PROJECT

SDN Controller (Open Daylight) based implementation of PCEP (Path Computation Element Protocol) for network path instantiation.

Under the guidance of : **Prof.GURPREET NANDA**
Submitted by: Jashandeep Kaur

ABSTRACT

The Path Computation Element (PCE) has become established as a core element of Software Defined Networking (SDN) systems. It will compute optimal paths for traffic inside a network for any definition of "optimal" and may conjointly monitor changes in resource accessibility and traffic demands to manage and update the paths.

Traditionally, the PCE has been implemented to derive paths for MPLS Label Switched Paths (LSPs). These paths are provided using the Path Computation Element Communication Protocol (PCEP) to the head end of the LSP for signaling in the MPLS network.

SDN has a far wider capability than just simply signaled MPLS traffic engineered networks, and the PCE could be utilized to discover paths in a widespread series of use cases including static LSPs, service function chaining (SFC), segment routing, and indeed any arrangement of routed or switched network. It is, therefore logical to consider PCEP as a general southbound control protocol to be used in these environments to permit the PCE to be absolutely enabled as a central controller.

This report primarily emphasis on Path Computation Element Protocol and based architecture which is broadly deployed in SDN based MPLS networks and numerous different as SDN WAN since its growing its original stateless condition to other capabilities, including the stateful, active and instantiation functionalities. This drives the implementation of novel solutions enabling effective software defined networking (SDN).

TABLE OF CONTENTS

ACKNOWLEDGEMENT

First of all, I am grateful to **The Almighty God** for the accomplishment of this project.

I am highly indebted to Mr. Gurpreet Nanda for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

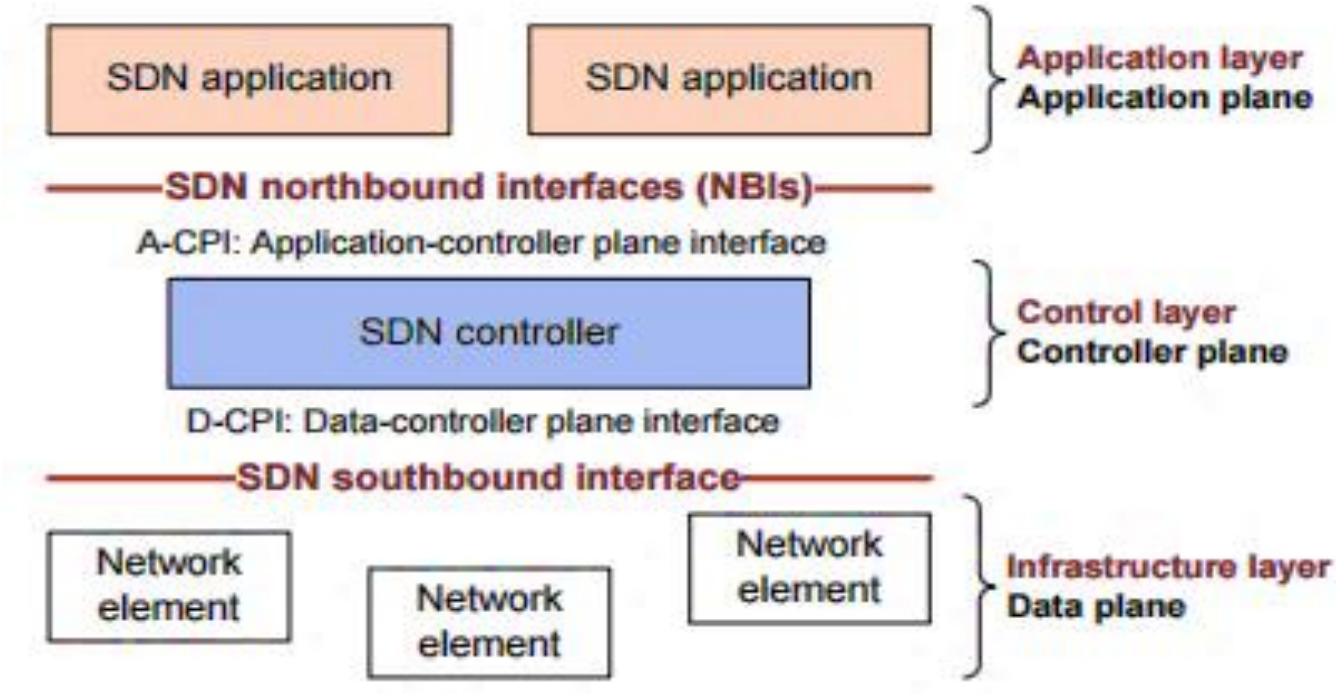
I would like to express my gratitude towards my parents & member of University of Alberta for their kind co-operation and encouragement which help me in completion of this project.

My thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities.

CHAPTER 1
BASIC SDN INTRODUCTION

1.SDN AND NFV INTRODUCTION

SDN is a new kind of software based open and programmable network model that separates network control plane from forwarding data plane. SDN provides software based network services that make it capable to deploy and manage networks in better way and adapt to rapidly changing cloud computing services. SDN networks are easier to deploy and adapt than traditional networks. SDN reduces network complexity by masking bottom-layer complexity and providing efficient configuration and management for upper layers. SDN projects a new system for networking that better supports new network architectures and new service innovations.



Network Functions Virtualisation virtualized the traditional network function such as a Session Border Gateway, a firewall, a Broadband Network Gateway (BNG) into virtually running machines which are based on Commercial off the Shelf(COTS) hardware. This cut off the operating cost for vendor specific services

and service cards within routers or switches and allowing operator to use their x86 server infrastructure to run network operations.

It is possible to virtualise routers and switches and CPE, the limitations of what is possible relate to packet throughput and latency requirements. Low latency Input/ Output (I/O) centric applications typically run best on network hardware. Less delay critical, compute intensive applications can be more cost effective to run on x86 hardware, rather than dedicated service cards on routers.

1.1 Challenges and Problems Facing SDN Development

SDN has been widely recognized revolutionary standard by industry leaders as a technology to shape the future of carriers' networks. Though, numerous challenges and complications still be present when implementing and developing SDN networks. SDN faces the following challenges during deployment:

- Difficult to measure return on investment (ROI)
  It is difficult to measure the impact of introducing SDN on carriers' overall network investment, but this is sure that it will be significant. SDN apply standardized hardware and wide-range of software components to carry out network functions which greatly reduces hardware costs, but increasing software costs substantially. Presently, there is not adequate comprehensive functioning data to evaluate the deployment costs of traditional networks and SDN networks. Introducing a new technology or deploying a new software defined network will continue simultaneously with existing services provided by carriers. For a short duration, the CAPEX will increase, but as the OPEX of new SDN networks reduces and SDN is widely applied, the overall CTO of networks will gradually decrease.

- Unformed industry chain
  Core SDN capabilities are open and programmable. The SDN industry chain includes various types of organizations, including standardization and open source organizations, application developers, carriers, telecom vendors and

network users. These organizations have yet up to consensus on how to deploy SDN networks for the utmost profit of all parties. As the carriers' existing networks also comprises of network solutions delivered by different telecom vendors. In order to ensure the effective commercial deployment of SDN, these organisations must collectively determine the way to form an effective and workable industry chain based on a open SDN platform.

- Open potentials in the infant stage
  SDN based technology will bring new profits through simplifying O&M (operation and maintenance) and networks, and reducing costs related to O&M and network construction. The carriers' SDN networks must have open capabilities to ripe these benefits, which are still in infant stage as SDN architecture-based application innovation has just begun. Mainly network carriers are concentrating only on reducing network construction and O&M costs through SDN. Therefore, SDN has not still to its peak value of deployment and development in terms of businesses and related various network industry deployment standards.

- Lack of structural unity and skilled personnel
  The network companies usually set up separate departments for different networking technologies. For example, IP department for data networks, the transmission department is responsible for transport networks, and wireless department for wireless networks. In SDN End-to-End architecture, different networks might have overlying services. For instance, in an IP and optical scenario, the transport network and IP network must synchronize to uniformly compute or reinstate paths. Carriers must proficiently integrate various departments to carry out network planning, developing, and maintenance. Such collaborative structural design poses a higher constraint on organizational structure and personnel skills.

SDN confronts the following problems as it continues to progress:

- Open interface standardization and interoperability

Southbound interface protocols carry on to evolve and vary in presentation. Northbound interface standardization is still very modern and has yet to be acknowledged in the industry. Startup research on eastbound and westbound interfaces for large-scale networking is yet doubtful in the industry. On top, yet there is no detailed and specific standards interpreting SDN orchestrator functions, service data models, and management functions of different layers.

- Scalability, security and stability
  The SDN networks primarily use software based SDN controllers to centrally design routes and use a centralized control mode, dissimilar to traditional networks that use distributed control planes. This approach is more appropriate for small networks. If we stretch this approach towards large scale network implementation, it presents great challenges to the reliability and extensibility of the centralized SDN architecture as multiple controllers are required for network development. The communications on an SDN network has a whole new set of potential risks as SDN controllers are inherently open. In order, to account for these risks, and quarantined protection mechanism must be advanced to ensure smooth progressing of SDN.

- Performance and reliability
  SDN controller software architecture and performance however demand further optimization. On large scale SDN networks, the controller reliability and performance creates high service flow delivery requirements whereas the standard chip specification needed by generalized hardware for the forwarding layer has not yet been released.

- Compatibility issues with prevailing networking O&M systems
  The procedure of merging SDN networks with existing infrastructures is a challenging and demanding job to ensure cooperation and proper connections between previously existing non-SDN devices/software and newly deployed SDN devices software.

1.2 SDN Significance of WANs

SDN implements software-defined functions through open and programmable interfaces which separates the control plane from the forwarding plane and replaces the original distributed control with the centralized control. The standard SDN network architecture consist of a control layer, a forwarding layer (infrastructure layer), and an application layer. As contrasted with traditional network architecture, SDN architecture only need data forwarding through bottom data layer present in universal commercial devices. The upper layer is an independent software system responsible for centralized control which defines the type and function of network devices, deploys and manages network devices through automatic remote configuration, and delivers mandatory network functions, parameters, and services. The application of SDN on WANs will revolutionize the traditional telecom network architecture.

The new SDN architecture helps WANs in the following approaches:

- Simplifies networks
  SDN network architecture removes most control protocols and separates control plane from forwarding plane, which simplifies and unifies the forwarding plane. Hardware becomes more universal, and southbound interfaces are standardized to let devices of different vendors to interconnect, which diminishes device complexity and hardware costs.

- Automates the deployment of service
  In an SDN network, the controller controls the whole network. The various network services, such as masks internal network details, L2VPN and L3VPN, and allows End-to-End service automation are provided and deployed by SDN controller.

- Reduces CAPEX
  The standards for southbound interfaces among the SDN controller and forwarders if mature, network devices function as white box devices, dropping carriers' purchasing costs on forwarders thus lowering overall operating costs.

- Accelerates network innovation

  The programmable and open nature of SDN network architecture make vendors capable to rapidly accelerate service innovation and roll out new services.

  − SDN offers open northbound network interfaces to permit upper-layer applications to explore required network resources and services in a flexible and differentiated way, hence accelerating network innovation.

