**AALBORG UNIVERSITY**

Master of Science Degree in Engineering

Innovative Communications Technologies and Entrepreneurship

Master Thesis

**Energy Efficient Mechanism for Next Generation Networks: Adaptive Resource Allocation**

Anton Nikolaev Katov

June, 2014

Submitted by: …………………………………….

(Anton Nikolaev Katov)

Submitted on: …………………………………….

**Abstract**

The objective of this project is to propose a methodology for resource consolidation, which minimizes the power consumption of a large network, in which the resource overprovisioning is quite big. The project focuses on the operation of core MPLS networks. The proposed approach is based on a Software Defined Networking (SDN) scheme with a reconfigurable centralized controller, which turns off network elements (nodes/links). The switch on/off pattern of the elements is determined by a dynamic database, based on customers’ consumption statistics and real time network statistics. The proposed methodology describes the process of identifying time periods with lower traffic demand; ranking the network elements, based on their utilization and criticality; rerouting the traffic off the least utilized elements; and finally switching off the appropriate nodes or links. An algorithm for traffic rerouting, based on MPLS traffic engineering techniques is proposed and its performance is evaluated in terms of the achieved energy efficiency in accordance with predefined connectivity and quality of service constraints.

**Table of Contents**

[Abstract iii](#_Toc389640568)

[Chapter 1 1](#_Toc389640569)

[Introduction 1](#_Toc389640570)

[1.1. Problem Formulation 2](#_Toc389640571)

[Chapter 2 3](#_Toc389640572)

[Software-Defined Networking 3](#_Toc389640573)

[2.1. SDN Advantages 3](#_Toc389640574)

[2.2. SDN main characteristics 4](#_Toc389640575)

[2.3. OpenFlow protocol 6](#_Toc389640576)

[2.4. SDN Applications 7](#_Toc389640577)

[Chapter3 8](#_Toc389640578)

[Multi-Protocol Label Switching (MPLS) 8](#_Toc389640579)

[3.1. Basic concepts in MPLS 8](#_Toc389640580)

[3.1.1. Forwarding Equivalence Class (FEC) 9](#_Toc389640581)

[3.1.2. Label 9](#_Toc389640582)

[3.1.3. Label Switching Router (LSR) 10](#_Toc389640583)

[3.1.4. Label Switched Path (LSP) 10](#_Toc389640584)

[3.1.5. Label Distribution Protocols (LDP) 11](#_Toc389640585)

[3.1.6. LSP tunneling 12](#_Toc389640586)

[3.1.7. Multi-level label stack 12](#_Toc389640587)

[3.1.8. Label Information Base (LIB) 12](#_Toc389640588)

[3.2. MPLS Operation 13](#_Toc389640589)

[3.3. MPLS Applications 13](#_Toc389640590)

[3.3.1. Virtual Private Network (VPN) 13](#_Toc389640591)

[3.3.2. Traffic Engineering (TE) 15](#_Toc389640592)

[3.3.3. Quality of Service (QoS)/Class of Service (CoS) 15](#_Toc389640593)

[3.4. MPLS RSVP TE Summary 16](#_Toc389640594)

[3.4.1. General Configuration 16](#_Toc389640595)

[3.5. MPLS and SDN 18](#_Toc389640596)

[3.5.1. MPLS TE and MPLS VPNs with OpenFlow 19](#_Toc389640597)

[3.5.2. Hybrid MPLS SDN Network 20](#_Toc389640598)

[3.5.3. Current project approach 21](#_Toc389640599)

[Chapter 4 23](#_Toc389640600)

[Core Network Planning and Traffic Engineering 23](#_Toc389640601)

[4.1. Core network capacity planning overview 23](#_Toc389640602)

[4.2. Traffic demand matrices 24](#_Toc389640603)

[4.2.1. IP traffic matrix estimation techniques 25](#_Toc389640604)

[4.2.2. Statistics retrieval and usage 27](#_Toc389640605)

[4.3. Overprovisioning factor estimation 27](#_Toc389640606)

[Chapter 5 30](#_Toc389640607)

[Energy Efficiency Mechanisms 30](#_Toc389640608)

[5.1. Switching off network elements 30](#_Toc389640609)

[5.1.1. Energy-Saving Algorithm 30](#_Toc389640610)

[5.1.2. Dynamic link metric 31](#_Toc389640611)

[5.1.3. Switching off network elements 33](#_Toc389640612)

[5.1.4. Green game 34](#_Toc389640613)

[5.2. Summary and comparison 35](#_Toc389640614)

[Chapter 6 37](#_Toc389640615)

[Proposed Solution 37](#_Toc389640616)

[6.1. Proposed architecture 37](#_Toc389640617)

[6.2. General overview of the switch off/on process 38](#_Toc389640618)

[6.3. Rerouting algorithm representation 41](#_Toc389640619)

[Chapter 7 44](#_Toc389640620)

[Simulation Setup 44](#_Toc389640621)

[7.1. OMNeT++ 4.x Integrated Development Environment 44](#_Toc389640622)

[7.2. Simulation Setup 45](#_Toc389640623)

[7.2.1. Network topology 45](#_Toc389640624)

[7.2.2. Simulation Setup 46](#_Toc389640625)

[7.2.2.1. RSVP-TE capable routers (LSRx and ELSRx) 48](#_Toc389640626)

[7.2.2.2. Burst Hosts (recipx and senderx) 49](#_Toc389640627)

[7.2.2.3. IPv4NetworkConfigurator (configurator) 50](#_Toc389640628)

[7.2.2.4. LifecycleController 50](#_Toc389640629)

[7.2.2.5. ScenarioManager 50](#_Toc389640630)

[7.3. Simulation Scenario 50](#_Toc389640631)

[Chapter 8 54](#_Toc389640632)

[Result Analysis 54](#_Toc389640633)

[8.1 Initial network state 54](#_Toc389640634)

[8.1.1 Maximum utilization 54](#_Toc389640635)

[8.1.2 Latency 55](#_Toc389640636)

[8.1.3 Packet Loss 56](#_Toc389640637)

[8.2 Network performance after switching off process 56](#_Toc389640638)

[8.2.1 Network Utilization 56](#_Toc389640639)

[8.2.2 Latency 57](#_Toc389640640)

[8.2.3 Packet loss 57](#_Toc389640641)

[8.3 Achieved energy savings 58](#_Toc389640642)

[8.3.1 A glance at the big picture 59](#_Toc389640643)

[Chapter 9 60](#_Toc389640644)

[Conclusion and future work 60](#_Toc389640645)

[References 61](#_Toc389640646)

[Appendix A: *omnetpp.ini* 66](#_Toc389640647)

[Appendix B: *package.ned* 70](#_Toc389640648)

[Appendix C: *\_scenario.xml* 75](#_Toc389640649)

**List of Figures and Tables**

[*Fig. 2.1 Software-Defined Network Architecture, [15] 10*](#_Toc389593572)

[*Fig. 2.2 OpenFlow Instruction Set, [15] 11*](#_Toc389593573)

[*Fig. 3.1 MPLS packet header, [28] 14*](#_Toc389593574)

[*Fig. 3.2 MPLS Network, [30] 15*](#_Toc389593575)

[*Fig. 3.3 MPLS-based L3 VPN, [30] 19*](#_Toc389593576)

[*Fig. 3.4 Basic Scheme of SDN MPLS network, [29] 24*](#_Toc389593577)

[*Fig. 3.5 SDN architecture that decouples the edge control plane and the fabric control plane, [44] 25*](#_Toc389593578)

[*Fig. 3.6 Hybrid SDN MPLS architecture 26*](#_Toc389593579)

[*Fig. 4.1 Network planning methodology, [37] 28*](#_Toc389593580)

[*Fig. 4.2 Demand deduction estimation technique, [41] 31*](#_Toc389593581)

[*Fig. 4.3 Simulation results, [39] 32*](#_Toc389593582)

[*Fig. 5.1 Dynamic link metric flow diagram, [13] 36*](#_Toc389593583)

[*Fig. 5.2 Turning off algorithm, [47] 37*](#_Toc389593584)

[*Fig. 6.1 Software Defined Networking Controller Scheme 41*](#_Toc389593585)

[*Fig. 6.2 Flow diagram of the switch off/on process 43*](#_Toc389593586)

[*Table 6.1 Node ranking database 44*](#_Toc389593587)

[*Fig. 6.3 Re-routing algorithm flow diagram 46*](#_Toc389593588)

[*Fig. 7.1 Super core network topology 50*](#_Toc389593589)

[*Fig. 7.2 OMNeT++ implementation of the super core network topology 51*](#_Toc389593590)

[*Fig. 7.3 RSVP-TE router module 52*](#_Toc389593591)

[*Fig. 7.4 Burst Host module 53*](#_Toc389593592)

[*Fig. 7.5 RSVP session sequence chart 56*](#_Toc389593593)

[*Fig. 7.6 Switching off scenario: a) switching off sequence; b) resulting topology 56*](#_Toc389593594)

[*Fig. 8.1 Maximum Utilization Bar Chart 58*](#_Toc389593595)

[*Fig. 8.2 Maximum end to end delay values for every BurstHost traffic sink 59*](#_Toc389593596)

[*Fig. 8.3 End to end delay vector line chart 59*](#_Toc389593597)

