,       /   \  |   \ /-+--+         /          ,
.          ,     /       \+--+  X     |        /         ,
          ,     /         |OB|-/ \    |    +----+        ,
.          ,    /          +--+-\  \+--+    |    |     ,
          ,    /          /   \  \--|OD|---| R4 |     ,
.    ,     +----+ /-----/      +--+ +--+    +----+ ,
      ,    |    |/             |OC|/             ,
.  ,       | R3 |              +--+              ,
   ,       +----+                               ,
 ./////////////////////////////////////////////
```
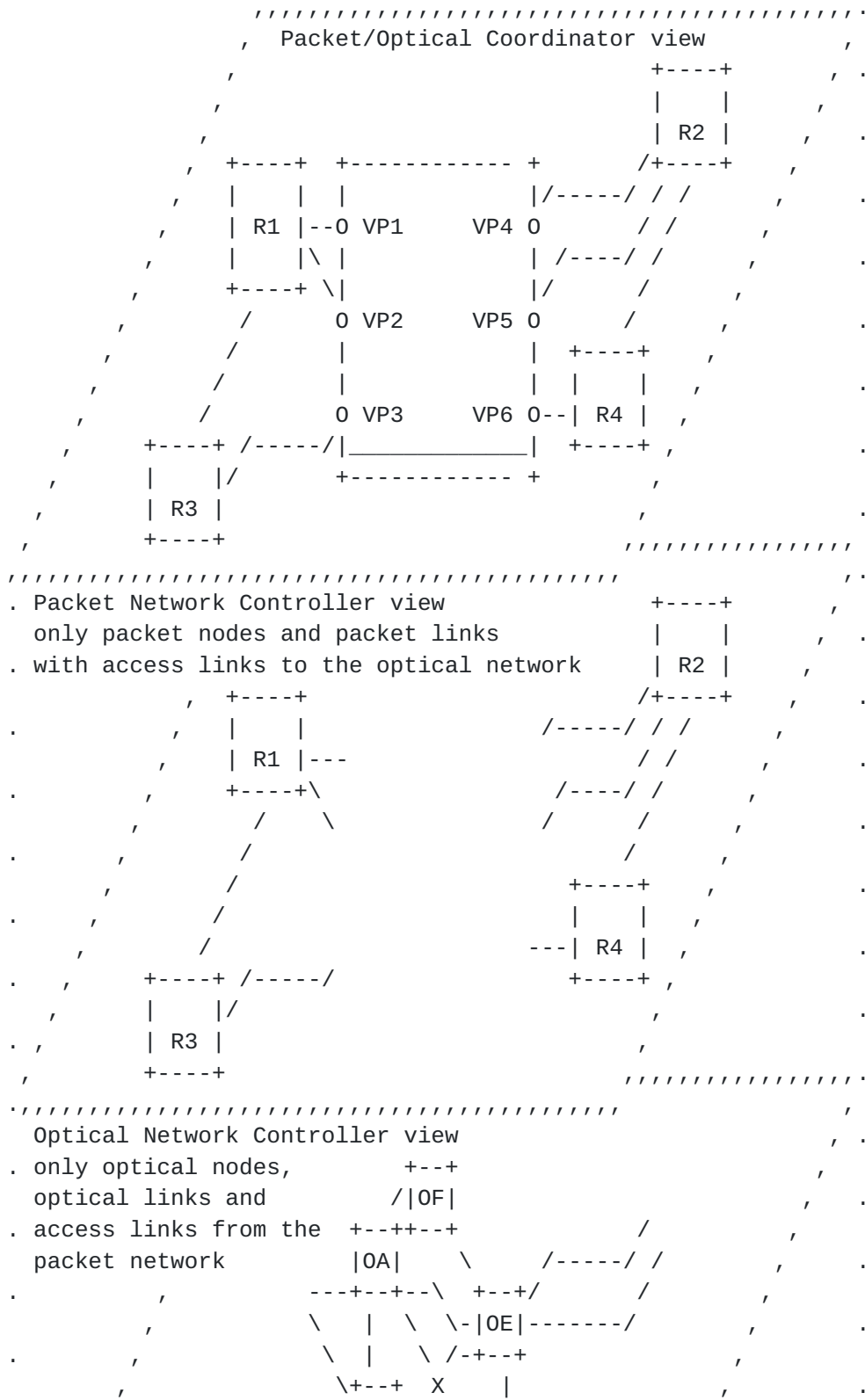
            Figure 2 - Packet and Optical Topology Abstractions

   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 end-to-end path
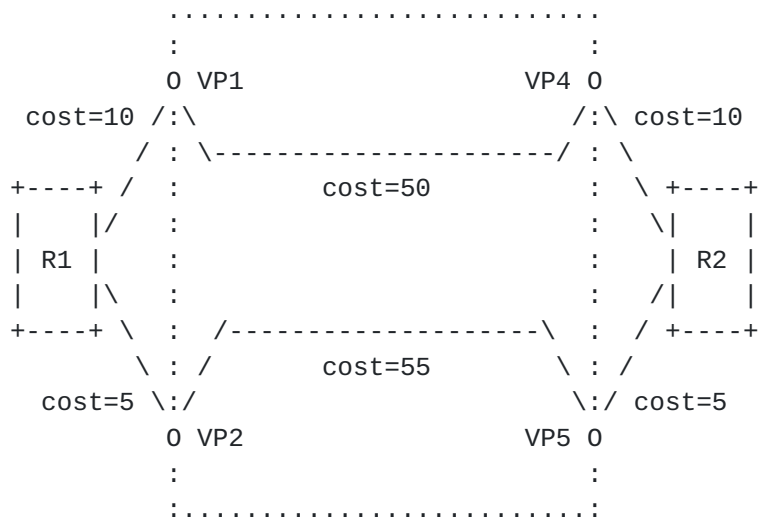   crossing optical network.

```
              ............................