  − SDN programmability allows the control plane to deliver policies to network devices, improving network agility.

  − Programmability and openness of SDN make capable to service application in weeks as compared with years on traditional networks.

- Escalation in network utilization

  SDN network architecture offers centralized control to manage several network devices. Network O&M personnel plan networks, adjust paths, optimize network resources, and improve network utilization based on a global network view and network traffic status.

1.2 SDN Layer Architecture

Mainly, SDN is based on a broad concept of separation between a controller entity and a controlled entity. The controller manipulates the controlled entity via an interface which are mainly API requests through some library or system call. Though, such interfaces may be amplified via some protocol definition, and use local inter-process communication (IPC) or a protocol which may also act remotely; the protocol may be defined as an open standard or in a proprietary manner.

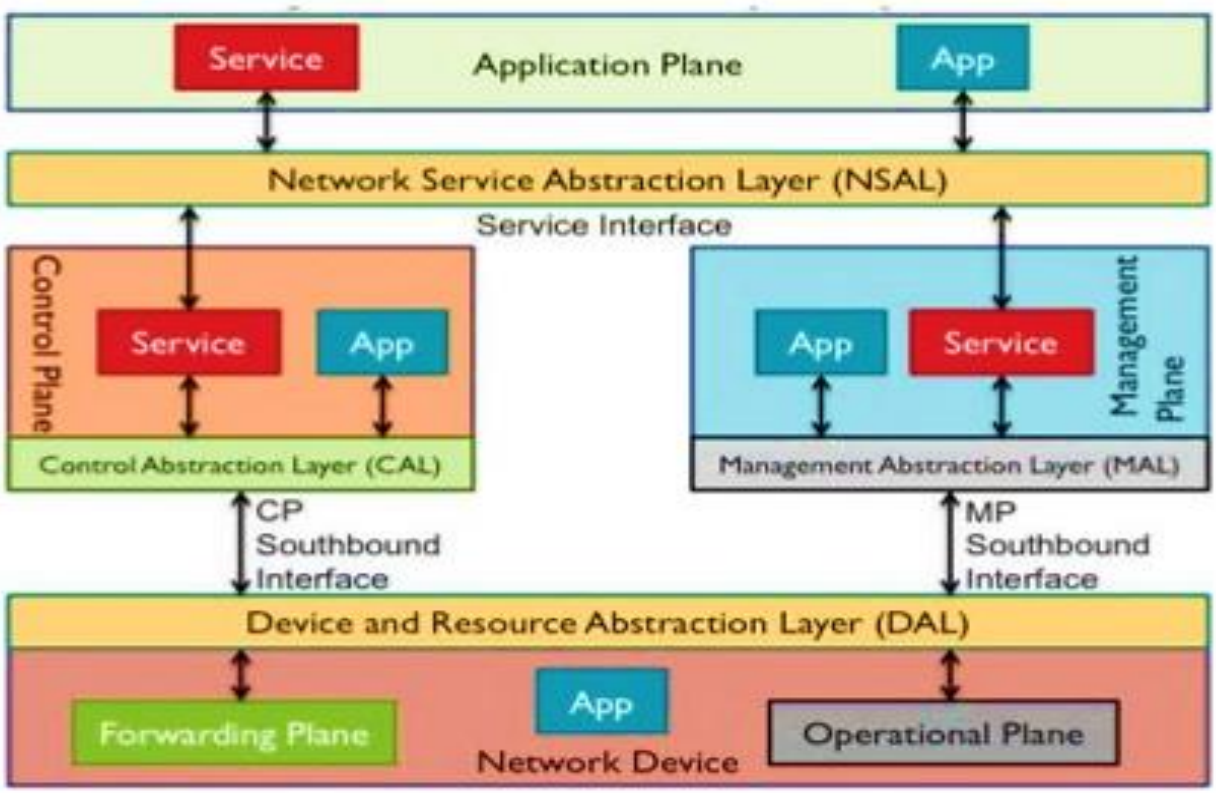


Fig 1.2: SDN Layer Architecture

- Forwarding Plane – this plane is responsible for managing packets in the data path based on the instructions received from the control plane. There are various actions performed by forwarding plane which mainly include forwarding, changing and dropping packets. The forwarding plane is basically the termination and operational point for control-plane services and applications. The forwarding plane is extensively described to as the "data path" or the “data plane” as it contains forwarding resources such as classifiers.

- Operational Plane –
  accountable for dealing with the operational state of the network device, for example whether or not the device is in operational state or idle, the number of ports available, the status of each port, and so on. The operational plane is generally the termination end for management-plane applications and services. The operational plane interrelates to network device resources including ports, memory, and so forth. We be aware that a few contributors

of the IRTF SDNRG have a distinct opinion regarding the definition of the operational plane. That is, one will argue that the operational aircraft does no longer represent a "plane", but it's an amalgamation of capabilities at the forwarding plane.

- Control Plane –
  responsible for making decisions and creating choices on how packets have to be forwarded through one or additional network devices and pushing such selections all the way down to the network devices for execution. The control plane typically focuses totally on the forwarding plane and fewer on the operational plane of the device. The control plane may be inquisitive about operational-plane info, that could consist of, as an instance, the present state of a specific port or its abilities. The control plane's principal process is to manipulate the forwarding tables that are residing within the forwarding plane, primarily based on the external service requests or the network topology.

- Management Plane –
  liable for configuring, monitoring and preserving network devices, for example, making selections relating to state of a network device. The management plane sometimes focuses totally on the operational plane of the network device and fewer at the forwarding plane. The management plane can be used to configure the forwarding plane, however it does so occasionally and through a greater comprehensive method than the control plane. As an instance, the management plane may additionally set up all or part of the forwarding regulations immediately, even though such motion might be anticipated to be taken sparingly.

- Application Plane –
  The plane where services and applications that describe network conduct exist. Programs that without delay (or normally) support the operation of the forwarding plane (along with routing processes in the control plane) aren't considered a part of the application plane. Consider that applications may be applied in a dispensed and modular fashion and, consequently, will usually span multiple planes

Additionally, have four abstraction layers:

- The Network Services Abstraction Layer (NSAL) provides service abstractions for use by applications and other services.
- The Management Abstraction Layer (MAL) abstracts the Management-Plane Southbound Interface and the DAL from the applications and services of the management plane.
- The Control Abstraction Layer (CAL) abstracts the Control-Plane Southbound Interface and the DAL from the services and applications of the control plane.
- The Device and resource Abstraction Layer (DAL) abstracts the resources of the device's forwarding and operational planes to the control and management planes. Variations of DAL might summarise each planes or both of the two and may summary any plane of the device to either the control or management plane.

CHAPTER 2
OPENDAYLIGHT

2.1 Opendaylight Controller:

Hosted by the Linux operating system Foundation, OpenDaylight development (ODL) is an open source SDN project aimed toward improving software program-defined networking (SDN) by way of presenting a network-led and enterprise-supported framework for the OpenDaylight Controller, that has been renamed the OpenDaylight Platform. It's open to anyone, such as customers and end users, and it gives a shared platform for those with SDN target to work collectively to discover new solutions.

Under the Linux foundation, OpenDaylight collaborate for the OpenFlow protocol, however can also guide different open SDN standards.
The OpenFlow protocol, taken into consideration the first SDN standard, defines the open protocol that lets in the SDN Controller to operate with the forwarding plane and build modifications to the network. This offers agencies the potential to better adapt to their dynamically changing desires, and have larger management over their networks.

The OpenDaylight Controller is in a position to implement in a range of production network environments. It could assist a modular controller framework, but can offer help for different SDN standards and future protocols.

The OpenDaylight Controller exposes open northbound APIs, which are utilized by applications. Those applications use the Controller to gather info concerning the network, run algorithms to conduct analytics, after which use the OpenDaylight Controller to create new policies throughout the network.

The OpenDaylight Controller is implemented entirely in software, and is saved within its very own Java digital system (JVM). This suggests it will be deployed on hardware and software package platforms that support Java.

2.2 OpenDaylight PCEP plugin

The OpenDaylight PCEP plugin offers all simple service units essenial to build-up a PCE-based controller. Additionally, it offers LSP management practicality for Active Stateful PCE - the cornerstone for majority of PCE-enabled SDN solutions. It consists of the subsequent components:

•PCEP session handling
•Protocol library
•Stateful PCE LSP-DB
•Active Stateful PCE LSP Operations

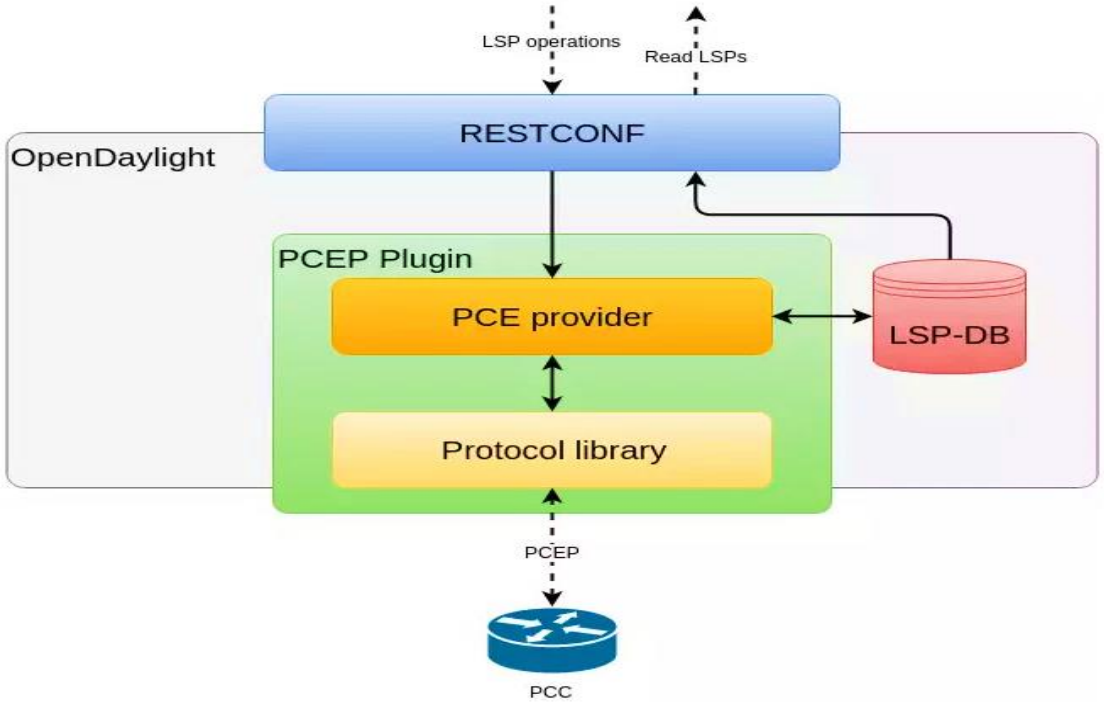


Fig 2.2 : Opendaylight OpenFlow PCEP Plugin

2.3 NETCONF-YANG

The Network Configuration Protocol (NETCONF) [RFC6241] is an IETF network management protocol. NETCONF provides mechanisms to put in, manage, and delete the configuration of network devices.