[*Fig. 8.4 Mean utilization vector: a) RSVP.LSR3 b) RSVP.LSR5 c) RSVP.LSR8 60*](#_Toc389593598)

[*Fig. 8.5 End to end delay vector line chart 61*](#_Toc389593599)

[*Fig. 8.6 Achieved energy conservation chart 62*](#_Toc389593600)

**List of Abbreviations**

**API** Application Programming Interface

**AS** Autonomous System

**ASIC** Application Specific Integrated Circuit

**ATM** Asynchronous Transfer Mode

**BGP** Border Gateway Protocol

**CE** Customer Edge

**CoS** Class of Service

**CR-LDP** Constrained-based Routing using LDP

**CSPF** Constrained Shortest Path First

**DiffServ** Differentiated Services

**DS3** Digital Signal 3

**EEE** Energy Efficient Energy

**FEC** Forwarding Equivalence Class

**GRE** Generic Routing Encapsulation

**IDE** Integrated Development Environment

**IEEE** Institute of Electrical and Electronics Engineers

**IETF** Internet Engineering Task Force

**IGP** Interior Gateway Protocol

**ILP** Integer Linear Programming

**IntServ** Integrated Services

**IPFIX** IP Flow Information eXport

**IPSec** Internet Protocol Security

**IS-IS** Intermediate System to Intermediate System

**LDP** Label Distribution Protocol

**LER** Label Edge Router

**LIB** Label Information Base

**LIFO** Last-in First-out

**LSP** Label Switched Path

**LSR** Label Switching Router

**LTE** Long-Term Evolution

**L2TP** Layer 2 Tunneling Protocol

**MP-iBGP** Multi-Protocol interior BGP

**MPLS** Multi-Protocol Label Switching

**MPLS-TP** Multi-Protocol Label Switching – Transport Profile

**NAT** Network Address Translation

**ONF** Open Network Foundation

**OP** Overprovisioning Factor

**OPEX** Operating Expense

**OSPF** Open Shortest Path First

**OTWG** Optical Transport Working Group

**PE** Provider Edge

**POP** Point of Presence

**PPTP** Point-to-Point Tunneling Protocol

**QoS** Quality of Service

**RFC** Request for Comments

**RSVP-TE** Resource Reservation Protocol for Traffic Engineering

**SDN** Software-Defined Networking

**SLA** Service Level Agreement

**SNMP** Simple Network Management Protocol

**SONET** Synchronous Optical Networking

**TE** Traffic Engineering

**TCP** Transmission Control Protocol

**TDM** Time-division Multiplexing

**TTL** Time to live

**UDP** User Datagram Protocol

**VC** Virtual Circuit

**VPLS** Virtual Private LAN Services

**VPN** Virtual Private Network

**VRF** Virtual Routing and Forwarding

**WAN** Wide Area Network

**Chapter 1**

**Introduction**

The development of 3G and 4G wireless technologies introduced an immense number of new services and applications that are now ubiquitously available to the mobile data users. The ever-increasing variety of high-bandwidth graphics and video applications dominate the wireless networks and generate a huge demand for high speed access. As a result LTE has experienced the most successful launch of any mobile technology in history. According to a survey published by Informa Telecoms & Media in partnership with Mobidia [1] more than 110 LTE networks have already been deployed globally and address close to half a billion people in more than 50 countries. The adoption of the technology reached 50 million people in record time. The enhanced data rates of theoretically up to 50Mbps uplink and 100 Mbps downlink have led to a steady increase of the average data consumption per user that reaches more than 130% in South Korea, compared to the average consumption of 3G mobile data users.

With this surge in traffic growth the network operators face the need to develop and implement new efficient and cost-effective solutions, characterized by all-IP Ethernet backhaul and large increase in mobile backhaul and core capacity. For example, according to [2] the backhaul capacity requirement for a suburban radio cell site will increase gradually and level off at around 75-90Mbps. As to the core capacity some mobile service provides estimate more than 3-times increase of their core bandwidth requirements, for example from less than 40 Gbps to 130 Gbps.

The increase of bandwidth demand also leads to significant transformation in the backhaul requirements and migration to all-IP Ethernet backhaul. The predominant technology of choice is identified to be IP/MPLS [3] and the newly develop variation of IP/MPLS – MPLS-TP [4].

These design requirements tend to meet the peak demand in the network. However, research results show considerable variations in network utilization, based time on the day, week, etc. or based on the customer preferences [7]. This introduces the need of adaptive solutions for resource allocation, which can bring benefits to both network operators and end users and introduce considerable energy savings.

One of the implications of the significant increase in traffic demand is an increase in the energy consumption of the underlying network of the data services providers. In general the power used up by the ICT sector has shown growth by an average rate of 10% per year and is estimated to account for about 1.8% of the total worldwide energy consumption [10]. Consequently, the problem related to the energy efficiency of the communications network infrastructure has started to attract more attention. The proposed solutions include the Energy Efficient Ethernet (EEE) standard, ratified by IEEE in 2010, which aims to reduce the overall power consumption of Ethernet ports by 50%. The standard’s main point is to introduce low power mode for idle link intervals. Performance evaluation of EEE is presented in [8]. In [9] a green reconfigurable router energy efficient design is described, which addresses topics such as power-aware routing through green virtual networks and rate adaptive processing inside routers. An overview of the energy efficient solutions in optical networks is given in [10]. The concept of switching on and off network nodes and links is presented in [12]. The proposed algorithm switches off network links or nodes iteratively based on predetermined link/node ranking and then verifies if the new topology can meet the current traffic demand. Their results show that the number of active links can be reduced by up to 25% and the number of nodes by up to 10%, thus offering considerable energy savings. In [13] a modification of OSPF protocol that allows aggregation of traffic load to certain links is proposed. This is achieved by modifying the link weights. In [14] different approach for resource consolidation is proposed. The evaluation of the elements that can be safely switched off is based on the estimation of the Sharpley value.

## Problem Formulation

In this project the problem of over-provisioning of network infrastructure is addressed. According to [11], the process of capacity planning of core and backhaul networks ensures the provisioning of sufficient bandwidth that guarantees certain SLA requirements. To battle traffic burst, however, the bandwidth is over-provisioned relative to the measured average rate. In mobile networks, however, there are significant variations in network utilization, based on time of the day, week, etc. or based on the customer preferences. The objective of this project is to develop energy efficient mechanism for resource allocation and node operation in the context of backhaul and core networks. The differentiation will be based on a database characterizing the user behavior. The database might be based on network operator’s statistics and should introduce a set of classes that define different usage patterns based on time of the day, time of the week and user preferences. Each class should be associated with different amount of provided bandwidth. This will be achieved by switching off entire nodes or just device ports and rerouting the traffic through the active nodes, using traffic engineering techniques.

**Chapter 2**

**Software-Defined Networking**

Software-defined networking (SDN) is a new networking architecture that is proposed as a facilitating technology for network evolution and network virtualization. It has attracted significant attention from both academic researchers and industry. One the main organizations that contribute to the development of SDN is the Open Network Foundation (ONF) – a non-profit industry consortium of network operators, service providers and vendors that promotes the SDN architecture and drives the standardization process of its major elements. [15, 16]

In [15] ONF defines SDN as a technology where “network control is decoupled from forwarding and is directly programmable”. It concentrates the network intelligence in software-based central controllers, which aims to bring better and more efficient control, customizability and adaptability. As a result the networking devices are turned into packet forwarding equipment that can be programmed via an open interface. The main benefits that the SDN technology might offer are listed in [15] and summarized below:

* Centralized unified control of network devices from different vendors
* Better automation and control, as an abstraction of the real network is created
* Simplified and quicker implementation of innovations, as the network control is centralized and there is no need every individual device to be reconfigured
* Improved network reliability and security, because of fewer configuration errors and unified policy enforcement, provided by the automated management and the centralized control
* Ability to easily adapt the network operation to changing user needs, as centralized network state information is available and can be exploited



## SDN Advantages

The rapid development of new trends in computing, such as server virtualization, cloud services, immense variety of mobile data applications, etc., determine the need of new network architectures that can successful cope with the changing environment. Some of the key points summarized in [15] and listed below:

* **Changing traffic patterns** – a main characteristics is the shift from the traditional client-server applications, where one client accesses one server, to application where multiple databases and servers are accessed before the data is returned to the client
* **Rise of the mobile data services –** the use of mobile devices (smartphones, notebooks, tablets, etc.) increases immensely. These devices have to be accommodates as a part of the corporate networks in order to meet compliance requirements and protect the privacy of corporate data and intellectual property
* **Growth of cloud services** – the access on demand of applications, infrastructures or others IT resources introduces the need of scalability and adaptability of computing storage and network resources
* **Big data processing** – presents the need for parallel processing on multiple servers, which need reliable connection to each other and high available bandwidth

As a result the traditional network design struggle to meet the requirements of the rapidly changing demand patterns, because of the constraints of the current architectures, such as:

* **Static nature** – the large number of the employed protocol, each of which meets specific needs of the network design, make changes in large networks rather troublesome. One of the primary reasons for this is the fact that the network management tools operate on device level and therefore any topology change triggers manual reconfiguration process of multiple devices. To avoid the chance of service interruption, such changes are usually avoided. This static nature of the contemporary network architectures severely contradicts to the dynamic nature of the traffic patterns.
* **Incoherent policies –** to implement a new consistent policy throughout the network may require reconfiguration of thousands of devices, which sometimes might be impossible. The incoherence might lead to security vulnerability, non-compliance with regulations, etc.
* **Scalability limitations –** the huge rise of traffic demand might require changes in scale of the network architecture. The rise in the number of devices, however, highly increases complexity.
* **Vendor dependence –** network operators strive to meet customer requirements by constantly evolving the provided services and introducing high-customization. This might require specific configuration of the employed devices, which can be impossible because of the lack of open interfaces.