              :                          :
              O VP1                 VP4 O
     cost=10 /:\                       /:\ cost=10
            / : \--------------------/ : \
      +----+ /  :       cost=50       :  \ +----+
      |    |/   :                     :   \|    |
      | R1 |    :                     :    | R2 |
      |    |\   :                     :   /|    |
      +----+ \  :  /-------------------\  :  / +----+
            \ : /        cost=55        \ : /
      cost=5 \:/                         \:/ cost=5
              O VP2                 VP5 O
              :                          :
              :..........................:
```

                  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.
   This is further discussed in section 3.3.

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.

```
                         +--------------+
                         | Multi-domain |
                         | Controller   |
                         +---+------+---+
                             |      |
                  +------------+     |
                  |               +-----------+
            +------V-----+               |
            |            |               |
            | TE Domain  |         +------V-----+
            | Controller |         |            |
            |     1      |         | TE Domain  |
            +------+-----+         | Controller |
                   |               |     2      |
                   |               +------+-----+
            .........V..........          |
            :                 :           |
          +-----+             :           |
          |     |             :           |
          |  X  |             :    .........V..........
          |     |             :    :                  :
          +-----+       +-----+    +-----+             :
            :           |     |    |     |             :
        +-----+  +-----+ /|     |    |     |\ +-----+    +-----+
        |     |  |     | |/ +-----+    +-----+ \| |     |    |     |
        |  A  |--|  B  |    :          :    |  G  |----|  H  |
        |     |  |     | |\  :          : /| |     |    |     |
        +-----+  +-----+ \+-----+    +-----+/ +-----+    +-----+
          :           |     |    |     |             :
          :           |  D  |------|  F  |             :
          :           |     |    |     |         +-----+
          :           +-----+    +-----+         |     |
          :             :          :             |  Y  |
          :             :          :             |     |
          :   Domain 1  :          : Domain 2    +-----+
          :.............:          :.............:
```
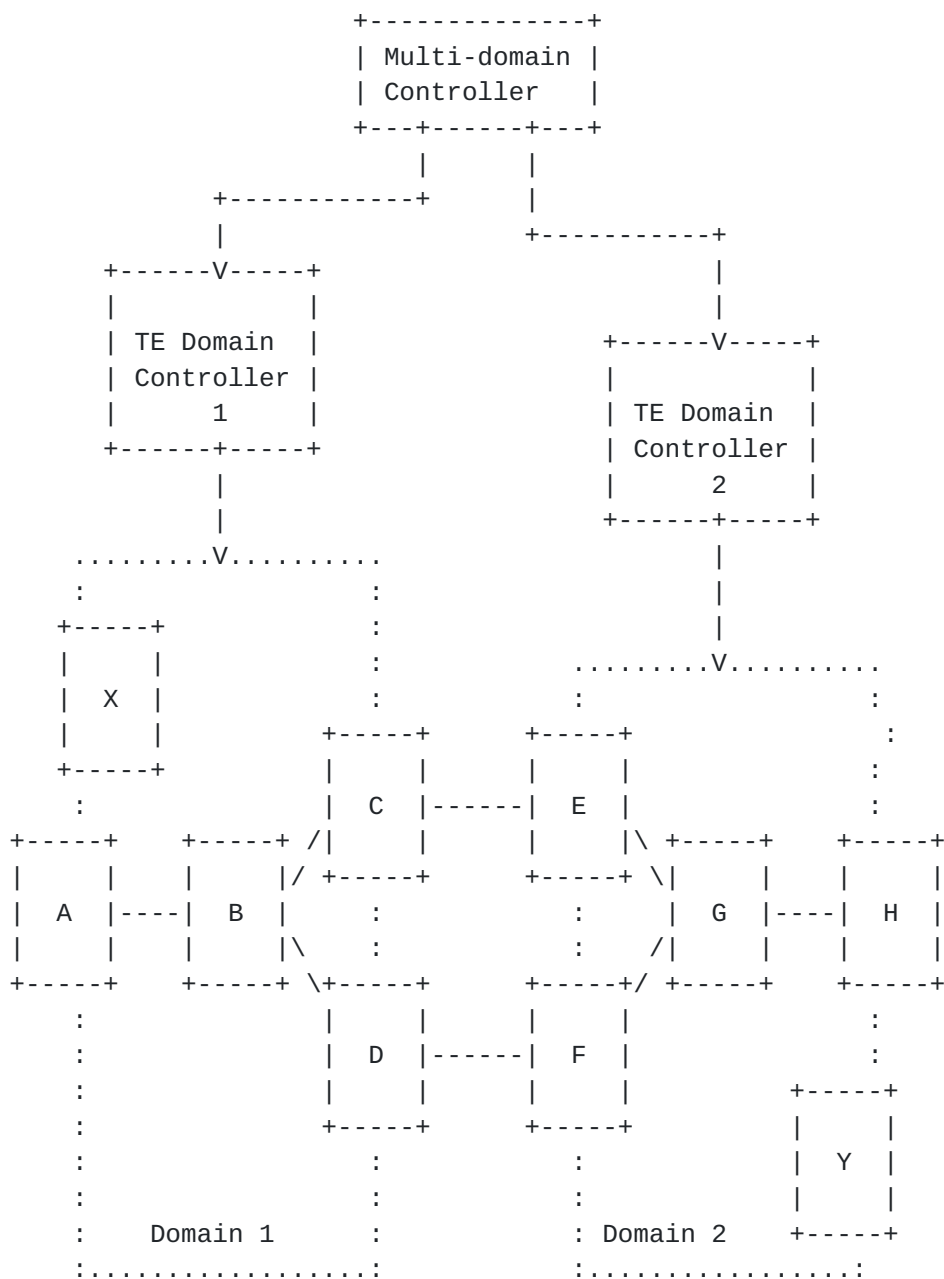
                 Figure 4 - Multi-domain multi-link interconnection

   In order to setup an end-to-end multi-domain TE path (e.g., between
   nodes A and H), the multi-domain controller 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), the multi-domain controller 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 multi-domain controller 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 multi-domain controller 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 multi-domain controller
    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 multi-domain controller)

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 TE domain controllers to limit the
number of potential optimal end-to-end paths and then request path
computation to fewer TE 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.