NETCONF protocol operations are accomplished as far flung procedure calls (RPCs). The NETCONF protocol makes use of XML-based data encoding for the configuration data in addition to the protocol messages. Current studies, such as [ESNet] and [PENet], have proven that NETCONF functions better than SNMP.

Moreover, the YANG data modeling language has been evolved for specifying NETCONF data models and protocol operations. YANG is a data modeling language used to model configuration and state data manipulated by the way of NETCONF protocol , NETCONF notifications and NETCONF remote procedure calls.

YANG models the hierarchical organization of data as a tree, wherein which each node has either a value or a set of child nodes. Moreover , YANG structures data models into modules and submodules, allowing augmentation and reusability. YANG models will describe constraints to be enforced on the data. Additionally, YANG has a set of base datatypes and permits custom-defined datatypes in addition.
YANG allows the definition of NETCONF RPCs, which permits the protocol to have an extensible quantity of commands. For RPC definitions, the operations names, input parameters, and output parameters are defined using YANG data definition statements.
In accordance, the YANG model is suitable for specifying DAL for the forwarding and operational planes. NETCONF is suitable for the MPSI. NETCONF is a management protocol , which became not (at first) designed for fast CP updates, and it may not be appropriate for addressing the necessities of CPSI.

CHAPTER 3
PCEP PROTOCOL

3.PCEP Introduction

Path Computation Element - "An entity (component, application, or network node) that is capable of computing a network path or route based on a network graph and applying computational constraints." [from RFC 4655]

A PCE is probably a network node, or reserved computational platform that is resource-aware and has the capacity to take into account more than one constraints for a variety of path computation issues and switching technologies. The PCE Communication Protocol (PCEP) is employed between a Path Computation Client (PCC) and a PCE, or among multiple PCEs.

The PCE architecture represents a vision of networks that splits path computation for services, the sign of end-to-end connections, and actual packet forwarding. The definition of online and offline path computation relies on the reachability of the PCE from network and Network Management System (NMS) nodes and the type of optimization request that may considerably impact the optimization response time from the PCE to the PCC.

The PCEP messaging mechanism facilitates the specification of objective functions (requested algorithm and optimization criteria),computation endpoints (source and destination node addresses), and the associated constraints such as encoding type, the switching capability, and traffic parameters (e.g., requested bandwidth).

The PCE is a control-plane service that provides services for control-plane applications. PCEP may be used as an east-west interface between PCEs that may act as domain control entities (services and applications). The PCE operating institution group is specifying extensions [PCE Active] that allow an active PCE to control, using PCEP, MPLS or GMPLS Label Switched Paths (LSPs), thus making it applicable for the CPSI for MPLS and GMPLS switches.

Path Computation is the method of calculating the route through a network that should be taken by associate MPLS or GMPLS traffic engineered tunnel of a defined size, delay and jitter so that it will meet the requirements of the bandwidth reservation that it is supporting. The path computation element is a computing function within the network that the MPLS Label Edge Router has elected to delegate this calculation to. The Path Computation Element Protocol is the protocol run between the MPLS Label Edge Router (LER), known as the Path Computation Client (PCC) and the PCE. This protocol supports the signalling of the path characteristics from the PCC to the PCE.

To calculate the path, the PCE utilises its information of the availability in the network based on its view of the Traffic Engineering Database (TED). The TED contains the set of all of the links inside the MPLS domain, their characteristics and their available bandwidth. This information is distributed by traffic engineering enhancements to the IGP running in a given domain.

One solution for a PCE to achieve access to the TED is truly to peer with the IGP; in this manner a PCE gains access to a TED that is effectively shared between all of the nodes (PCC and PCE) within a given area. It is possible to use another out of band mechanisms to obtain access to the TED, or probably to supplement the information in the TED, for example, information about non active links that are not part of the IGP topology. These mechanisms must, however, support continual updates and must be capable of scaling to support all of the nodes within the domain. It is possible that some of the work of I2RS may improve the topology information available to a PCE.

The PCE calculates a path within the domain that it is responsible for and returns the results of the path to the PCC within the PCEP protocol. This path computation is returned as an MPLS explicit route object which provides a set of IP addresses that the MPLS Label Switched Path for the reservation must pass through. It is possible for a PCE to return a less prescriptive path computation by returning one or more loose hops as part of a path calculation. A loose hop would be, for example, an abstract IP address (which refers to a set of nodes) or an AS number.

In this case further action will be required by another PCC to complete the end-to-end path calculation.

3.2 PCE and PCEP AS A SERVICE

PCEP need to be deployed as an active stateful PCE with a purpose to utilise the IETF PCE structure for SDN. The central differentiation between a passive and active stateful PCE is that the active stateful PCE will manage and control the setup and tear down of the LSP resources that it's accountable for. This can be achieved with the aid of the PCC delegate the set of LSPs to the PCE that it desires the PCE to regulate and control. As soon as a PCE has been delegated to regulate the LSP then it can direct the PCC to update and replace the LSP to reflect changing conditions in the network, including assigning it a new direction. Within the PCE architecture the PCC still remains in control of the LSP, and update requests that violate the local policy held at the PC might result in the PCE request being rejected.

The capability for the PCE to modify the LSPs that it is responsible for agrees it to pro-actively alter reservations in response either to transforming network conditions or because of additional reservations being requested in the network. Due to the fact, PCE has been precise to support both MPLS and GMPLS capabilities. This together can be utilized by applications wishing to optimise the mapping of MPLS bearers to the optical layer; an example of this is shown in "In Operation Network Planning" - IEEE Communications Magazine January 2014.The paper shows that a PCE based optimisation tool will be used to prevent spectrum fragmentation in optical networks that support variable sized frequency slots. This can be achieved by allowing a controller to modify the allocation of lightpaths within the optical spectrum to group smaller light paths and free up larger contiguous blocks of spectrum.

The stateful PCE architecture has also been extended, as described in "PCEP Extensions for PCE-initiated LSP Setup in a Stateful PCE Model" to permit a PCE to request that a PCC provoke an LSP. This lets application driven reservation of

resources in the network and turns the PCE into a component of a fully-fledged bandwidth management implementation.

This kind of stateful PCE will support the following use cases:

- Optimisation of network resources across packet and optical transport layers.
- Handling of on demand bandwidth requests from a bandwidth management function (either triggered from the OSS or from a web services interface, or from a future SDN application API).
- Re-optimisation, re-establishment and prioritisation of reservations in the event of network disruption.
- Maintenance of bandwidth reservations in the network.

It should be noted that there is a significant commonality between integrating the packet transport layer and the optical transport layer using OpenFlow (as being defined by the ONF) and offering the same functions using PCE. The fundamental difference between the two solutions is simply that PCE is optimised for MPLS transport and IP routers and OpenFlow is optimised for Ethernet switches and Ethernet transport. An individual vendor or carrier's preference for PCEP or OpenFlow for this application will depend on the approach they are taking to SDN. Carriers looking to re-use existing routing and transport architectures may consider PCEP, those looking for a white-box and NFV implementation of SDN will naturally view OpenFlow to be a better fit.

3.3 Bringing PCE to the SDN

The Software Defined Networking (SDN) movement provides wider benefits to build networks more customized, application centric, efficient, and programmable. There are a lot diverse approaches to build SDN. OpenFlow and VXLAN protocols are acquiring more traction in data center environments, while Segment Routing and PCEP/BGP-LS are carriers' SDN technology option for clear explanations. Like OpenFlow/VXLAN Segment Routing and PCEP/BGP-LS doesn't need forklifting their existing network, these protocols based applications can be

implemented by just doing Software upgrade. The migration path to PCE-based SDN is evolutionary, with lower OPEX and CAPEX as compared to alternate approaches.

SDN-based MPLS network is on ultimate efficiency when its running PCE and BGP-LS as it offers visibility, flexibility and control over the network.
This indicates not only basic things expected from a controller, i.e. its ability to manage (create, modify and delete) LSPs (Label Switched Paths) without touching a single line of config on a router, but also provide other benefits like:

- Bandwidth-on-demand and auto-bandwidth calendaring are services provided by SDN controller.
- Traffic optimization and distribution by efficiently using optical links.
- Service orchestration across multiple MPLS networks.
- Planning maintenance windows for routers with point and click.

And last but not the least, this is an enabler for integrating an MPLS core with an Optical core.
This hurts many service providers: the independent build out of routing and DWDM domain, costing both CAPEX and OPEX.
This happens because the MPLS domain does not talk to the optical domain at all. And this happens because the two teams managing the two networks talk neither.

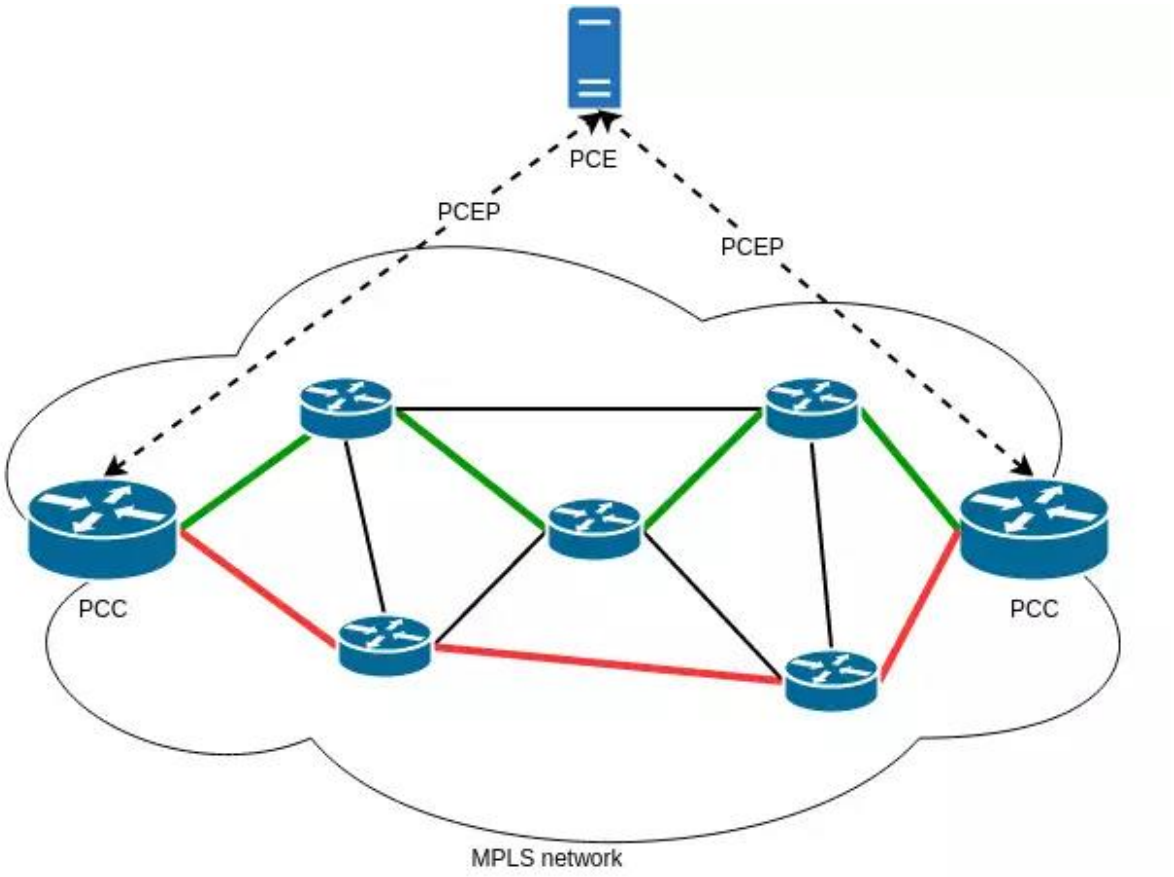


Fig3.3: PCE-based architecture

With the aid of extending the concept of PCE to SDN, an SDN controller, which firstly does not know about how the paths throughout domains, now gets the potential to compute end-to-end paths throughout multi-domain, multi-layer networks.