The existing incompliance between network capabilities and customer requirements determine the need for diverse solution. SDN architecture is being developed to meet the emerging conditions.

## SDN main characteristics

The contemporary network infrastructure typically consists of communications links that connect switching devices such as switches and routers. The decisions about data flowing through the network are performed by each network element, which means that the data and control planes are closely coupled. The implementation of new applications on such architectures should therefore be deployed directly on the infrastructure, which might be a very complicated task because of the lack of common control interface to various devices. To complicate further common network equipment usually holds limited vendor specific interfaces that restrict network changes, such as deploying new protocols or updated versions of the existing network protocols. [16]

SDN has been designed to simplify network configuration and facilitate innovation. SDN paradigm decouples the control plane and the data plane and concentrates the data forwarding decisions into a centralized software controller. As a result the underlying network devices’ functions are reduced to simple data forwarding. Instead of programming thousands of devices the network configuration is performed on simplified network abstraction. This allows the implementation of various software modules that can exert dynamic control on the network functions. [15] A basic scheme of SDN architecture is presented on figure 2.1.

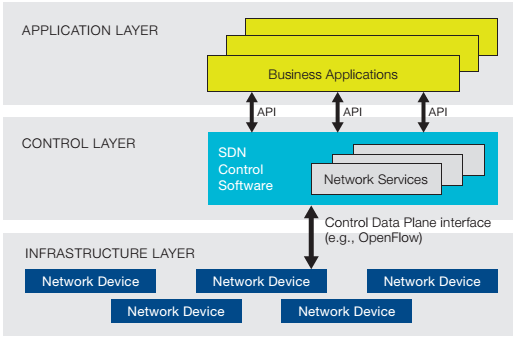


Fig. 2.1 Software-Defined Network Architecture, [15]

The centralized control function of the SDN architecture allows consistent policies to be enforced with ease. Common networking functionalities can also be configured via the supported APIs. The deployment of services, such as routing, security, access control, bandwidth management, traffic engineering, quality of service, energy optimization can be configured much easily. According to [15] the goal of the SDN developers is to ensure multi-vendor support.

## OpenFlow protocol

OpenFlow is a standardized protocol that defines the communication between the control and the data forwarding plane in the SDN architecture. It moves the control out of the networking devices (routers, switches, etc.) into the centralized controller. The protocol uses the concept of flows that use match rules to determine how the packets will be handled. The protocol is configured on both sides – the device and the controller. The forwarding device in an OpenFlow scenario is an OpenFlow switch that contains one or more flow tables and an abstraction layer that communicates with the controller. The flow tables are filled with flow entries which define how the packet will be forwarded, depending on the particular flow they are part of. [15, 16] The flow entries have the following fields:

* *match fields –* might contain information from the packet headers, ingress port or metadata and matches the packets to a certain flow
* *counters –* collect statistic about the particular flow
* *actions –* define how the incoming packets to be handled

An example of the OpenFlow instruction set is presented on figure 2.2.

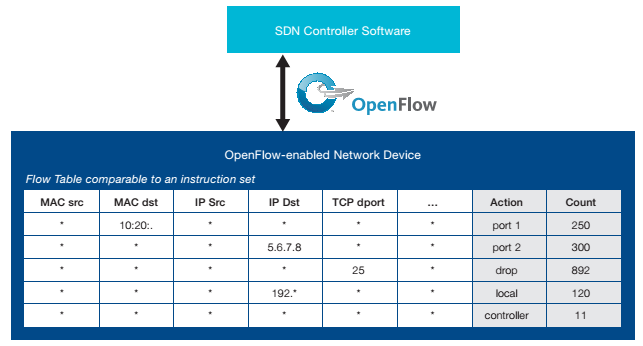


Fig. 2.2 OpenFlow Instruction Set, [15]

An OpenFlow switch essentially receives data packets, extracts the packet header and matches the value to the entries in the flow table. If the value is found the packet is forwarded according to the instructions in the *actions* fields. In case the value does not match any of the entries, the packet is handled according to the instructions defined in the *table-miss* entry. The packet can be either dropped, forwarded to the next flow table or send to the OpenFlow controller via the control channel. Another possibility, employed in switches that have both OpenFlow and non-OpenFlow ports, is to forward the packet using standard IP-forwarding schemes. The OpenFlow switch communicates with the controller over a secure channel. The controller adds, removes or updates the entries in the flow table. [15]

## SDN Applications

The SDN architecture is claimed to greatly simplify network management and provide an immense number of new services via the programmable software modules. A summary of the application scenario that will benefit from employing the OpenFlow architecture are described in [16] and briefly summarized below.

* **Enterprise networks** – the centralized control function of SDN can be particularly beneficial for enterprise networks in different ways. For example, network complexity can be reduced by removing middleboxes and configuring their functionality within the network controller. Different network functions implemented via SDN include NAT, firewalls, load balancers [18], [19] and network access control [20]. An approach for realizing consistent network upgrade, using high-level abstractions is described in [21].
* **Data centers** – power consumption management is a major issue in data centers, as they often operate below capacity in order to be able to meet peak demands. In [22] a network power manager is described that turns off a subset of switches in a way to minimize power consumption while ensuring the required traffic conditions. A real life example of SDN application in the context of data centers is presented in [23] and [24]. They describe SDN-based network connecting Google data centers worldwide. The deployment was motivated by the need of customized routing and traffic engineering, as well as scalability, fault tolerance and control that could not be achieved with traditional WAN networks.
* **Infrastructure-based wireless access networks** – an SDN solution for enterprise wireless LAN networks is proposed in [25]. The solution builds an abstraction of the access point infrastructure that separates the association state from the physical access point. The purpose is to ensure proactive mobility management and load balancing.
* **Optical Networks** - According to the Optical Transport Working Group (OTWG), part of the Open Network Foundation (ONF), SDN solutions for optical transport networks can provide multiple advantages, such as improving optical transport network control and management flexibility, enabling deployment of third-party management and control systems and deploying new services by leveraging virtualization and SDN [26].

**Chapter3**

**Multi-Protocol Label Switching (MPLS)**

MPLS is a technology that aims to combine the advantages of Layer 2 fast switching and Layer 3 routing and forwarding. The main characteristic of MPLS is that it implements a connection-oriented model over the traditional connectionless IP routed networks. MPLS gained huge popularity over the years and is identified as a key enabling technology for IP-based converged telecommunication services. MPLS was first standardized by IETF in [RFC 3031](http://tools.ietf.org/html/rfc3031) in 2001, but constantly continues to evolve with the introduction of MPLS over IPv6 in RFC 4029 (March 2005) and MPLS Transport Profile (MPLS-TP) in RFC 6215 (April 2011). The most recent research topics focus on the implementation of the MPLS control plane using OpenFlow and SDN ([28], [29], [30]).

MPLS has been widely adopted by service providers around the world as replacement of ATM and Frame-Relay. According to a study by Nemertes Research, quoted in [31], 84% of the enterprises have been using MPLS for their WANs by 2009. The immense success of MPLS is explained not only with the numerous advantages the technology brings, but also with its compatibility with the old technologies. The main benefits the technology brings are listed in [30] and summarized below:

* MPLS can be implemented over the most widely used legacy or new infrastructures (DS3, SONET, 10/100/1000/10G Ethernet) and networks (IP, ATM, Frame Relay, Ethernet, and TDM). Thus it enables the migration towards IP-based infrastructures.
* MPLS provides traffic engineering capabilities, which enables more efficient use of the available bandwidth
* MPLS supports Quality of Service (QoS)
* MPLS reduces core routers processing requirements since they simply forward packets based on labels
* MPLS provides the appropriate level of security and reduces the need of encryption on public IP networks
* MPLS enables the deployment of provider-network-based VPNs, which reduces the configuration and management complexity for the end customer.



## Basic concepts in MPLS

MPLS is a technology that optimizes the traffic forwarding in a network by avoiding complex lookups in the routing table. The traffic is directed based on labels contained in an MPLS packet header. The labels define only the local node to node communication and are swapped on every node. This process allows very fast switching through the MPLS core. MPLS relies on traditional IP routing protocols to determine the best routes and to receive topology updates and predetermines the path the packet will take through the network. This process is performed by the MPLS edge router and thus reduces the processing requirements for the core switching routers. These paths are called Label-Switched Paths (LSPs). [30] The basic concepts of MPLS are summarized in [33] and will be examined in the following sections.

## Forwarding Equivalence Class (FEC)

MPLS technology is based on classification. It groups the packets that will be forwarded in the same way in forwarding equivalence classes (FEC). The classification criteria may vary and include source address, destination address, source port, destination port, protocol type and VPN. The packets belonging to a specific FEC will then be forwarded to the same LSP. When a packet arrives the router will examine it and determine whether it belongs to and existing FEC. FEC are neither labels, nor packets, but logical entities created by the router.