```
                      +--------------+
                      | Cloud Network|
                      | Orchestrator |
                      +--------------+
                        |  |  |   |
          +------------+  |  |  +-----------------------+
          |               |  +-----------------+        |
          |         +--------V---+              |        |
          |         |            |              |        |
          |         | TE Network |              |        |
     +------V-----+ | Controller |       +------V-----+  |
     | DC         | +-----------+        | DC         |  |
     | Controller |     |                | Controller |  |
     +-----------+      |    +-----+     +-----------+  |
         |         ....V...|    |........        |       |
         |         :       | P  |      :         |       |
     .....V.....   :      /+-----+\     :    .....V.....  |
     :         : +-----+ /   |    \ +-----+  :        :  |
     :  DC1 || : |     |/    |     \|     |  :  DC2 || :  |
     :   ||||----| PE1 |     |      | PE2 |----   ||||  :  |
     : _||||||| : |    |\    |     /|     |  : _|||||| :  |
     :         : +-----+ \ +-----+ / +-----+  :        :  |
     :.........:   :      \|    |/      :    :.........:  |
                   :.......| PE3 |.......:                |
                           |     |                        |
                         +-----+           +---------V--+
                         .....|.....       | DC         |
                         :         :       | Controller |
                         :  DC3 || :       +-----------+
                         :   |||| :               |
                         : _|||||| <-----------------+
                         :         :
                         :.........:
```
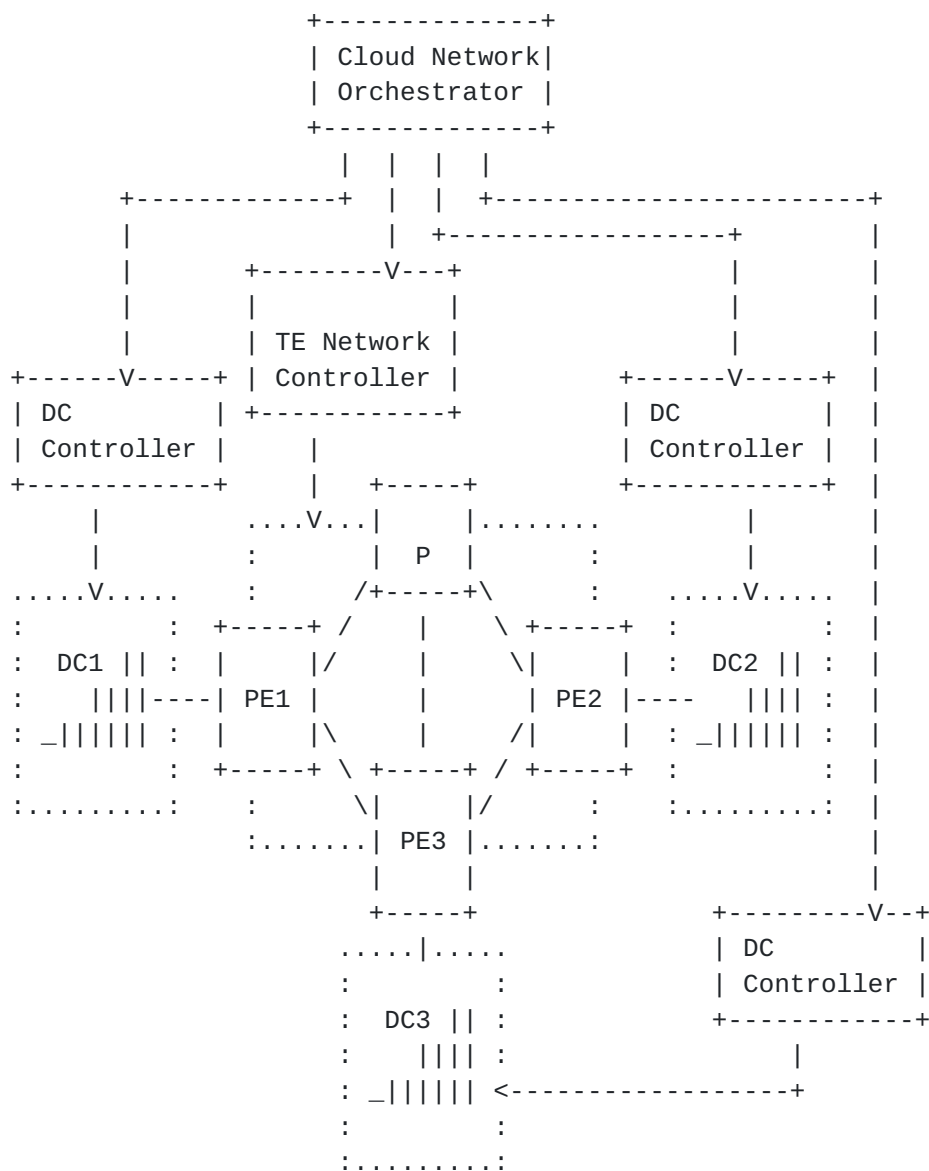
                 Figure 5 - Data Center Interconnection Use Case

   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 network 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 network orchestrator can request to the TE network 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

The YANG data model for requesting path computation is 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].

There are many benefits in aligning the data model used for path computation requests with the YANG data models used for TE topology information and for TE Tunnels configuration and management:

o There is no need for an error-prone mapping or correlation of information.

o It is possible to use the same endpoint identifiers in path computation requests and in the topology modeling.

o The attributes used for path computation constraints are the same as those used when setting up a TE Tunnel.

[3.1.2](). Benefits of a single interface

The system integration effort is typically lower if a single,
consistent interface is used by controllers, 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.

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 its client thus allowing a PCE available within its
client to perform multi-domain path computation by its own, without
requesting path computations to the underlying SDN controllers.