SDN architecture drives a software controller to manage and control the flow of packets from source to destination without the router having to make wise forwarding decisions. To determine how traffic will flow across more than one network nodes and domains, the SDN controller need to have the capability to compute multiple end-to-end paths for various traffic flows, some spanning different domains, while also keeping in account the constraints such as bandwidth, QoS, and latency requirements.

Network vendors, specially service providers, face a some challenges when implementing and deploying SDN in their IP/MPLS networks for traffic engineering purposes. First of all, for an SDN protocol such as OpenFlow to

communicate end-to-end path information to routers, the OpenFlow controller ought to understand the concept of MPLS forwarding and have all the logic or features of an MPLS router implemented in it. Every other important task is that SDN requires all the network nodes along the path to support the SDN protocol in use. This may require changing or upgrading all network nodes, potentially increasing expenditures, downtime, and alter management risks.

With a dedicated PCE server, the path is computed and sent to the head-end router to allow source routing-based forwarding of packets. Running PCE from a dedicated server additionally avoids over-loading the processors in head-end routers. In addition, PCE provides all existing MPLS-TE functionalities without the necessity for deploying protocols such as RSVP-TE and the associated overhead from running extra protocols in the network. Lastly, with PCE, solely the head-end router requests to be upgraded to better understand the path computation messages, thereby saving time and significant expenses for the network provider.

Adoption of PCE with SDN additionally paves the method for PCEP (Path Computation Communication Protocol) to grow to be an SDN protocol. As an SDN protocol, PCEP may be used by SDN controllers for path computation or for an SDN orchestrator to interact with other SDN controllers and provision distinct types of paths: a segment of an LSP, end-to-end LSPs (Label Switched Paths), or even to provide forwarding instructions to a single node.

In fact, extending the PCE components to feature as an SDN central controller lets an existing network to evolve without any issuses into an SDN enabled network with minimal adjustments to the cutting-edge infrastructure.

CHAPTER 4
RELATED WORK DONE

The PCE architecture, outlined in RFC 4655, simplifies path computation by separating network topology determination from path creation. Each tasks have conventionally been done by the provider edge router that receives the client path request. This router, referred to as the "head-end router," should support an Interior Gateway Protocol (IGP) such as OSPF to determine topology among its routing domain, and should additionally support the Border Gateway Protocol (BGP) if paths cross AS boundaries. Adding complex path computation will overwhelm the router CPU.
The PCE IETF Working Organisational Group was created in 2005, and RFC 4655 was revealed and published in 2006. The initial RFC outlined PCE architecture, and consequent RFCs have stuffed in details. Currently work is under way to add capabilities and address problems within the design.

Telecom service suppliers and providers have found PCE particularly striking since upgrading entire MPLS networks would be extremely pricey and disruptive. The flexibility of PCE to integrate optical network parameters in path computation is an supplementary improvement, as is the capability to make paths across routing domains and AS's.

CHAPTER 5
DESCRIPTION ABOUT YOUR WORK

In this project SDN controller (open daylight) enabled implementation of PCEP for network path instantiation is experimentally demonstrated.

An active stateful PCE with instantiation is a full controller (provisioning, modification and release of LSPs) and therefore, it could be integrated with the SDN/OpenFlow controller. Also, interfaces are internal and both components have access to a single instance of the TED and LSPDB. Moreover, this full integration is driven by the fact that having common, shared date structures and state simplifies concurrent access and updates. This can help enabling secure, dynamic, optimal, and inter-area as well as inter-domain traffic engineered path setup.

This approach doesn't require forklifting the existing network, the protocols based solution can be implemented by doing Software upgrade and also solve the multi-domain problem. PCE should be able to have topology visibility not only in single domain, but also build a topology database across multiple domains which helps provisioning LSP across multiple domains.

CHAPTER 6
DEPLOYMENT DETAILS (include hardware and software requirement)

5.1 Prerequisite:

Basic understanding of opendaylight and cisco network configuration and protocols.

For basic understanding started with opendaylight karaf supporting openflow protocol with mininet connectivity.We can explore various karaf features and play with them.

5.2 Hardware and Software

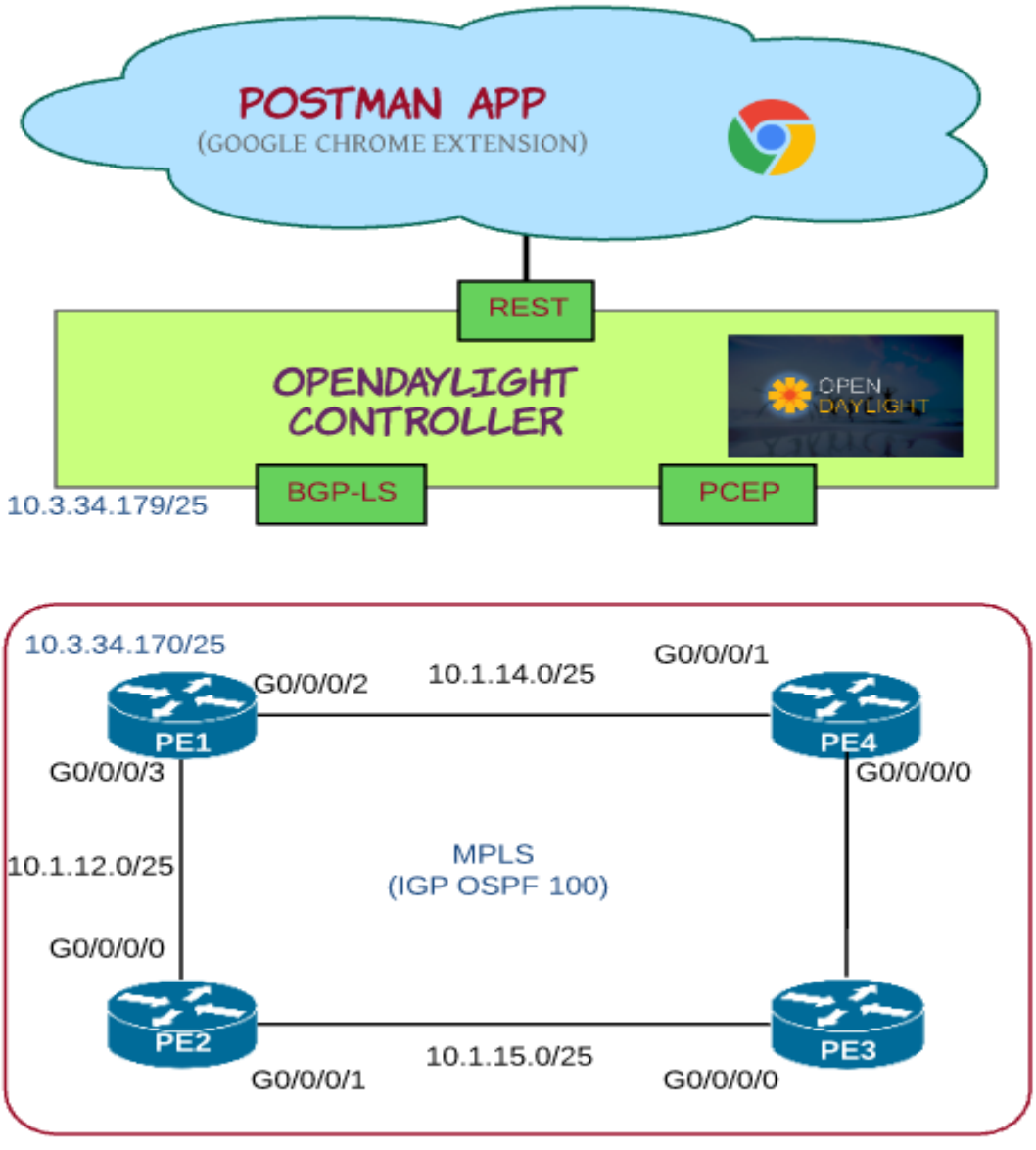
Dell servers(Mint lab), ESXi client , Ubuntu Virtual server, Cisco ios XRv, juniper vrouter and wireshark.

5.3 Deployment problems :

- During making internal connections on ESxi client for various routers and my controller faced some connectivity problems as the network construction is difficult task.
- Also faced some errors with clustering5.xml file on my controller due to standalone mode so let my controller work actually removed that file.
- Inspite of these specific problems faced some other common problems like understanding NETconf , hello packets checking , ports to call, registering devices , type of service registered ,node addition in controller and mounting cisco devices on controller whose solutions can be looked upon internet.