## Label

A label is short fixed-length identifier that points to a specific FEC. A label may represent only a single FEC, but a FEC can correspond to multiple labels. The label is part of the packet header and is only locally significant, as it does not carry and topology information. Figure 3.1 presents the label format and its place in the packet header:

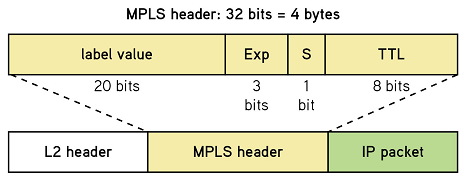


Fig. 3.1 MPLS packet header, [28]

The label is 32 bits in length and consists of the following fields:

* Label value – determines the next hop in the forwarding process and is 20 bits in length
* Exp (experimental) – used for QoS mapping
* S (stacking bit) – used when there are multiple levels of MPLS labels and indicates whether the label is at the bottom of the label stack
* TTL (time to live) – it has the same function as in IP forwarding and essentially represents the hop limit or lifespan of the packets. Once the limit is reached the packet is discarded

The MPLS label serves as a connection identifier and is inserted between the data link layer header and the network layer header.

## Label Switching Router (LSR)

LSRs are the fundamental components of the MPLS network and can be three different types depending on their function in the network:

* Label Edge Router (LER) – situated on the periphery of the network and serves as a gateway between the MPLS network and the WAN or the Internet. A LER can be:
  + Ingress router – it is the entry point of the MPLS network. When a packet arrives it decides whether the packet should be forwarded through the MPLS network, determines the FEC, the packet belongs to, and encapsulates it with an MPLS header, based on the information it carries.
  + Egress router – it is the exit point of the MPLS network. It performs a normal IP look-up and forwards the packet according to the appropriate IP routing protocol.
* Transit router – it is any router in the middle of the MPLS network and performs simple switching, based on the label value.
* Penultimate router – it is the router before the last hop in the MPLS network. As the packet will not be switched to another transit router, the penultimate router removes the MPLS header, before forwarding the packet to the egress router. The use of penultimate router configuration is optional, as the MPLS header can also be removed by the egress router. In that case the penultimate router operates as a transit router.

## Label Switched Path (LSP)

Label switched path defines the path the packets from a particular FEC will follow through the MPLS network. The LSP is a unidirectional path from the ingress to the egress router and functions like a virtual circuit. The LSP is established by a signaling protocol, such as Label Distribution Protocol (LDP) or Resource Reservation Protocol for Traffic Engineering (RSVP-TE). Simple schematic of a MPLS network that illustrates the concept of LSP is presented on figure 3.2:

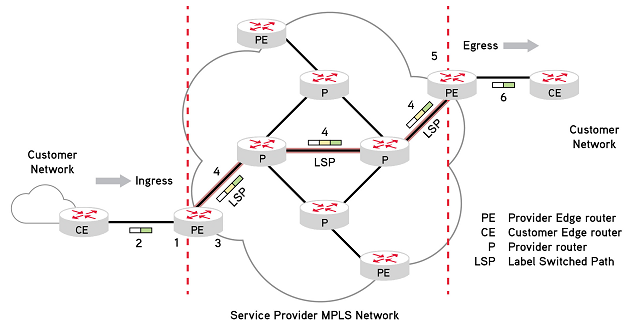


Fig. 3.2 MPLS Network, [30]

## Label Distribution Protocols (LDP)

LDP is a signaling protocol that classifies FECs, distributes label binding information and establishes LSPs. LDP is standardized by IETF and its specifications are defined in RFC 5036. The LDP concepts are described in [34] and summarized below.

The protocol enables the LSRs to request, distribute and release label binding information. The label distribution method is called hop-by-hop forwarding and is performed along the normally routed paths, chosen by the underlying Interior Gateway Protocol (IGP) routing protocol. The resulting label paths (LSPs) are then used to forward label traffic across the MPLS network. The basic concepts of LDP are:

* LDP peers – these are two LSRs that exchange labels and FEC mapping information over an LDP session.
* LDP session – defines the communication interval between the LDP peers
* LDP messages – four main categories
  + *Discovery messages* – LSRs indicate their presence by periodically sending hello messages
  + *Session messages* – used to establish, maintain and terminate LDP sessions
  + *Advertisement messages* – used to create, alter and remove label mappings to a FEC
  + *Notification messages* – used to provide advisory information and distribute error information

There are various label distribution protocols supported by MPLS. They can be defined as two major types:

* Protocols dedicated for label distribution – LDP and Constrained-based Routing using LDP (CR-LDP)
* Existing protocols extended to support label distribution – Border Gateway Protocol (BGP) and Resource Reservation Protocol with extension for Traffic Engineering (RSVP-TE). RSVP-TE operation is described in details in the following chapter

## LSP tunneling

MPLS supports LSP tunneling. The technology allows an LSP tunnel to be established not only on a traditional path, but also on a specific path that is not defined by the routing protocol. In this case the LSP is an LSP tunnel. If the tunnel follows exactly the hop-by-hop route established by the routing protocol, it is called a hop-by-hop routed tunnel. If the tunnel is specified differently, the tunnel is called explicitly routed tunnel. The key application of LSP tunnels is traffic engineering and allows the implementation of a variety of policies that optimize networks performance. For example, the tunnel can redirect the traffic away from failure, congestion or bottlenecks. The protocol used for LSP tunnel configuration is RSVP-TE.

## Multi-level label stack

MPLS allows one LSP to be tunneled inside another LSP. This is realized by encapsulation of a MPLS packet inside another MPLS packet by adding a number of labels, organized as a last-in first-out (LIFO) stack. At the ingress and the egress of each tunnel PUSH and POP operations can be performed on top of the stack. The depth of the stack is no limited and can consist of any number of levels. Label stacking brings two distinct advantages. First, it allows combining LDP and RSVP-TE protocols on different level on the stack, thus combining the advantages of both of them. Second, label stacking allows the creation of MPLS VPNs inside the provider’s network, which brings multiple opportunities for the end customers.

## Label Information Base (LIB)

As network topology is established by the routing protocols and the LSPs are signaled, each LSR builds a label information base (LIB). A LIB is a locally specific table that identifies how the packets should be forwarded. The table contains information about the FECs every label corresponds to and about the outbound port to forward the packet.

## MPLS Operation

A typical MPLS topology with all the corresponding main components is presented on figure 2. The customer edge (CE) router communicates with the provider edge router, which in MPLS networks is also called label edge router (LER). The ingress LER encapsulates the packets with MPLS headers and forwards them through the MPLS core. Within the core the LSRs switch the packets based on the MPLS labels. At the egress side the LER removes the labels and forwards the packet according to the underlying routing protocol. In summary MPLS data flow follows the following steps [30]:

1. Before forwarding the packets the LERs determine the LSPs, based on the routing information from the underlying routing protocol. The LIBs are then built up by the MPLS signaling protocol.
2. When a non-MPLS packet form the CE router is received by the LER, the packet is inspected, mapped to a particular FEC, labeled with the appropriate label and forwarded to the next hop along the LSP.
3. Each transit LSR forwards the packet in accordance with its LIB. The label is swapped at each hop.
4. At the egress LER the packed is stripped off the label and forwarded to the CE router.

## MPLS Applications

The primary reason for the creation of MPLS was to improve route lookup and forwarding speed. With the technological improvements of application specific integrated circuit (ASIC) technology, however, the routing performance was greatly improved and as result this is no longer the main advantage of MPLS. The main benefits the MPLS offers are the seamless integration between layer 2 networks and the improved performance in Traffic Engineering (TE), Quality of Service (QoS) and Virtual Private Networks (VPNs). The main applications of MPLS are presented in [30] and [31] and summarized in the following sections.

## Virtual Private Network (VPN)

Virtual private networks are in general private networks configured over a public network, such as the service provider network or the Internet. VPN provide a unified network connection between geographically remote private networks. In traditional VPN the data between private networks is transported through the public infrastructure over tunneling protocols, such as IPSec, GRE, L2TP and PPTP. The fundamental operation of MPLS is based on LSPs that are essentially tunnels. This gives a natural advantage of the MPLS-based VPNs.

The VPNs vary in their configuration and can be group in two main categories:

* Customer-based – configured entirely on equipment located on the customer premises and using tunneling protocols across the public network. A very widely-used protocol is IPSec, which adds encryption to IP.
* Network-based – configured on the service provider equipment and managed by the provider, such as MPLS VPNs.

MPLS VPN is claimed to provide distinct advantages over the customer-based VPN, such increase scalability and cost savings, as it is supported on the provider’s equipment. Furthermore, a study of Miercom cited in [30] showed that MPLS VPNs reduce the need of encryption and provide the same level of security as Frame Relay or ATM.

Two main types of VPN can be distinguished:

* **L3 VPNs** – the most widely spread MPLS VPN configuration is based on the Multi-Protocol interior BGP (MP-iBGP) extension of BGP. MP-iBGP is used to distribute the VPN routing information across the provider network. The routing information of the customer’s network is fed to the provider edge (PE) router by the customer edge (CE) router. The PE router creates a Virtual Routing and Forwarding (VRF) table and stores this private routing information. A separate VRF table is maintained for every separate VPN and the users have access only to sites within the same VPN. The IP addressing scheme is managed by the provider and consistent with the customer’s preferences. The L3 VPN data packets are encapsulated using a two-level MPLS label stack. The inner level carries the VPN specific information and the outer label is used for forwarding the data through the provider MPLS. A basic L3 VPN network topology is presented on figure 3.3.