In case the client 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 a client to request underlying
SDN controllers for path computation even in case it 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 client, to
take optimal path computation decisions by its own versus sending

too many requests to underlying SDN Domain Controllers to compute 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".

The concept of a "detailed connectivity matrix" is defined in [TE-
TOPO] to provide specific TE attributes (e.g., delay, SRLGs and
summary TE metrics) as an extension of the "basic connectivity
matrix", which is based on the "connectivity matrix" defined in
[RFC7446].

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

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).

```
                 ............................
                 :  /-------------------\  :
                 : /       cost=65        \ :
                 :/    available-bw=10G    \:
                 O VP1                  VP4 O
          cost=10 /:\                    /:\ cost=10
              / : \--------------------/ : \
         +----+ /  :         cost=50       :  \ +----+
         |    |/   :     available-bw=2G   :   \|    |
         | R1 |    :                       :    | R2 |
         |    |\   :                       :   /|    |
         +----+ \  :   /-------------------\  :  / +----+
              \ : /        cost=55         \ : /
         cost=5 \:/    available-bw=3G      \:/ cost=5
                 O VP2                  VP5 O
                 :                         :
                 :.........................:
```
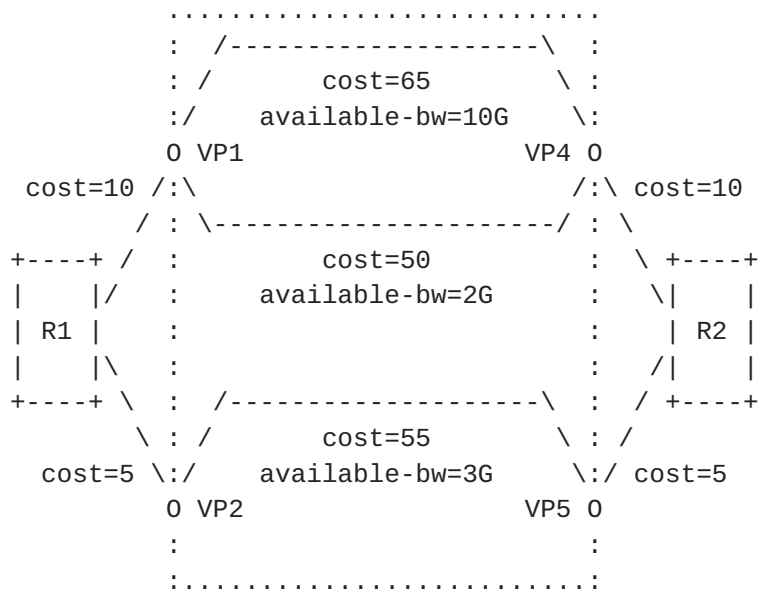
Figure 6 - Packet/Optical Path Computation Example with multiple
choices

Reporting all the information, as in Figure 6, using the "detailed
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 "basic 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 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 connectivity matrix" while not
changing the feasibility of the path.

The "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 "detailed
connectivity matrix" updated which means that there another tradeoff
between the accuracy (i.e., providing "all" the information that
might be needed by the client'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
client's PCE computes paths using not updated information.

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

Instead, if the information in the "detailed 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 client'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 client'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.

Using the approach proposed in this document, the client, 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, representing the abstract view of 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 multi-domain controller 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:

o Multi-domain controller'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

o 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

o So what the multi-domain 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
controller 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
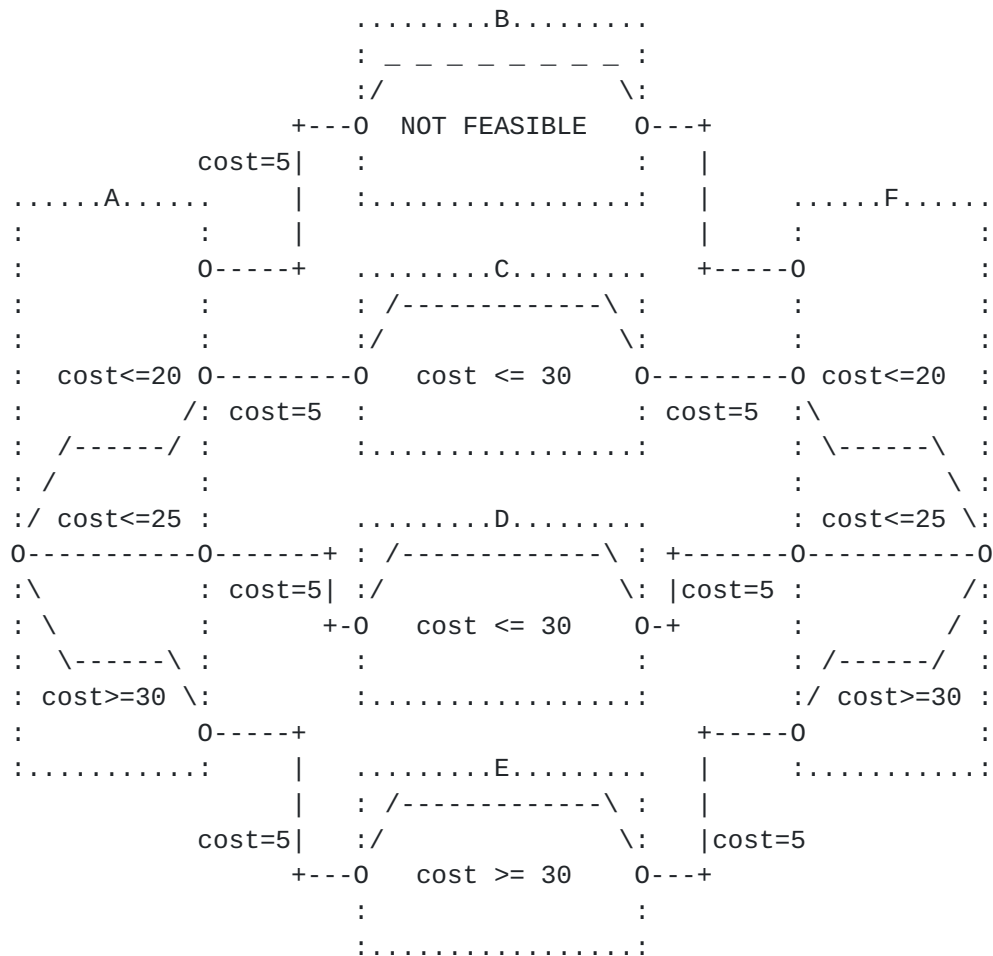be selected between domains B, C, D and E.