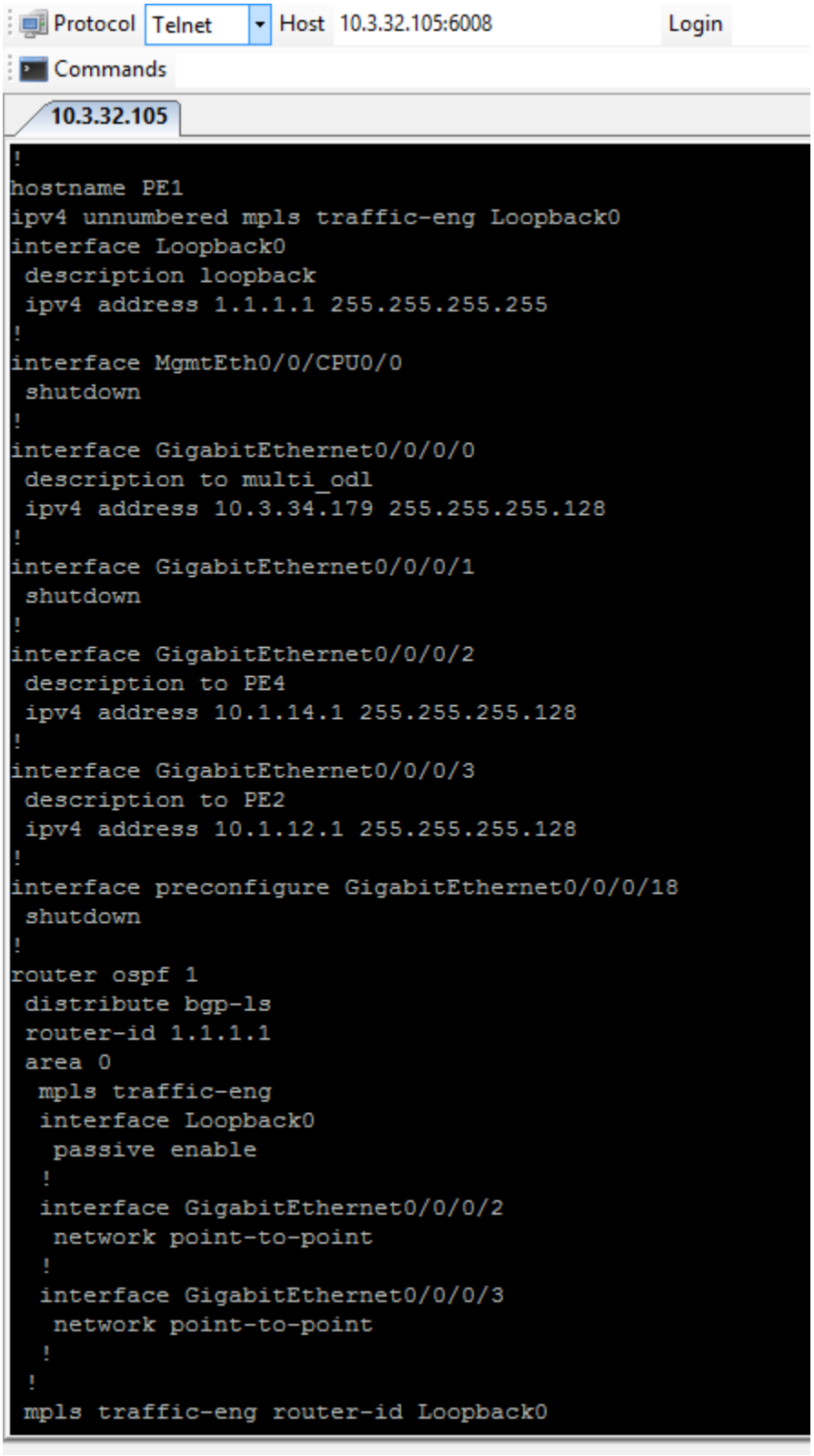
CHAPTER 7
TEST RESULT

STEP 1: Network topology (mpls and igp)



PE1 (IGP and MPLS)

Running-config



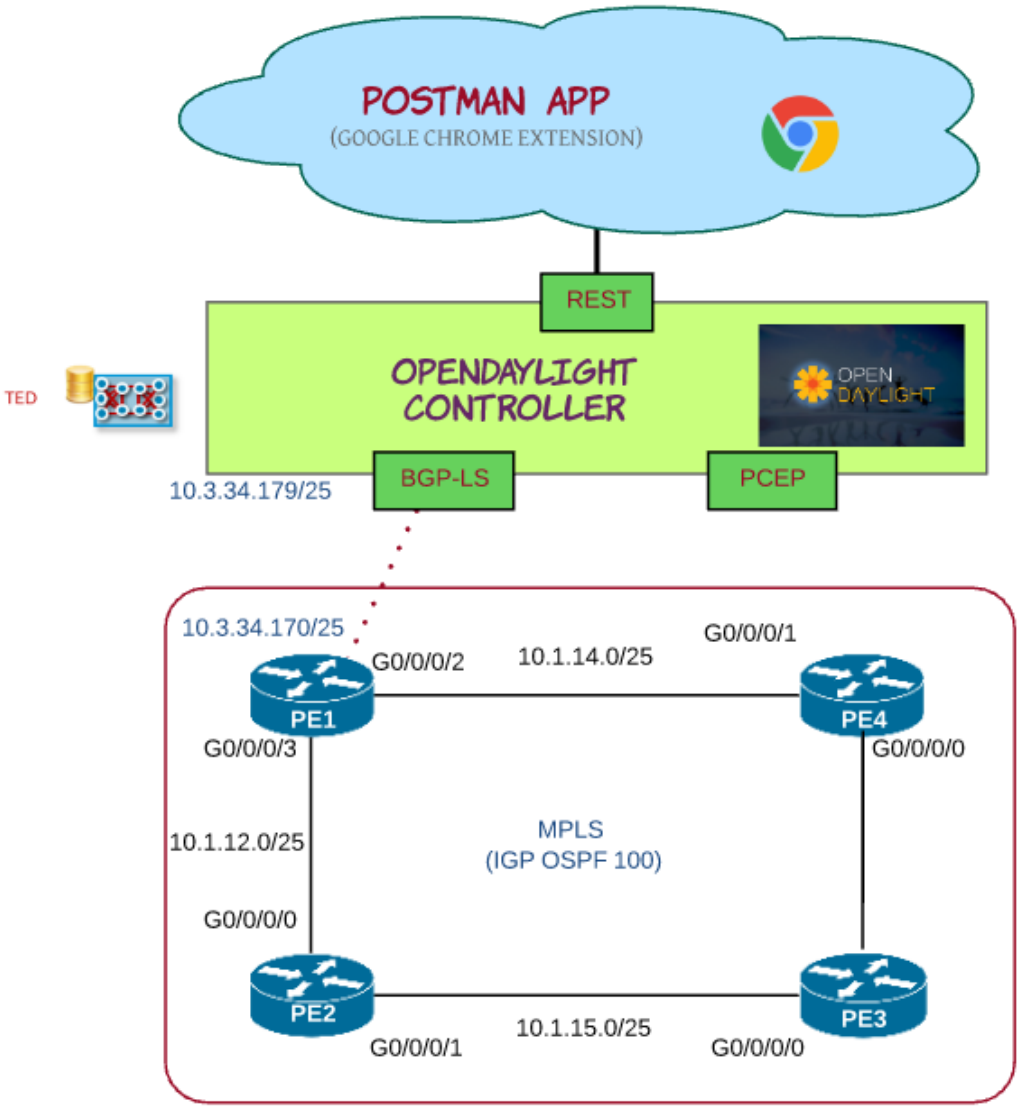
```
 mpls traffic-eng router-id Loopback0
!
router bgp 100
 bgp router-id 1.1.1.1
 address-family ipv4 unicast
  network 1.1.1.1/32
 !
 address-family link-state link-state
 !
 neighbor 10.3.34.170
  remote-as 100
  address-family link-state link-state
  !
 !
!
rsvp
 interface GigabitEthernet0/0/0/2
  bandwidth percentage 100
 !
 interface GigabitEthernet0/0/0/3
  bandwidth percentage 100
 !
!
mpls traffic-eng
 interface GigabitEthernet0/0/0/2
 !
 interface GigabitEthernet0/0/0/3
 !
```

Output of MPLS on PE1


10.3.32.105
RP/0/0/CPU0:PE1(config-if)#do show mpls traffic-eng topology ospf
Tue Mar 21 00:36:55.960 UTC
My_System_id: 1.1.1.1 (OSPF 1 area 0)
My_BC_Model_Type: RDM
Signalling error holddown: 10 sec Global Link Generation 1806
IGP Id: 1.1.1.1, MPLS TE Id: 1.1.1.1 Router Node (OSPF 1 area 0)
Link[0]:Point-to-Point, Nbr IGP Id:4.4.4.4, Nbr Node Id:-1, gen:1803
Frag Id:6, Intf Address:10.1.14.1, Intf Id:0
Nbr Intf Address:10.1.14.4, Nbr Intf Id:0
TE Metric:1, IGP Metric:1
Attribute Flags: 0x0
Ext Admin Group:
Length: 256 bits
Value : 0x::
Attribute Names:
Switching Capability:None, Encoding:unassigned
BC Model ID:RDM
Physical BW:1000000 (kbps), Max Reservable BW Global:1000000 (kbps)
Max Reservable BW Sub:0 (kbps)
Total Allocated BW (kbps)
Global Pool Reservable BW (kbps)
Sub Pool Reservable BW (kbps)
bw[0]: 0 1000000 0
bw[1]: 0 1000000 0
bw[2]: 0 1000000 0
bw[3]: 0 1000000 0
bw[4]: 0 1000000 0
bw[5]: 0 1000000 0
bw[6]: 0 1000000 0
bw[7]: 0 1000000 0
Link[1]:Point-to-Point, Nbr IGP Id:2.2.2.2, Nbr Node Id:2, gen:1804
Frag Id:7, Intf Address:10.1.12.1, Intf Id:0
Nbr Intf Address:10.1.12.2, Nbr Intf Id:0
TE Metric:1, IGP Metric:1
Attribute Flags: 0x0

Step 2: Network Topology(BGP-LS)