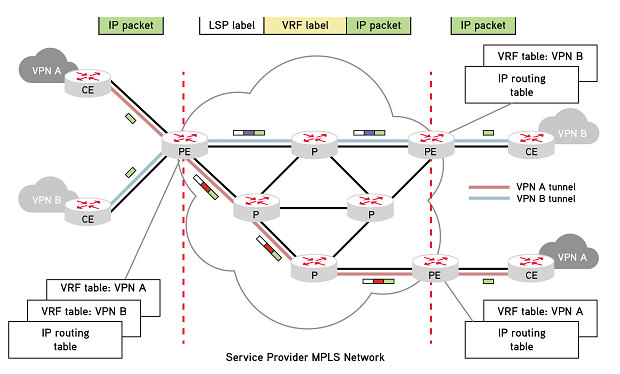


Fig. 3.3 MPLS-based L3 VPN, [30]

* **L2 VPNs** – the L2 VPN concept distinguishes between two types – point-to-point or multi-point, also referred as Virtual Private LAN Services (VPLS). VPLS use Ethernet interface and enables geographically separated networks to be connected on the same layer 3 networks over the provider’s infrastructure, thus, simplifying the configuration from the customer side. In L2 VPNs, in contrast with the layer 3 configuration, the connection between the CE and the PE routers is only a layer 2 connection. The PE router then switches the incoming traffic into preconfigured tunnels. The L2 VPNs also use two-level MPLS label stack. The inner label is a Virtual Circuit (VC) label that identifies the VLAN, VPN or connection at the end point. The advantage of VPLS is that it can transparently carry any enterprise protocol configured on the customer network. The disadvantage of the technology is the scaling limitations, as a full mesh of LSPs must be configured between all the customer sites.

## Traffic Engineering (TE)

Traffic Engineering (TE) is utilized to provide better bandwidth management and utilization by redirecting some of the traffic through suboptimal explicit path in addition to the shortest path defined by the underlying IGP. The explicit path is usually determined in compliance with specific bandwidth guarantees. This is realized by Constrained-Based Routing algorithms. MPLS TE provides resilience and reliability by optimizing the LSP connections and enabling backup LSPs in case of failure. This is realized by RSVP-TE Fast Reroute.

TE in MPLS is realized via extensions of the IGPs, such as OSPF and IS-IS, that allows the protocol update to carry additional information, such as link bandwidth, utilization, delay, priority, etc. MPLS TE engineering is configured, using signaling protocols, such as RSVP-TE. It is usually implemented in the core of the MPLS, in contrast with the QoS that is deployed at the edge.

## Quality of Service (QoS)/Class of Service (CoS)

As it was mentioned above QoS is configured at the edge of the MPLS network and enables high priority packets to be forwarded preferentially. QoS and CoS can be provided in MPLS networks, where traffic engineering is deployed, and give the opportunity SLA agreements with the customers to be established.

Two fundamental mechanisms can be differentiated:

* Integrated Services (IntServ) – this mechanism offers guaranteed traffic parameters end-to-end. The QoS requirements are signaled across the network, using RSVP and uses hard allocation of resources.
* Differentiated Services (DiffServ) – this a less strict mechanism that provides CoS by classifying traffic into different priority level, but does not provide end-to-end guarantees of the service. The DiffServ data is included in the IP packet headers and is mapped in to the label information.

QoS and TE greatly improve the network performance and are among the fundamental concepts that turn IP/MPLS into the key enabling technology for providing converged telecommunication services over a single network infrastructure.

## MPLS RSVP TE Summary

Resource reservation protocol (RSVP) is a signaling protocol for MPLS systems that reserves resources for IP unicast or multicast flows. RSVP traffic engineering (TE) is an extension of the protocol that allows the establishment of label switched paths (LSP) that can be used for traffic engineering. It is mainly used in core networks to ensure that QoS requirements are fulfilled and to enable load balancing between the available paths. TE essentially redirects some of the traffic through the suboptimal path, thus allowing better bandwidth management and utilization. [27]

In the current project however RSVP TE is employed to steer the traffic off the underutilized links during low demand time periods, such as in night time. The traffic is then concentrated in a subset of node/links, which permits the rest of the nodes/links to be switched off safely without experiencing packet loss. The technique will be explained in the following paragraphs.

## General Configuration

A summary of the RSVP TE general configuration is described at [27] and summarized below. The label switched paths (LSP) or traffic engineering tunnels are used to link provider edge (PE) routers across the core of the service provider network. Generally, TE tunnels are mapped one-to-one with the MPLS LSPs. TE tunnels or LSPs essentially represent data flows between specific source and destination and can have various attributes associated with them. Such attribute might be, for example, bandwidth requirements. LSPs define the path through the MPLS core and are set up by a signaling protocol such a RSVP TE. The signaling protocol links LSPs to specific label values. Data packets are then propagated through the path using MPLS label switching.

When RSVP TE is used each LSP has a bandwidth value associated with it. For a given LSP the utilized routing algorithm then selects the shortest path with enough available bandwidth. This is done using constrained shortest path first algorithm (CSPF). OSPF or IS-IS with extensions for traffic engineering can be used.

As was already mentioned LSPs connect two PE routers through the MPLS core, therefore the LSPs are configured on the edge or headend routers. The headend router receives information about the shortest path with enough available bandwidth on the links from the IGP (OSPF, IS-IS, etc.) updates. The LSP is then defined by the headend router and mapped to the TE tunnel. The information is then propagated to the other routers in the LSP and the required bandwidth is reserved by the RSVP TE protocol.

The steps for creation of a MPLS TE tunnel LSP are depicted below:

1. The implemented IGP on the headend router performs CSPF calculation to determine the shortest path with available resources. The output is in the form of an ordered set of IP addresses.
2. The information about the next-hop addresses in the TE LSP is then passed to the RSVP process.
3. The RSVP then performs resources request and confirmation by generating PATH and RESERVATION messages.
4. Once the process is completed RSVP signals that the LSP is established.
5. The TE tunnel is then available for the IGP to be used.

RSVP path messages are sent by the headend router to identify the path and the used resources. As the path messages traverse the network the information about the resource availability of the participating routers is obtained and stored. In this way the information about the traffic flow is propagated and it is determined whether the bandwidth requirement can be fulfilled.

There are four main messages used in the RSVP TE signaling process:

* ***RSVP PATH message*** – generated by the headend router and traversed through the LSP path. At every hop it checks the availability of resources and stores the information. The path message contains the following objects:
  + *SESSION –* defines the source and destination by the IP addresses of the corresponding loopback interfaces of the headend and tailend routers
  + *SESSION\_ATTRIBUTE –* defines the bandwidth requirement and necessary resources for the specific LSP tunnel
  + *EXPLICIT\_ROUTE –* contains a list of next hops, determined either manually or by the implemented constrained-based SPF
  + *RECORD\_ROUTE -* stores the local router’s outgoing interface address part of the LSP path
  + *SENDER\_TEMPLATE –* defines the interface address that will be used as LSP-ID for the tunnel
* ***RSVP RESERVATION message*** – generated by the tailend router to confirm the reservation of resources. It also performs the mapping of the TE tunnel to the particular LSP. It is performed first by the tailend router and then propagated through the LSP.
* ***RSVP ERROR messages*** – generated in case the required resources are not available.
* ***RSVP TEAR messages* –** these messages clear the *PATH* or the *RESERVATION* state on the routers.

The steps of the RSVP TE protocol follows are described below:

1. The objects of the PATH messages mentioned above are defined by the headend router and the message is sent to the next hop router in LSP.
2. As the next hop router receives the path messages, it removes the attributes that define its own interface from the EXPLICIT\_ROUTE object and records information about the outgoing interface in the RECORD\_ROUTE object. This process is repeated in every next hop.
3. When the PATH message reaches the tailend router it triggers the generation of RESERVATION message. The messages defines the POP label of the LSP tunnel and reinitiates the RECORD\_ROUTE object, that contains the outgoing interface of every router in the LSP path back to the headend router.
4. The intermediate routers in the LSP path prepend the RECORD\_ROUTE object with the outgoing interface and generate the value of the LABEL object in the RESERVATION message.
5. When the headend router receives the RESRVATION message, the traffic engineered LSP, associated with specific bandwidth or resource requirements, is defined by the RECORD\_ROUTE object.

## MPLS and SDN

The possibility of hybrid implementation of MPLS network capabilities via SDN controller is a hot topic in the telecommunications sector nowadays. The combination of MPLS data forwarding capabilities with SDN’s orchestration capabilities has been suggested to bring multiple advantages [43] that could help the network operators to quickly adapt to the emerging service trends and take full advantage of the technological improvements.