```
                        .........B.........
                        : _ _ _ _ _ _ _ :
                        :/               \:
                    +---O   NOT FEASIBLE   O---+
              cost=5|    :               :    |
   ......A......     |   :................:    |      ......F......
   :          :      |                         |    :           :
   :          O-----+   .........C.........    +-----O          :
   :          :         : /-------------\ :         :           :
   :          :         :/               \:         :           :
   :  cost<=20 O---------O   cost <= 30    O---------O cost<=20  :
   :          /: cost=5  :                 : cost=5  :\          :
   :  /------/ :         :................:          : \------\  :
   : /        :                                      :        \ :
   :/ cost<=25 :        .........D.........          : cost<=25 \:
   O-----------O-------+ : /-------------\ : +-------O-----------O
   :\          : cost=5| :/               \: |cost=5 :          /:
   : \         :       +-O    cost <= 30    O-+       :         / :
   :  \------\ :         :                 :          : /------/  :
   : cost>=30 \:         :................:           :/ cost>=30 :
   :          O-----+                         +-----O           :
   :..........:     |    .........E.........    |     :.........:
                    |    : /-------------\ :    |
              cost=5|    :/               \:    |cost=5
                    +---O    cost >= 30    O---+
                        :                 :
                        :................:
```

        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 Multi-domain controller 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
   Multi-domain controller 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 Multi-domain controller 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 Multi-domain controller 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).

```
                    .........B.........
                    :                 :
                 +---O             O---+
  ......A......  |   :...............: |     ......F......
  :           :  |   |                 |   :           :
  :        O-----+   .........C.........   +-----O     :
  :        :     : /-------------\ :       :     :
  :        :     :/               \:       :     :
  : cost=15 O---------O    cost = 25    O---------O  cost=10  :
  :       /: cost=5  :                 : cost=5  :\         :
  :  /------/ :        :...............:        : \------\  :
  : /        :                                  :        \ :
  :/ cost=10 :        .........D.........        : cost=15 \:
  O----------O-------+ : /-------------\ : +-------O----------O
  :          : cost=5| :/               \: |cost=5 :         :
  :          :     +-O    cost = 15    O-+     :         :
  :          :     :                 :     :         :
  :          :     :...............:     :         :
  :        O-----+                       +-----O         :
  :..........:   |   .........E.........   |   :..........:
                 |   :                 :   |
               +---O                 O---+
                 :...............:
```
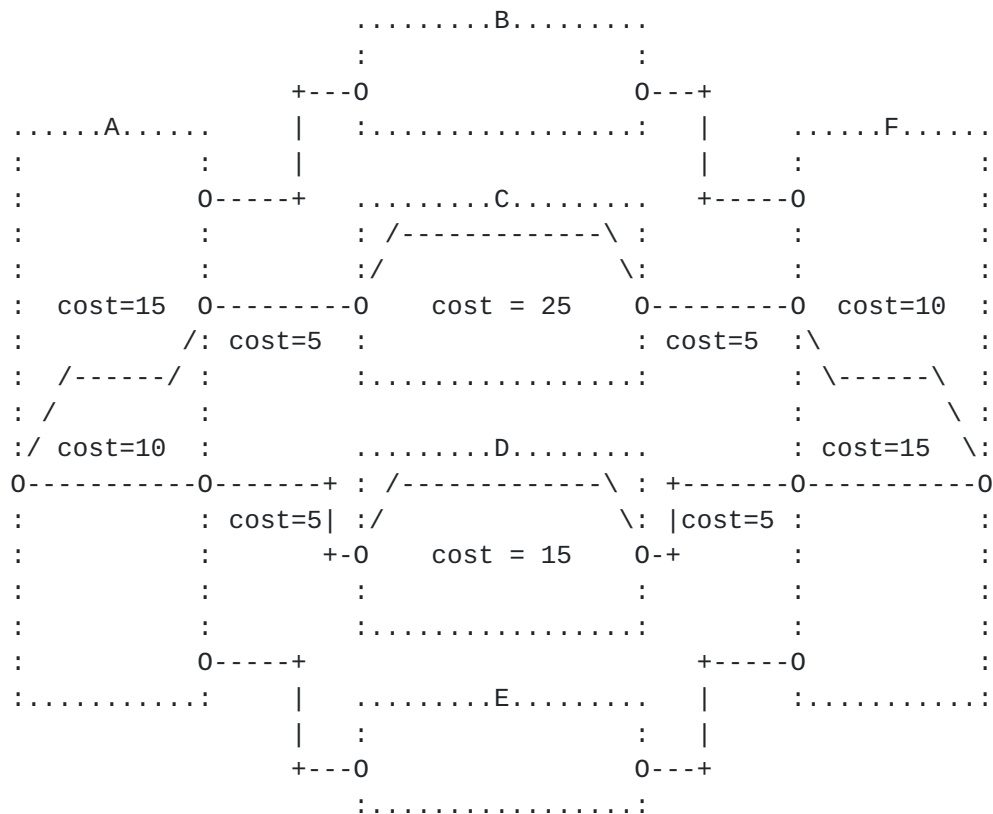
                Figure 8   - Multi-domain with many domains (Path Computation
                              information)

Based on these requests, the Multi-domain controller 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 does not cause major issues in case the time between path
computation and path setup is short (especially if compared with the
time that would be needed to update the information of a very
detailed connectivity matrix).