NOTE: Used for Traffic Engineering

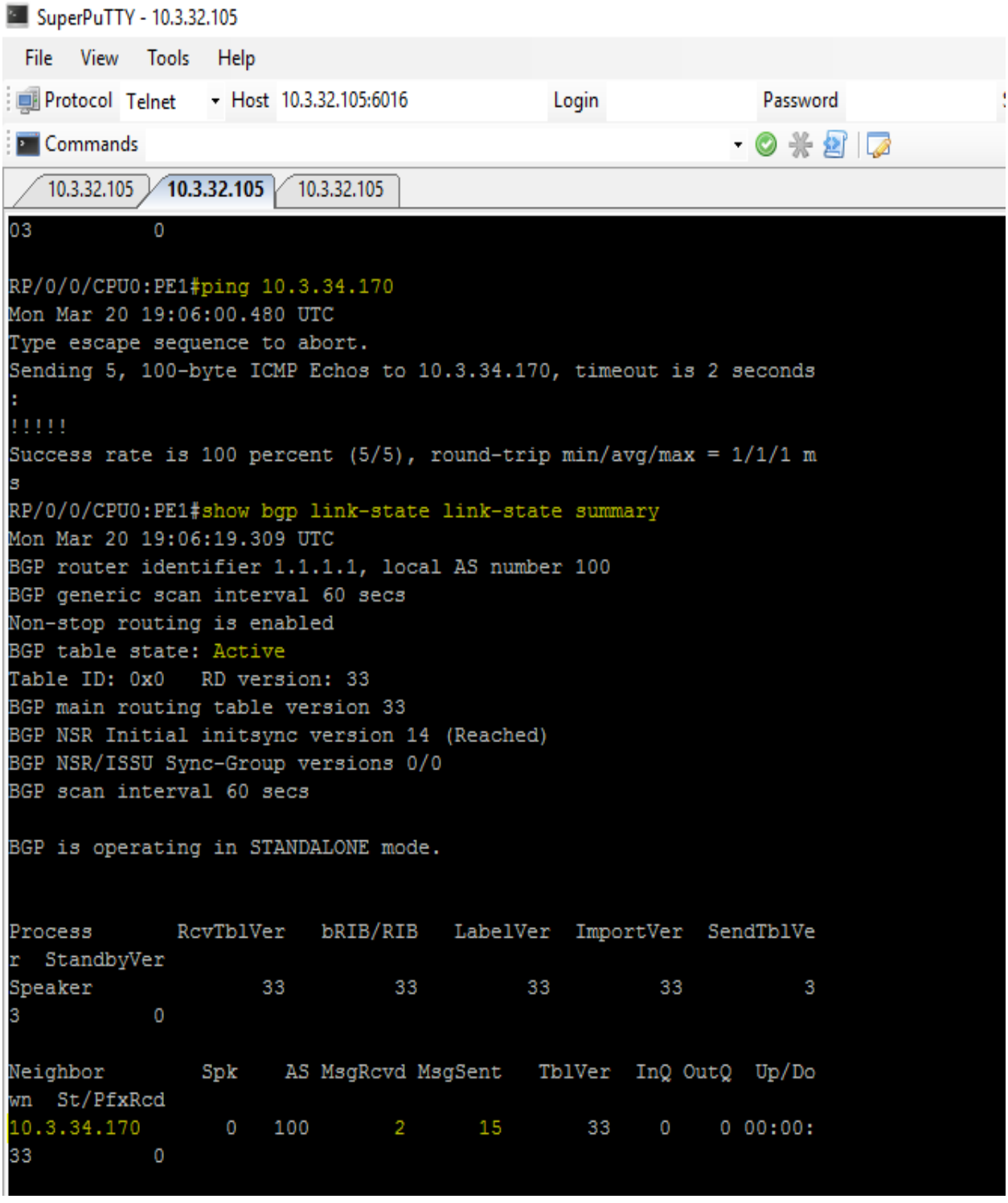


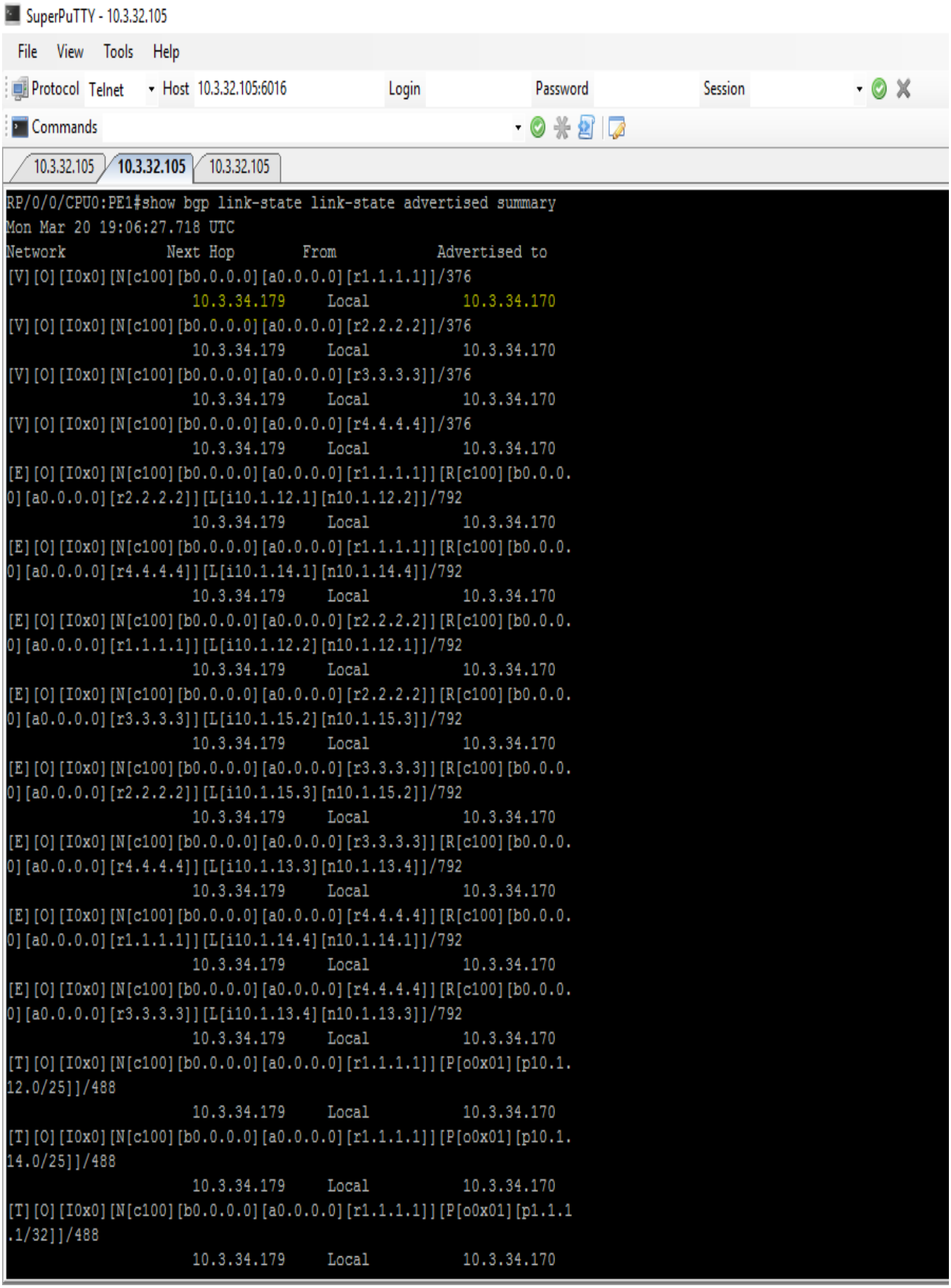
PE1 (BGP-LS)

```
RP/0/0/CPU0:PE1#show run router bgp
Tue Mar 21 00:54:15.349 UTC
router bgp 100
 bgp router-id 1.1.1.1
 address-family ipv4 unicast
  network 1.1.1.1/32
 !
 address-family link-state link-state
 !
 neighbor 10.3.34.170
  remote-as 100
  address-family link-state link-state
  !
 !
!

RP/0/0/CPU0:PE1#show run router ospf
Tue Mar 21 00:54:40.067 UTC
router ospf 1
 distribute bgp-ls
 router-id 1.1.1.1
 area 0
  mpls traffic-eng
  interface Loopback0
   passive enable
  !
  interface GigabitEthernet0/0/0/2
   network point-to-point
  !
  interface GigabitEthernet0/0/0/3
   network point-to-point
  !
 !
 mpls traffic-eng router-id Loopback0
!

RP/0/0/CPU0:PE1#
```

In OPENDAYLIGHT edit 41-bgp-example.xml in «./etc/opendaylight/karaf » directory. And edited BGP rib-id(10.3.34.170),AS(100) and BGP peer host ip(10.3.34.179).

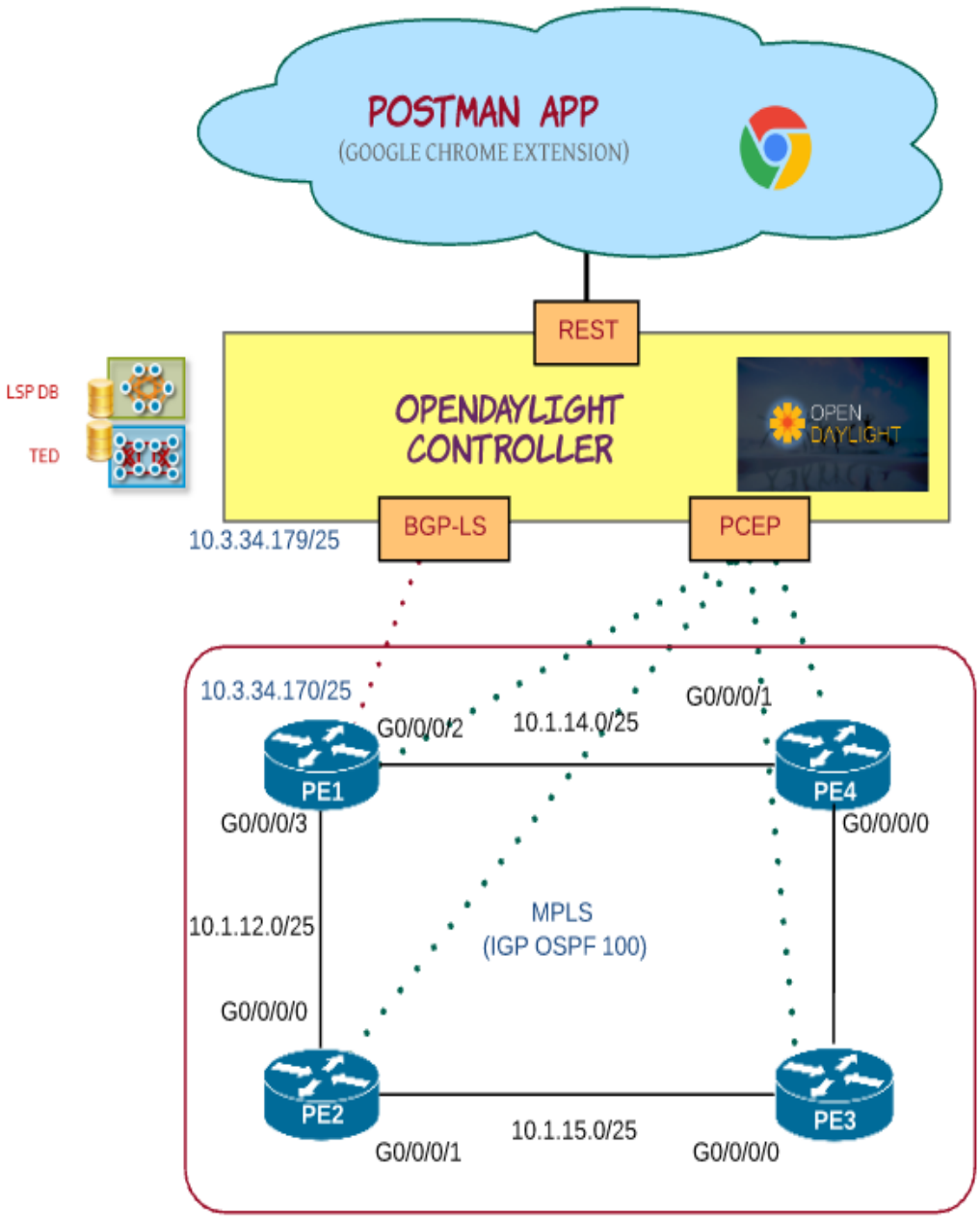
OUTPUT of BGP-LS neighbor at PE1


SuperPuTTY - 10.3.32.105
File View Tools Help
Protocol Telnet Host 10.3.32.105:6016 Login Password
Commands
10.3.32.105 10.3.32.105 10.3.32.105
03 0
RP/0/0/CPU0:PE1#ping 10.3.34.170
Mon Mar 20 19:06:00.480 UTC
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.3.34.170, timeout is 2 seconds
:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 m
s
RP/0/0/CPU0:PE1#show bgp link-state link-state summary
Mon Mar 20 19:06:19.309 UTC
BGP router identifier 1.1.1.1, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0 RD version: 33
BGP main routing table version 33
BGP NSR Initial initsync version 14 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs
BGP is operating in STANDALONE mode.
Process RcvTblVer bRIB/RIB LabelVer ImportVer SendTblVe
r StandbyVer
Speaker 33 33 33 33 3
3 0
Neighbor Spk AS MsgRcvd MsgSent TblVer InQ OutQ Up/Do
wn St/PfxRcd
10.3.34.170 0 100 2 15 33 0 0 00:00:
33 0