* **Provides centralized control and reduces the complexity of network operations** – the main benefits of MPLS networks are the improved performance in TE and VPNs. The implementation of these services, however, requires the deployment of multiple protocols (such as OSPF, LDP, RSVP TE and possibly iBGP and MP BGP), which results in high configuration complexity, excessive traffic of update and control messages and large convergence time in case of frequent network changes. [29] In the SDN based approach, these functions can be performed by the controller’s software modules, which reduce network devices’ functions to simple data forwarding.
* **Enables better monitoring and diagnostics –** a network abstraction is created, which offers an overall view of the network performance and allows dynamic reconfiguration of network devices in order to meet changing demand or user requirements. In the case of MPLS this can allow the creation of the LSP mesh and the optimization of the LSPs to be implemented as software modules in the controller, which runs in the background without interfering with the network operation.
* **Allows simplified and quicker implementation of innovations** - as the network control is centralized and there is no need every individual device to be reconfigured. This can enable, for example, experimentation or deployment of new routing protocols as software modules, using the legacy equipment. This could greatly reduce the cost of implementation of innovations.
* **Enables better use of the deployed resources and reduces OPEX** – in this project a solution that dynamically allocates required resources with respect to the current traffic demand is suggested. It can be implemented as controller’s software module that uses TE techniques to steer traffic off the least utilized network elements and then switches them off. This results is more efficient resource utilization and reduction of OPEX by energy savings.

## MPLS TE and MPLS VPNs with OpenFlow

The traffic engineering scenario in a typical MPLS network brings multiple advantages and notable improvement in the bandwidth management and utilization. The implementation of an efficient traffic engineered network can prove to be quite complicated, as the configuration of multiple protocols (such as OSPF, LDP, RSVP TE and possibly iBGP and MP BGP) might be required. Furthermore, frequent changes in the network can result in excessive traffic of update and control messages and large convergence time for the above-mentioned protocols.

An alternative Software Defined Networking (SDN) solution is described in [28] and [29]. The described solution retains the MPLS data plane, but implements the control plane using SDN Network Operating System (NOS) and set of software modules that perform the various network control functions. The architecture is presented on figure 3.4.

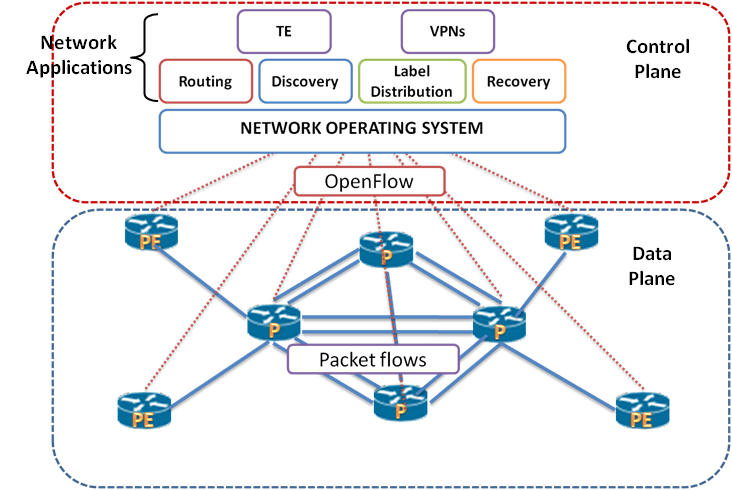


Fig. 3.4 Basic Scheme of SDN MPLS network, [29]

In this architecture the Network Operating System (NOS) provides an abstraction of the underlying network. The various network applications (such as routing and discovery, label distribution, TE and VPNs, recovery mechanisms) are implemented as software modules that operate on the virtual overlay of the network. The authors claim that implementation of these modules is relatively simple and the main benefit is the improved maintenance and update process of the applications. Furthermore, no distributed protocol is needed as the NOS has complete knowledge of the system. In the end the forwarding decisions are communicated to the network devices (OpenFlow switches) in the form of flow tables via the OpenFlow protocol.

## Hybrid MPLS SDN Network

A different approach that focuses more on improving the existing network architectures, rather than suggesting a completely novel approach, is described in [44]. The advantage of this approach is that it combines the effective switching capabilities, already existing within the MPLS core, and the flexibility and superior control of SDN. The architecture is presented on following figure 3.5:

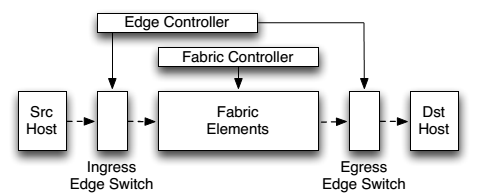


Fig. 3.5 SDN architecture that decouples the edge control plane and the fabric control plane, [44]

The architecture introduces a new element in the SDN paradigm, called “fabric controller”. The idea is the host-network relationship and complicated network services to be provided by the edge controller, while the fabric controller communicates simple packet transport functions. Thus, more efficient packet switching within the core may be achieved.

## Current project approach

The approach used in this project is based on the model described in [44]. The centralized controller communicates with the edge devices, while the core forwarding functions remain the same as in traditional MPLS network. The goal is to develop a solution that can be more easily implemented in existing network infrastructures, while offering distinct SDN advantages, such as reduced network complexity, centralized control and efficient utilization of network resources. A general scheme of the solution is presented on figure 3.6. The solution allows dynamic configuration of RSVP TE LSPs, based on specified time or traffic variations. Essentially, the controller monitors the traffic, performs CSPF calculations and controls the edge routers, which perform the RSVP signaling accurately at the required time. This allows the necessary resources to be reserved or released when needed. The traffic can be then steered off the underutilized links or nodes, which in turn can be safely switched off.

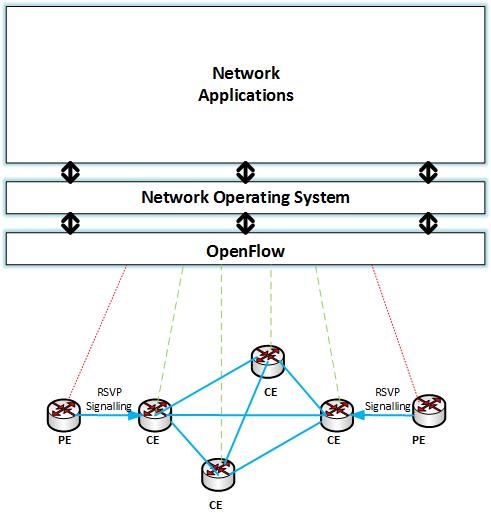


Fig. 3.6 Hybrid SDN MPLS architecture

**Chapter 4**

**Core Network Planning and Traffic Engineering**

The ever increasing demand for high-speed data services triggers continuous strive for innovations, service expansion and lowering of the prices in the telecommunications market. In this highly competitive market environment reliability may prove to be a key differentiator between the numerous service providers. [38] As a result Service Level Agreement (SLA) Assurance becomes a primary concern. According to [37], the primary problem of SLA Assurance is “to ensure there is sufficient capacity, relative to the actual offered traffic load”. Three main techniques that target effective service provisioning and SLA assurance can be distinguished:

* Capacity planning – also essential for the effective implementation of QoS/CoS and TE techniques
* QoS/CoS mechanisms – DiffServ, IntServ
* Traffic Engineering



## Core network capacity planning overview

According to [11], core networks are characterized with high bandwidth links and highly aggregated traffic. As a result the SLA requirements (delay, jitter, packet loss and availability) can be translated into bandwidth requirement, i.e. SLA parameters can be ensured if enough bandwidth is provided. An effective capacity planning methodology was described in [11] and [37] and will be summarized here.

The fundamental requirements for effective capacity planning are to implement accurate technique for measuring the current network load and to determine how much bandwidth has to be provisioned relative to the network load. The methodology can be applied with respect to the aggregate traffic or on per-class basis if DiffServ is utilized. A scheme of the proposed solution is presented on fig. 4.1 and the steps it follows are summarized:

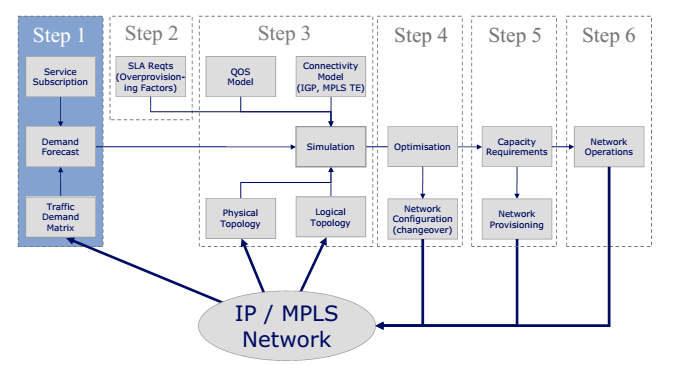


Fig. 4.1 Network planning methodology, [37]

1. Estimate the traffic demand matrix and account for future traffic growth.
2. In order to ensure fulfillment of the SLA requirements, estimate the appropriate overprovisioning factor.
3. Perform network simulations, taking into account the forecasted demand, the calculated overprovisioning and some possible failure cases in order to verify the desired network functionality and identify if there is a need for network upgrade.
4. Utilize network optimization techniques, such as network engineering and traffic engineering and compare different TE approaches. Optimization decisions include:
   * Which TE approach to be chosen – IGP TE or MPLS TE
   * How often to re-optimize
   * In case of MPLS TE:
     + Choose between core mesh or edge mesh
     + Determine appropriate tunnel sizing
     + Choose statically or dynamically established tunnels
5. Provide the estimated network capacity.
6. Verify efficient network operation by monitoring performance parameters.