In most of the cases, there is even no need to guarantee that the
path that has been setup is the exactly same as the path that has
been returned by path computation, especially if has the same or
even better metrics. Depending on the abstraction level applied by
the server, the client may also not know the actual computed path.

The most important requirement is that the required global
objectives (e.g., multi-domain path metrics and constraints) are
met. For this reason a path verification phase is necessary to
verify that the actual path that has been setup meets the global
objectives (for example in a multi-domain network, the resulting

end-to-end path meets the required end-to-end metrics and
constraints).

In most of the cases, even if the setup path is not exactly the same
as the path returned by path computation, its metrics and
constraints are "good enough" (the path verification passes
successfully). In the few corner cases where the path verification
fails, it is possible repeat the whole process (path computation,
path setup and path verification).

In case the stateless solution is not sufficient, a stateful
solution, based on "compute-only" TE tunnel, could be used 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, does not guaranteed that because notifications may not
be reliable or delivered on time. Path verification is needed also
when stateful path computation is used.

The stateful path computation has also the following drawbacks:

o Several messages required for any path computation

o Requires persistent storage in the provider controller

o Need for garbage collection for stranded paths

o 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].

In this case, sending a single request for multiple path
computations, instead of sending multiple requests for each path
computation, would reduce the protocol overhead and it would consume
less resources (e.g., threads in the client and server).

In the context of a typical multi-domain TE network, there could
multiple choices for the ingress/egress points of a domain and the
Multi-domain controller 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 Multi-domain controller 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 Multi-domain controller receives a
request to setup a group of multiple end to end connections. The
multi-domain controller 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 the
possibility of not being able to satisfy the diversity requirement.
In this case, it is preferable to compute a sub-optimal primary path
for which a diversely routed secondary path exists.

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
                 ...........
```

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 [RFC5440].

```
    +---- synchronization* [synchronization-id]
       +---- synchronization-id    uint32
       +---- svec
       |  +---- relaxable?          boolean
       |  +---- disjointness?       te-types:te-path-disjointness
       |  +---- request-id-number*  uint32
       +---- svec-constraints
       |  +---- path-metric-bound* [metric-type]
       |     +---- metric-type    identityref
       |     +---- upper-bound?   uint64
       +---- path-srlgs-values
       |  +---- usage?   identityref
       |  +---- values*   srlg
       +---- path-srlgs-names
       |  +---- path-srlgs-name* [usage]
       |     +---- usage        identityref
       |     +---- srlg-name* [name]
       |        +---- name    string
       +---- 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
[RFC5440]:

```
   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-values
                    |  +--ro path-affinities-value* [usage]
                    |     +--ro usage     identityref
                    |     +--ro value?   admin-groups
                    +--ro path-affinity-names
                    |  +--ro path-affinity-name* [usage]
                    |     +--ro usage            identityref
                    |     +--ro affinity-name* [name]
                    |        +--ro name     string
                    +--ro path-srlgs-values
                    |  +--ro usage?    identityref
                    |  +--ro values*   srlg
                    +--ro path-srlgs-names
                    |  +--ro path-srlgs-name* [usage]
```

```
                     |     +--ro usage       identityref
                     |     +--ro srlg-name* [name]
                     |        +--ro name    string
                     +--ro path-route-objects
                     ...........
```

It also allows to request in the input of RPC which information
(metrics, srlg and/or affinities) should be returned:

```
module: ietf-te-path-computation
  augment /te:tunnels-rpc/te:input/te:tunnel-info:
    +---- path-request* [request-id]
    |  +---- request-id              uint32
        ...........
    |  +---- requested-metrics* [metric-type]
    |  |  +---- metric-type    identityref
    |  +---- return-srlgs?           boolean
    |  +---- return-affinities?      boolean
        ...........
```

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 client to compute the best
end-to-end path. If they are missing the client 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
  augment /te:tunnels-rpc/te:input/te:tunnel-info:
    +---- path-request* [request-id]
    |  +---- request-id              uint32
    |  +---- te-topology-identifier
    |  |  +---- provider-id?   te-types:te-global-id
```

```
|  |  +---- client-id?      te-types:te-global-id
|  |  +---- topology-id?    te-types:te-topology-id
|  +---- source?                    inet:ip-address
|  +---- destination?               inet:ip-address
|  +---- src-tp-id?            binary
|  +---- dst-tp-id?            binary
|  +---- bidirectional?        boolean
|  +---- encoding?             identityref
|  +---- switching-type?       identityref
|  +---- explicit-route-objects
|  |  +---- route-object-exclude-always* [index]
|  |  |  +---- index          uint32
|  |  |  +---- (type)?
|  |  |     +--:(num-unnum-hop)
|  |  |     |  +---- num-unnum-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
|  |  |     +--:(as-number)
|  |  |     |  +---- as-number-hop
|  |  |     |     +---- as-number?   binary
|  |  |     |     +---- hop-type?    te-hop-type
|  |  |     +--:(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)?
|  |        +--:(num-unnum-hop)
|  |        |  +---- num-unnum-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
     |  |        +--:(as-number)
     |  |        |  +---- as-number-hop
     |  |        |     +---- as-number?   binary
     |  |        |     +---- hop-type?    te-hop-type
     |  |        +--:(label)
     |  |        |  +---- label-hop
     |  |        |     +---- te-label
     |  |        |        +---- (technology)?
     |  |        |        |  +--:(generic)
     |  |        |        |     +---- generic?
     |  |        |        |             rt-types:generalized-label
     |  |        |        +---- direction?   te-label-direction
     |  |        +--:(srlg)
     |  |           +---- srlg
     |  |              +---- srlg?   uint32
     |  +---- path-constraints
     |  |  +---- te-bandwidth
     |  |  |  +---- (technology)?