SuperPuTTY - 10.3.32.105
File View Tools Help
Protocol Telnet Host 10.3.32.105:6016 Login Password Session
Commands
10.3.32.105 10.3.32.105 10.3.32.105
RP/0/0/CPU0:PE1#show bgp link-state link-state advertised summary
Mon Mar 20 19:06:27.718 UTC
Network Next Hop From Advertised to
[V][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r1.1.1.1]]/376
10.3.34.179 Local 10.3.34.170
[V][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r2.2.2.2]]/376
10.3.34.179 Local 10.3.34.170
[V][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r3.3.3.3]]/376
10.3.34.179 Local 10.3.34.170
[V][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r4.4.4.4]]/376
10.3.34.179 Local 10.3.34.170
[E][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r1.1.1.1]][R[c100][b0.0.0.
0][a0.0.0.0][r2.2.2.2]][L[i10.1.12.1][n10.1.12.2]]/792
10.3.34.179 Local 10.3.34.170
[E][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r1.1.1.1]][R[c100][b0.0.0.
0][a0.0.0.0][r4.4.4.4]][L[i10.1.14.1][n10.1.14.4]]/792
10.3.34.179 Local 10.3.34.170
[E][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r2.2.2.2]][R[c100][b0.0.0.
0][a0.0.0.0][r1.1.1.1]][L[i10.1.12.2][n10.1.12.1]]/792
10.3.34.179 Local 10.3.34.170
[E][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r2.2.2.2]][R[c100][b0.0.0.
0][a0.0.0.0][r3.3.3.3]][L[i10.1.15.2][n10.1.15.3]]/792
10.3.34.179 Local 10.3.34.170
[E][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r3.3.3.3]][R[c100][b0.0.0.
0][a0.0.0.0][r2.2.2.2]][L[i10.1.15.3][n10.1.15.2]]/792
10.3.34.179 Local 10.3.34.170
[E][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r3.3.3.3]][R[c100][b0.0.0.
0][a0.0.0.0][r4.4.4.4]][L[i10.1.13.3][n10.1.13.4]]/792
10.3.34.179 Local 10.3.34.170
[E][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r4.4.4.4]][R[c100][b0.0.0.
0][a0.0.0.0][r1.1.1.1]][L[i10.1.14.4][n10.1.14.1]]/792
10.3.34.179 Local 10.3.34.170
[E][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r4.4.4.4]][R[c100][b0.0.0.
0][a0.0.0.0][r3.3.3.3]][L[i10.1.13.4][n10.1.13.3]]/792
10.3.34.179 Local 10.3.34.170
[T][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r1.1.1.1]][P[o0x01][p10.1.
12.0/25]]/488
10.3.34.179 Local 10.3.34.170
[T][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r1.1.1.1]][P[o0x01][p10.1.
14.0/25]]/488
10.3.34.179 Local 10.3.34.170
[T][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r1.1.1.1]][P[o0x01][p1.1.1
.1/32]]/488
10.3.34.179 Local 10.3.34.170

```
[E][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r3.3.3.3]][R[c100][b0.0.0.
0][a0.0.0.0][r4.4.4.4]][L[i10.1.13.3][n10.1.13.4]]/792
                     10.3.34.179     Local           10.3.34.170
[E][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r4.4.4.4]][R[c100][b0.0.0.
0][a0.0.0.0][r1.1.1.1]][L[i10.1.14.4][n10.1.14.1]]/792
                     10.3.34.179     Local           10.3.34.170
[E][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r4.4.4.4]][R[c100][b0.0.0.
0][a0.0.0.0][r3.3.3.3]][L[i10.1.13.4][n10.1.13.3]]/792
                     10.3.34.179     Local           10.3.34.170
[T][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r1.1.1.1]][P[o0x01][p10.1.
12.0/25]]/488
                     10.3.34.179     Local           10.3.34.170
[T][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r1.1.1.1]][P[o0x01][p10.1.
14.0/25]]/488
                     10.3.34.179     Local           10.3.34.170
[T][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r1.1.1.1]][P[o0x01][p1.1.1
.1/32]]/488
                     10.3.34.179     Local           10.3.34.170
[T][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r2.2.2.2]][P[o0x01][p10.1.
12.0/25]]/488
                     10.3.34.179     Local           10.3.34.170
[T][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r2.2.2.2]][P[o0x01][p10.1.
15.0/25]]/488
                     10.3.34.179     Local           10.3.34.170
[T][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r2.2.2.2]][P[o0x01][p2.2.2
.2/32]]/488
                     10.3.34.179     Local           10.3.34.170
[T][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r3.3.3.3]][P[o0x01][p10.1.
13.0/25]]/488
                     10.3.34.179     Local           10.3.34.170
[T][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r3.3.3.3]][P[o0x01][p10.1.
15.0/25]]/488
                     10.3.34.179     Local           10.3.34.170
[T][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r4.4.4.4]][P[o0x01][p10.1.
13.0/25]]/488
                     10.3.34.179     Local           10.3.34.170
[T][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r4.4.4.4]][P[o0x01][p10.1.
14.0/25]]/488
                     10.3.34.179     Local           10.3.34.170
[T][O][I0x0][N[c100][b0.0.0.0][a0.0.0.0][r4.4.4.4]][P[o0x01][p4.4.4
.4/32]]/488
                     10.3.34.179     Local           10.3.34.170

Processed 23 prefixes, 23 paths
```

Final Step (3): PCEP



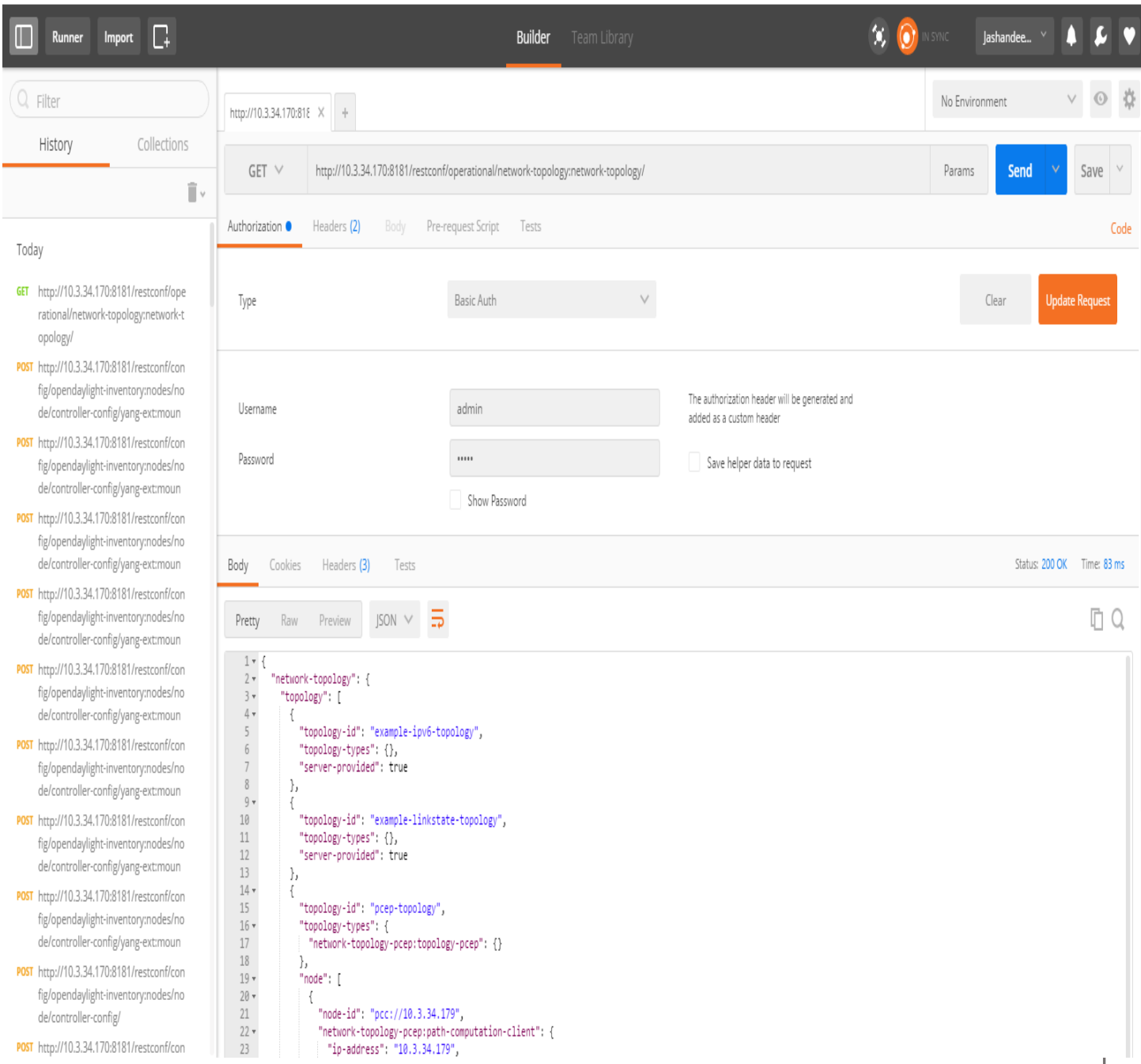
PE1(PCEP protocol with network instantiation)

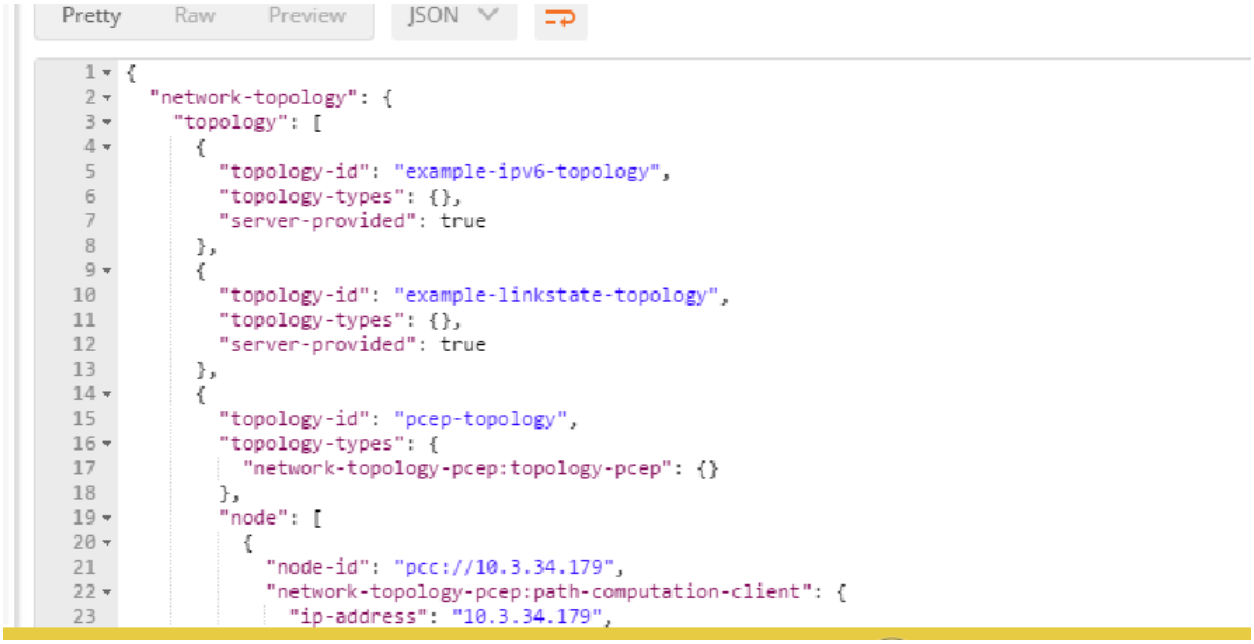
```
RP/0/0/CPU0:PE1#show run mpls traffic-eng
Tue Mar 21 01:22:20.583 UTC
mpls traffic-eng
 interface GigabitEthernet0/0/0/2
 !
 interface GigabitEthernet0/0/0/3
 !
 pce
  peer source ipv4 10.3.34.179
  peer ipv4 10.3.34.170
  !
  stateful-client
   instantiation
  !
 !
 auto-tunnel pcc
  tunnel-id min 1 max 1024
 !
!

RP/0/0/CPU0:PE1#
```