## Traffic demand matrices

In [40], a traffic demand matrix is defined as “the amount of data transmitted between every pair of ingress and egress points in a network”. Traffic matrices give an overview of the current network state, the end-to-end traffic demands and are the fundamental input in the network planning process. Traffic matrices represent estimation of the aggregated traffic on the basis of IP prefix, router, point of presence (POP) or autonomous system. They can be broadly classified in two main groups [41]:

* **Internal traffic matrices** – calculated within the provider operated network devices. They can focus on any portion of the network. For example, they can examine the traffic only in the network core or can provide a broader overview, including the edge routers. They provide enough information to deal with changes within the network core.
* **External traffic matrices** – represent AS-to-AS traffic demands and therefore include external sources and destinations of traffic. They can be used, for example, to determine the impact of shifting between various Border Gateway Protocol (BGP) peering locations or failure of an entire border router.

The traffic demand matrices give an accurate representation of the traffic distribution and are used to predict the network behavior when changes occur. The usage scenarios are summarized in [37] and listed below:

* **Capacity planning** – traffic demand matrices information is the major input in the capacity planning process. They identify the available capacity as well as the need for additional bandwidth and can be calculated on per CoS basis. Together with information about the routing model, they can be used to predict the impact of demand growth or changes of traffic patterns, such as relocation of BGP peering or event-based traffic bursts.
* **Changes in topology analysis** – determines the network performance in case of failure, IGP metric changes, MPLS LSP shifts, addition of new elements or removal of old elements
* **Network Optimization** – finds bottlenecks or estimates routing mechanism and TE techniques

## IP traffic matrix estimation techniques

Various methods have been developed to accurately estimate traffic matrices. In order to derive a traffic matrix, information about the traffic patterns has to be collected. The approaches for traffic matrix construction can broadly be classified in two main groups:

* **Direct measurements** – the traffic matrix is constructed by mapping the traffic measurements to individual demands. The statistics that can be collected include:
  + **IP flow statistics** – a standard protocols for exporting IP flow statistics are IP Flow Information eXport (IPFIX) protocol and NetFlow.
  + **MPLS LSP accounting –** when MPLS is deployed the LSPs essentially represent aggregated traffic. In this case LDP or RSVP TE statistics can be used to construct the traffic demand matrix.

Building traffic demand matrices using direct measurement techniques, however, may often produce inaccurate results. The reason for this is that measurements can produce unreliable data (e.g. in case of RSVP counter resets or NetFlow cache overflow), inconsistent data (resulting from timescale differences with link statistics) or can miss important data (e.g. in case LSPs do no cover the traffic to BGP peers). Furthermore, directs measurement might result in high overhead. Traffic statistics might accurately represent point-to-point traffic, but not end-to-end traffic.

* **Estimation** – estimation techniques try to overcome the disadvantages of the direct measurement techniques and to reduce the resulting overhead. The principle of estimation methods is to try to match an estimation of the traffic matrix to a set of measurements extracted from the network. Some of the proposed traffic matrix estimation techniques are reviewed in [40].
  + **Gravity models** – they are based on the assumption that the amount of data that node *n* send to node *m* is proportional to the total traffic that exits at node *m.*  This model, however, does not use any information about the interior network links and the estimates are not consistent with the link load measurements. As a result the gravity model is usually used in combination with statistical data that takes into consideration the measured link loads.
  + **Tomogravity model** – an improved method based on combination of the gravity model and network tomography methods is described in [42]. It provides solution for large-scale IP networks that is claimed to be practical and fast. The methodology is based on the link load measurements, obtained by SNMP, and topology and routing configuration information in order to reduce computational complexity. The tomogravity model consists of two main steps:
    1. Initial solution is obtained by the edge link load statistics and gravity model solution.
    2. A set of possible network tomography solutions is obtained and the solution closest to the gravity model solution is chosen by applying quadratic programming. In order to reduce the computational complexity the network routing and topology configuration is used.
  + **Demand deduction estimation –** this a technique developed by Cisco that targets large complex networks [41]. The technique is based on interface traffic measures and estimates the demand traffic between two edge points. It determines the traffic ingress and egress points and how the data actually traverses the network. The ingress and egress points are paired into individual demand in order to create a “demand mesh”. Once this is done, a system of simple equations is constructed and solved in order to determine the demand traffic. Knowledge about the network topology is required. The process is illustrated on the following figure:

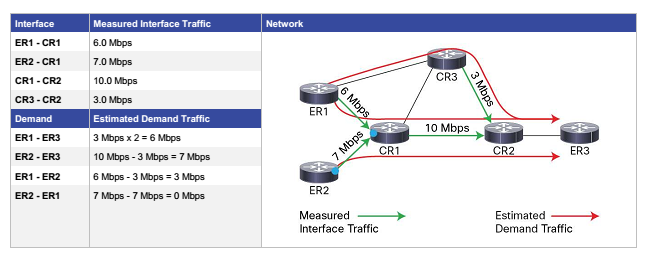


Fig. 4.2 Demand deduction estimation technique, [41]

Once the primary solution set is constructed, the result is refined by applying successive regression techniques based on additional measurements. For example, the primary solution set can be matched with LSP traffic measurements in order to narrow down the result.

## Statistics retrieval and usage

There are various commercially available tools that can perform network analyses and effectively calculate traffic demand matrices, using various estimation techniques. One possible example is the Cisco MATE Collector [11], which uses the aforementioned demand deduction techniques.

Regardless of the technique of choice, an important decision is how often the statistics will be collected. It will typically be in the form of packet or byte counters and can be aggregated over a chosen period to determine the average traffic demands. The sampling interval, however, has to be chosen carefully. If it is too large important traffic variations can be missed. If it is too small on the other hand, it can produce a great load on the system. According to [11], the common sampling intervals in backbone networks are 5, 10 or 15 minutes, depending on the size of the network.

## Overprovisioning factor estimation

The calculated traffic demands represent averages of the real traffic over the sampling interval and therefore do not take into account the variations within the interval. If there are large traffic bursts, they can cause SLA violations if the link bandwidth is not carefully planned. To eliminate this problem, the actual bandwidth is overprovisioned with respect to the measured average load. The amount of overprovisioning is defined as the overprovisioning factor (OP) [11].

The OP depends on the link speed and the arrival traffic distribution on the link. There are various theories concerning arrival distribution in IP networks. One possible assumption is that the traffic is self-similar or in other words the average rate is the same regardless of the time period. Another assumption claims that the traffic follows a Poisson distribution, which results in smaller variations in the average rate if a longer measurement period is chosen. The overall impact of these assumptions on the OP factor estimation is that for high-speed links the required overprovisioning is much greater for self-similar traffic.

The methodology, described in [11], and summarized here is based on the assumption that the “internet traffic is not self-similar at timescales relevant to QoS” [39]. This assumption is based on series of empirical measurements for different interface speeds and different mixes of traffic, which determine the resulting queuing delay. The simulation results are presented on the following figure and show that the required overprovisioning decreases when the link bandwidth increases, which is typical of Poisson traffic.

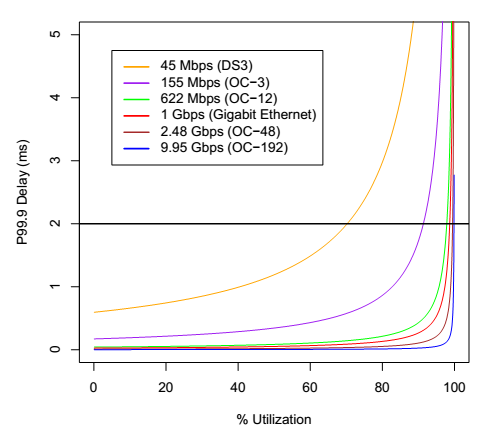


Fig. 4.3 Simulation results, [39]

The required overprovisioning factor for achieving particular delay SLA requirement can be roughly estimated from Figure 3, depending on the link bandwidth. The figure gives an estimate of the per link delay only. As the traffic will eventually pass multiple links, it will experience queuing delay multiple times. In [39] it was determined that the delay over multiple hops corresponds to the single hop delay multiplied with an appropriate “multiplication factor” (Table 1).

Table 4.1 Delay multiplication factors, [39]

|  |  |
| --- | --- |
| **Number of hops** | **Delay multiplication factor** |
| 1 | 1.0 |
| 2 | 1.7 |
| 3 | 1.9 |
| 4 | 2.2 |
| 5 | 2.5 |
| 6 | 2.8 |
| 7 | 3.0 |
| 8 | 3.3 |

To summarize, the required per link delay can be obtained by dividing the SLA delay requirement across the core with the appropriate multiplication factor. The OP can then be determined from figure 4.3.

**Chapter 5**

**Energy Efficiency Mechanisms**

The energy consumption of the network elements constitutes a considerable part of the network operation costs and current communications network designs are notably energy inefficient for two main reasons:

1. The network infrastructure is consistently overprovisioned in order to meet high traffic demands and occasional traffic bursts, but does not consider energy-saving mode during low network loads periods.
2. The network elements are always powered on, regardless of their utilization, in order to maintain network connectivity.

Various energy-saving models, considering both future internet architectures and improving current network design, are presented in [45]. Some of the discussed approaches are summarized in the current chapter and compared to the methodology proposed in the current project.