     |  |  |     +--:(generic)
     |  |  |        +---- generic?   te-bandwidth
     |  |  +---- setup-priority?         uint8
     |  |  +---- hold-priority?          uint8
     |  |  +---- signaling-type?         identityref
     |  |  +---- path-metric-bounds
     |  |  |  +---- path-metric-bound* [metric-type]
     |  |  |     +---- metric-type    identityref
     |  |  |     +---- upper-bound?   uint64
     |  |  +---- path-affinities-values
     |  |  |  +---- path-affinities-value* [usage]
     |  |  |     +---- usage    identityref
     |  |  |     +---- value?   admin-groups
     |  |  +---- path-affinity-names
     |  |  |  +---- path-affinity-name* [usage]
     |  |  |     +---- usage              identityref
     |  |  |     +---- affinity-name* [name]
     |  |  |        +---- name    string
     |  |  +---- path-srlgs-values
```

```
            |  |  |  +---- usage?    identityref
            |  |  |  +---- values*   srlg
            |  |  +---- path-srlgs-names
            |  |  |  +---- path-srlgs-name* [usage]
            |  |  |     +---- usage        identityref
            |  |  |     +---- srlg-name* [name]
            |  |  |        +---- name    string
            |  |  +---- disjointness?          te-types:te-path-
    disjointness
         |  +---- 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)?
         |  |     |  |  |        +--:(num-unnum-hop)
         |  |     |  |  |        |  +---- num-unnum-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
         |  |     |  |  |        +--:(as-number)
         |  |     |  |  |        |  +---- as-number-hop
         |  |     |  |  |        |     +---- as-number?   binary
         |  |     |  |  |        |     +---- hop-type?    te-hop-type
         |  |     |  |  |        +--:(label)
         |  |     |  |  |        |  +---- label-hop
         |  |     |  |  |        |     +---- te-label
         |  |     |  |  |        |        +---- (technology)?
         |  |     |  |  |        |        |  +--:(generic)
         |  |     |  |  |        |        |     +---- generic?
```

```
      | |      | | |          |          |                rt-
   types:generalized-label
      | |      | | |          |          +---- direction?
      | |      | | |          |                  te-label-direction
      | |      | | |          +--:(srlg)
      | |      | | |             +---- srlg
      | |      | | |                +---- srlg?   uint32
      | |      | | +---- explicit-route-include-objects
      | |      | |    +---- route-object-include-object* [index]
      | |      | |       +---- index          uint32
      | |      | |       +---- (type)?
      | |      | |           +--:(num-unnum-hop)
      | |      | |           | +---- num-unnum-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
      | |      | |           +--:(as-number)
      | |      | |           | +---- as-number-hop
      | |      | |           |    +---- as-number?   binary
      | |      | |           |    +---- hop-type?    te-hop-type
      | |      | |           +--:(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
        |  +---- return-srlgs?             boolean
        |  +---- return-affinities?        boolean
        |  +---- path-in-segment!
        |  |  +---- label-restrictions
        |  |     +---- label-restriction* [index]
        |  |        +---- restriction?    enumeration
        |  |        +---- index           uint32
        |  |        +---- label-start
        |  |        |  +---- te-label
        |  |        |     +---- (technology)?
        |  |        |     |  +--:(generic)
        |  |        |     |     +---- generic?     rt-types:generalized-
  label
        |  |        |     +---- direction?   te-label-direction
        |  |        +---- label-end
        |  |        |  +---- te-label
        |  |        |     +---- (technology)?
        |  |        |     |  +--:(generic)
        |  |        |     |     +---- generic?     rt-types:generalized-
  label
        |  |        |     +---- direction?   te-label-direction
        |  |        +---- label-step
        |  |        |  +---- (technology)?
        |  |        |     +--:(generic)
        |  |        |        +---- generic?   int32
        |  |        +---- range-bitmap?   binary
        |  +---- path-out-segment!
        |     +---- label-restrictions
        |        +---- label-restriction* [index]
        |           +---- restriction?    enumeration
        |           +---- index           uint32
        |           +---- label-start
        |           |  +---- te-label
        |           |     +---- (technology)?
        |           |     |  +--:(generic)
```

```
         |              |      |        +---- generic?      rt-types:generalized-
    label
         |              |      +---- direction?   te-label-direction
         |              +---- label-end
         |              | +---- te-label
         |              |    +---- (technology)?
         |              |    |  +--:(generic)
         |              |    |     +---- generic?      rt-types:generalized-
    label
         |              |      +---- direction?   te-label-direction
         |              +---- label-step
         |              | +---- (technology)?
         |              |    +--:(generic)
         |              |       +---- generic?   int32
         |              +---- range-bitmap?   binary
         +---- synchronization* [synchronization-id]
            +---- synchronization-id    uint32
            +---- svec
            | +---- relaxable?          boolean
            | +---- disjointness?       te-types:te-path-disjointness
            | +---- request-id-number*   uint32
            +---- svec-constraints
            | +---- path-metric-bound* [metric-type]
            |    +---- metric-type    identityref
            |    +---- upper-bound?   uint64
            +---- path-srlgs-values
            | +---- usage?   identityref
            | +---- values*   srlg
            +---- path-srlgs-names
            | +---- path-srlgs-name* [usage]
            |    +---- usage         identityref
            |    +---- srlg-name* [name]
            |       +---- name    string
            +---- exclude-objects
            | +---- excludes* [index]
            |    +---- index             uint32
            |    +---- (type)?
            |       +--:(num-unnum-hop)
            |       | +---- num-unnum-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
          |       +--:(as-number)
          |       | +---- as-number-hop
          |       |    +---- as-number?   binary
          |       |    +---- hop-type?     te-hop-type
          |       +--:(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-values
```

```
                    |  +--ro path-affinities-value* [usage]
                    |     +--ro usage    identityref
                    |     +--ro value?   admin-groups
                    +--ro path-affinity-names
                    |  +--ro path-affinity-name* [usage]
                    |     +--ro usage           identityref
                    |     +--ro affinity-name* [name]
                    |        +--ro name    string
                    +--ro path-srlgs-values
                    |  +--ro usage?    identityref
                    |  +--ro values*   srlg
                    +--ro path-srlgs-names
                    |  +--ro path-srlgs-name* [usage]
                    |     +--ro usage        identityref
                    |     +--ro srlg-name* [name]
                    |        +--ro name    string
                    +--ro path-route-objects
                       +--ro path-route-object* [index]
                          +--ro index            uint32
                          +--ro (type)?