PE1 (PCEP PEER)

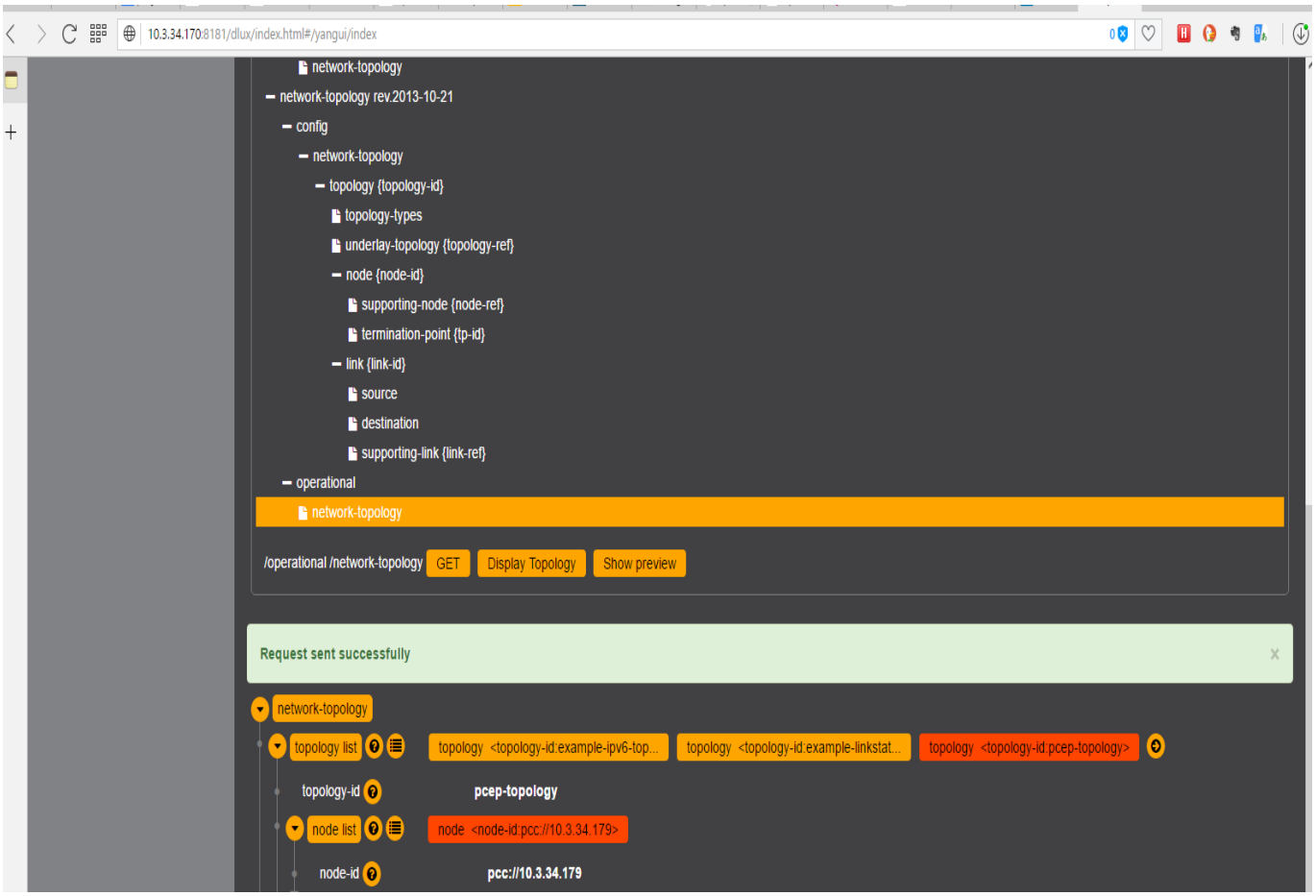
```
RP/0/0/CPU0:PE1# show mpls traffic-eng pce peer
Mon Mar 20 19:07:35.484 UTC
        Address    Precedence        State         Learned From
--------------- ------------ ------------ --------------------
    10.3.34.170           255           Up        Static config
```

OUTPUT of HTTP GET by POSTMAN


Runner
Import
Builder
Team Library
IN SYNC
Jashandee...
Filter
History
Collections
Today
GET http://10.3.34.170:8181/restconf/operational/network-topology:network-topology/
POST http://10.3.34.170:8181/restconf/config/opendaylight-inventory:nodes/node/controller-config/yang-ext:moun
http://10.3.34.170:818
No Environment
GET
http://10.3.34.170:8181/restconf/operational/network-topology:network-topology/
Params
Send
Save
Authorization
Headers (2)
Body
Pre-request Script
Tests
Code
Type
Basic Auth
Clear
Update Request
Username
admin
The authorization header will be generated and added as a custom header
Password
Save helper data to request
Show Password
Body
Cookies
Headers (3)
Tests
Status: 200 OK
Time: 83 ms
Pretty
Raw
Preview
JSON
{
"network-topology": {
"topology": [
{
"topology-id": "example-ipv6-topology",
"topology-types": {},
"server-provided": true
},
{
"topology-id": "example-linkstate-topology",
"topology-types": {},
"server-provided": true
},
{
"topology-id": "pcep-topology",
"topology-types": {
"network-topology-pcep:topology-pcep": {}
},
"node": [
{
"node-id": "pcc://10.3.34.179",
"network-topology-pcep:path-computation-client": {
"ip-address": "10.3.34.179",



Yang output for PCEP:



OUTPUT of NETWORK INSTANTIATION enabled on SDN based PCEP

```
RP/0/0/CPU0:PE1# show mpls traffic-eng pce peer stateful
Mon Mar 20 19:07:49.733 UTC

PCE Address 10.3.34.170
State Up
  PCEP has been up for: 00:02:05
Precedence 255
Learned through:
  Static Config
Sending KA every 30 s
Time out peer if no KA received for 120 s
Tolerance: Minimum KA 10 s

Stateful
  Update capability
  Instantiation capability


KA messages rxed 44 txed 47
PCEReq messages rxed 0, txed 0
PCERep messages rxed 0, txed 0
PCEErr messages rxed 0, txed 0
  Last error received:  None
  Last error sent:  None
PCE OPEN messages: rxed 7, txed 11
PCERpt messages rxed 0, txed 7
PCEUpd messages rxed 0, txed 0
PCEInit messages rxed 0, txed 0
PCEP session ID: local 6, remote 0

Average reply time from peer: 0 ms
Minimum reply time from peer: 0 ms
Maximum reply time from peer: 0 ms
0 requests timed out with this peer

RP/0/0/CPU0:PE1#
```

CHAPTER 8
FUTURE ENHANCEMENT

This project gives a view to further enhance the PCEP protocol by implementing on SDN WAN infrastructure and adding some tremendous features like auto route announce. Also this project demonstrate deployment of opendaylight with cisco Xrv routers which can be extended to multivendor implementation like Juniper, Alcatel and many more.

As cloud content and services proliferate the need for SDN WAN management grows. PCEapp can serve in that role to efficiently manage path placement across the WAN portion of a cloud-based service. Indeed, it would be possible to extend PCEapp to manage WAN link placement in a service function chain traversing remote clouds.

Note: *PCEapp describes an application running on top of the OpenDaylight controller. Its primary functions are to visualize an IP/MPLS network topology and enable the user to create, modify or delete MPLS TE or Segment Routed (SR) TE Paths between two or more nodes in the network. PCEapp replaces the cumbersome and time-consuming per-router CLI approach to explicit path configuration with a user-interface (UI) providing a global view of the network and simple UI dialogs for path management.*

CHAPTER 9
CONCLUSION

In Conclusion, A PCE-based SDN approach is an ideal solution for carrier companies. SDN's primary intention is to decouple the control plane from data plane, and have a centralized control plane mechanism that takes control and manage for path provisioning. PCEs provide centralized control of paths which are setup for flows in MPLS networks. Deploying a PCE, with suitable policy to the OSS, is the simplest way to gain SDN in conventional Transport network. With the aid of deploying PCEs, carrier provider's MPLS-based network can achieve considerable benefits of inter-domain routing, increased network performance, customizable path computation, and better network resources utilization.

The future of networking will form around SDN controller and its protocols. The aim is to make sure that powerful and fine-quality services and communication to be provided to customers in each part of world with maximum productiveness. In the upcoming future, the network devices will be fully invisible to end users however will be distributed physically and virtually. SDN makes the whole lot manifest and with powerful protocols like BGP and PCEP SDN-WAN will change efficiently and network services will be inexpensive and available to absolutely everyone.

10. REFERENCE

I. http://www.olddog.co.uk/AdrianFarrel-TheRoleOfPCEInAnSDNWorld.pdf
II. https://www.ofcom.org.uk/__data/assets/pdf_file/0013/32143/sdn_report.pdf
III. https://tools.ietf.org/html/rfc7426
IV. https://www.ixiacom.com/company/blog/pcepbgp-ls-based-sdn-approach-ideal-choice-service-providers
V. http://www.telecomlighthouse.com/need-quick-recipe-sdn-wan-mix-bgp-ls-pce/
VI. http://www.packetdesign.com/blog/pce-path-computation-sdn-world/
VII. OpenDaylight, "OpenDaylight: A Linux Foundation Collaborative Project," 2013. [Online]. Available: http://www.opendaylight.org
VIII. "Open networking foundation," 2014. [Online]. Available:https://www.opennetworking.org/
IX. Diego Kreutz, Member, IEEE, Fernando M. V. Ramos, Member, IEEE, Paulo Verissimo Fellow,
X. IEEE,Christian Esteve Rothenberg, Member, IEEE, Siamak Azodolmolky, Senior Member,
XI. IEEE,and Steve Uhlig, Member, IEEE" Software-Defined Networking: A Comprehensive Survey" a. VERSION 2.01
XII. Open networking foundation: SDN Architecture, Issue 1 June, 2014 ONF TR-502
XIII. The PACE project "PCE Primer" http://www.ictpace.net/files/3313/8929/2782/PCE_Primer.
XIV. Network functions virtualisation(NFV): Architectural Framework, ETSI GS NFV 002V1.1.1(2013-10)
XV. http://www.brianlinkletter.com/using-the-opendaylight-sdn-controller-with-the-mininet-network-emulator/