                             +--:(num-unnum-hop)
                             |  +--ro num-unnum-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
                             +--:(as-number)
                             |  +--ro as-number-hop
                             |     +--ro as-number?   binary
                             |     +--ro hop-type?    te-hop-type
                             +--:(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@2018-10-23.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 "2018-10-23" {
       description
         "Initial revision";
       reference
         "draft-ietf-teas-yang-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 requested-info {
      description
        "This grouping defines the information (e.g., metrics)
         which must be returned in the response";
      list requested-metrics {
        key 'metric-type';
        description
          "The list of the requested metrics
           The metrics listed here must be returned in the response.
           Returning other metrics in the response is optional.";
        leaf metric-type {
          type identityref {
            base te-types:path-metric-type;
          }
          description
            "The metric that must be returned in the response";
        }
      }
      leaf return-srlgs {
        type boolean;
        default false;
        description
          "If true, path srlgs must be returned in the response.
           If false, returning path srlgs in the response optional.";
      }
      leaf return-affinities {
        type boolean;
        default false;
        description
        "If true, path affinities must be returned in the response.
         If false, returning path affinities in the response is
         optional.";
      }
    }
```

```
   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";
          leaf index {
            type uint32;
            description "XRO subobject index";
          }
          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.";
          }
          uses te-types:generic-path-disjointness;
          leaf-list request-id-number {
            type uint32;
```

```
          description
            "This list reports the set of 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.";
    }
  }

  /*
   * These groupings should be removed when defined in te-types
   */

  grouping encoding-and-switching-type {
    description
      "Common grouping to define the LSP encoding and
       switching types";

    leaf encoding {
      type identityref {
        base te-types:lsp-encoding-types;
      }
      description "LSP encoding type";
      reference "RFC3945";
    }
    leaf switching-type {
      type identityref {
        base te-types:switching-capabilities;
```

```
        }
        description "LSP switching type";
        reference "RFC3945";
      }
    }

    grouping end-points {
      description
        "Common grouping to define the TE tunnel 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.";
      }
      leaf bidirectional {
        type boolean;
        default 'false';
        description "TE tunnel bidirectional";
      }
    }

    /**
     * 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.";
          }
          uses te-types:te-topology-identifier;
          uses end-points;
          uses encoding-and-switching-type;
          uses te-types:path-route-objects;
          uses te-types:generic-path-constraints;
          uses te-types:generic-path-optimization;
          uses requested-info;
          uses te:path-access-segment-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 [RFC8453]. 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 [RFC7950], [RFC7491] and [RFC8453].

   The YANG module defined in this draft is designed to be accessed via
   the NETCONF protocol [RFC6241] or RESTCONF protocol [RFC8040]. The
   lowest NETCONF layer is the secure transport layer, and the
   mandatory-to-implement secure transport is Secure Shell (SSH)
   [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-
   implement secure transport is TLS [RFC8446].

   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 [RFC7950] apply for this
   document as well.

   The NETCONF access control model [RFC8341] provides the means to
   restrict access for particular NETCONF or RESTCONF users to a
   preconfigured subset of all available NETCONF or RESTCONF protocol
   operations and content.

Note - The security analysis of each leaf is for further study.

8. IANA Considerations

This document registers the following URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made.

URI: urn:ietf:params:xml:ns:yang:ietf-te-path-computation
XML: N/A, the requested URI is an XML namespace.

This document registers a YANG module in the YANG Module Names registry [RFC7950].

name: ietf-te-path-computation
namespace: urn:ietf:params:xml:ns:yang:ietf-te-path-computation
prefix: tepc

9. References

9.1. Normative References

[RFC3688] Mealling, M., "The IETF XML Registry", RFC 3688, January
          2004.

[RFC5440] Vasseur, JP., Le Roux, JL. et al., "Path Computation
          Element (PCE) Communication Protocol (PCEP)", RFC 5440,
          March 2009.

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

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed.,
          and A. Bierman, Ed., "Network Configuration Protocol
          (NETCONF)", RFC 6241, June 2011.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure
          Shell (SSH)", RFC 6242, June 2011.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF
          Protocol", RFC 8040, January 2017.

   [RFC8341] Bierman, A., and M. Bjorklund, "Network Configuration
             Access Control Model", RFC 8341, March 2018.

   [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.

   [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF
             Protocol", RFC 8040, January 2017.

   [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol
             Version 1.3", RFC 8446, August 2018.

   [RFC8453] Ceccarelli, D., Lee, Y. et al., "Framework for Abstraction
             and Control of TE Networks (ACTN)", RFC8453, August 2018.

   [RFC8454] Lee, Y. et al., "Information Model for Abstraction and
             Control of TE Networks (ACTN)", RFC8454, September 2018.

   [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.

9.1. Informative References

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

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

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

   [RFC8233] Dhody, D. et al., "Extensions to the Path Computation
            Element Communication Protocol (PCEP) to Compute Service-
            Aware Label Switched Paths (LSPs)", RFC 8233, September
            2017

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

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

10. Acknowledgments

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 [RFC8233], 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
   detailed connectivity matrix (whose size we are estimating), no
   other topological details are available, therefore the size of
   the detailed 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 the detailed 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
   detailed connectivity matrixes, whose size need to be estimated
   here, are the ones relevant to the abstract TE nodes exported to
   the Orchestrator. These detailed 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 detailed 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 the information provided by the
detailed 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@gmail.com

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