## Switching off network elements

Several models that try to achieve energy savings by turning off a subset of network elements have been proposed. Critical for their operation is to assess the importance of the network elements based on their utilization or criticality and to perform routing or traffic engineering decisions accordingly. The switching off process should be performed in such a way so that the impact of the network performance is minimized. Three main points that should guide the design process are identified in [45]:

* Aggregation of the traffic to the most frequently used links and steering it off the elements that will be switched off
* Identification of the particular elements that will be switched off in order to minimize the impact on the network performance and maximize the resulting reduction of energy consumption
* Efficient network monitoring and management to turn the elements back on when needed

## Energy-Saving Algorithm

A specific algorithm that aims at providing an effective switch off process and minimization of the negative impact on network performance is proposed in [45]. The process is divided into three main steps:

1. Traffic aggregation – every link has a certain weight parameter. The algorithm analyzes the utilization of every link and compares it to a threshold. If the value is above the threshold, its weight will be increased which will result in a decrease of its utilization. If the value is below the threshold, the link will become a candidate for weight decrease. From all the candidates, only the links with higher utilization will endure a weight decrease. Thus, the traffic will be aggregated to these links. The links that are not being used will become candidates for switch off and will be transferred as an input for the second function.
2. Ranking the network elements – a certain ranking is assigned to every candidate for turning off. The ranking identifies the least important elements that have to be switched off first. The ranking is based on so called local centrality measure [46]. This method estimates the node significance by calculating the number of nearest and next nearest neighbours. The elements are then ranked based on local centrality measure and the history of utilization.
3. Turning off network elements – lowest rank elements will enter a sleep mode, but only if there is no traffic left on any of its links and the network remains fully connected. The sleeping period will be of fixed length and after that the node will enter a “pre-wake-up” state in order to evaluate if packets are destined to it or if the network performance has degraded significantly. If neither of the above is true, the node will re-enter the sleep mode. In case of degraded network performance, the highest ranking nodes will be consecutively switched on until the desired operation is restored.

## Dynamic link metric

An alternative approach for switching off network links is presented in [13]. It is based on an algorithm that aggregates the traffic to the most used links and then turns off the links that remain unused. The method exploits the OSPF routing protocol principle, based on link weights. The aggregation is performed by dynamically changing the weights of the links, causing the traffic to be routed through the most used elements. In order to avoid congestion, a threshold on the link load is set, so that it limits the maximum amount of traffic the link can carry. The method is illustrated by a flow diagram, presented on figure 5.1.

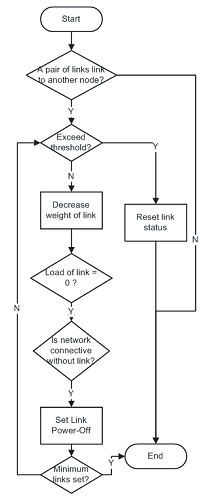


Fig. 5.1 Dynamic link metric flow diagram, [13]

The algorithm performs the following actions:

1. The set of parallel links between the nodes is identified and the load of each link is compared to the predefined threshold. It the load is below the threshold the link becomes a candidate for weight modification.
2. The link load of the candidates is compared and the link with higher load is chosen. Its weight is then decreased.
3. The modified link is excluded and the status of the rest of the links is examined.
4. The routing paths are recalculated, according to the new weights.
5. The links with no traffic are identified. If the network remains connected without them, the links are switched off.
6. The link load is compared to the threshold. It is below the maximum allowed value, the algorithm check whether the number of operating nodes is above the minimum number of node that guarantee network connectivity. If this is the case, the algorithm returns to step two.
7. If the load is above the threshold, the previous link weight is reapplied or the network link is powered back on.

## Switching off network elements

In [12] and [47] the concept of switching off both network nodes and links is approached in a different manner. The paper defines the problem as Integer Linear Programming (ILP) formulation that aims to find the minimum set of nodes and links that must be powered on so that the power consumption of the network is minimized. Knowledge about the topology and the link capacity, the maximum supported link utilization and the power consumption of the network elements is assumed. Two constraints are defined – flow conservation and maximum link utilization. The problem, however, is defined as NP-hard and a heuristic algorithm that offers a solution that can scale up to large networks is suggested.

The proposed heuristic algorithm selectively switches off network elements, re-computes the shortest path and verifies the network connectivity and QoS parameters. The algorithm targets wide area networks in which the resource overprovisioning is usually large. A diagram of the algorithm is presented on figure 5.2.

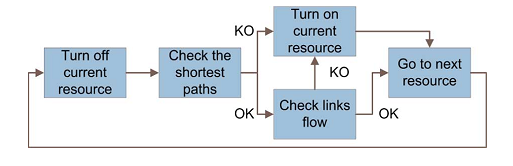


Fig. 5.2 Turning off algorithm, [47]

As a starting point it is assumed that all network elements are powered on. The algorithm iteratively switches off network elements, calculates the shortest path and checks the preset utilization constraints. If the utilization level remains below the threshold, the element is permanently switched off. Two different kinds of heuristics are then proposed – “node-oriented” and “link-oriented”. The nodes are examined first, as it is more difficult to turn off an entire node. The links are checked on a second stage. Different heuristic approaches for sorting the network elements are examined. The nodes are sorted either randomly, according to the number of links that are connected to each node (least-link), according to the amount of information flowing through each node (least-flow), or according to their power consumption level (most power). The links are sorted either randomly, according to the least-flow heuristic, or according to the most-power heuristic.

The paper examines two main scenarios:

* Test scenario on a simplified version of realistic topology that consists of four levels of nodes – core, backbone, metro and access.
* Examination of a number of randomly generated hierarchical topologies that consist of core, edge and aggregation nodes.

The traffic patterns are also randomly generated, assuming that only access nodes are sources and sinks. An overprovisioning factor of 50% is assumed for the real topology case, thus limiting the maximum allowed link utilization to 50%. For the randomly generated topologies the overprovisioning factor was varied.

In the randomly generated topologies the different heuristics were examined and compared to optimal solutions obtained by the ILP formulation (simplified by a number of assumptions. The results show much higher achievable energy savings (close to 70%), but with the cost of huge computational time (close to 105 s). The heuristic approaches produces results comparable with each other (close to 50%), but with computational time of about 5s.

In the evaluation of the realistic scenario the daily traffic variations are taken into account. The capacity of the links is chosen regarding the peak traffic demand. The night traffic is evaluated as 38% of the peak traffic. The maximum achievable power saving during the night is measured to be 36% using the optimal solution, and almost the same (35%) using most power – most power heuristic. The least flow-least flow (LF-LF) heuristics achieved maximum of 23% power saving. The lower result of the LF-LF heuristic, however, stems from the fact that it tries to power off first the lower capacity access nodes, which also consume less power. Thus, it leaves the high power consuming backbone and access nodes powered on.

## Green game

An approach for ranking network nodes according to their criticality in the network is presented in [14]. The approach is based on both the topology and the traffic matrix knowledge and utilizes the Shapley value as a criticality index. The Shapley value indicates how important is the element for the network operation and how much its absence would affect the network performance. The evaluation is based on a real network topology and traffic matrix and claims to achieve a good trade-off between energy savings and network performance.

The computation of the Shapley value, however, is evaluated and computationally complex. To overcome this limitation a Green-Game approach is proposed. It essentially identifies a subset of nodes that can effectively transport the required amount of traffic. To simplify the computational process a heuristic approach that eliminates the loop and augmented paths from the computational process and defines a maximum path length is suggested.

The numerical results, presented in [14], indicate that the Green-Game approach implementation can result in high-energy savings with little or no impact on the QoS parameters. Its computational complexity, however, can be considered a drawback when applying to real-life implementation.

## Summary and comparison

Several different approaches that aim to achieve energy efficiency through resource consolidation and switching off network elements have been described in the previous sections.

In section 5.3.2 dynamic link metric technique for traffic aggregation based on dynamically changing the link weights and rerouting the traffic to the most utilized links has been described. The algorithm suggest simple alternation of the OSPF routing protocol, but targets only links and not entire nodes. Furthermore, the links are chosen based on utilization parameters only, which might lead to suboptimal ranking of the network elements.

Another algorithm for traffic aggregation based on changing the weight parameters has been described in section 5.3.1. The approach targets both links and nodes and utilizes an optimized technique for element ranking, based on both utilization and local centrality parameters. The lower ranked network elements are then put into sleep mode, but they are assumed to keep their place in the network by periodic waking up. This can be considered as both an advantage and a disadvantage. The advantage is that the technique theoretically allows every underutilized network node (core, edge, or access) to be put into sleep mode, as packet can still be destined to an idle node. The disadvantage is that it offers suboptimal energy efficiency, because of the periodic waking up requirement.

An approach that focusses on accurate network elements ranking has been described in section 5.3.4. The approach is based on both the topology and the traffic matrix knowledge and utilizes the Shapley value as a criticality index. The algorithm performs accurate node ranking, but does not focus on the links, as they consider the achievable energy savings by switching off links negligible. After the selection of the optimal subset of nodes that will remain powered off, routing weights optimization and equal cost multipath traffic distribution is utilized. The approach offers a concise analysis on the achievable energy savings and the impact on QoS parameters, but its computational complexity may prove to be a draw back in real-life implementation. The paper does not consider network control plane mechanism for implementation.

An approach for switching off both network nodes and links has been described in section 5.3.3. The paper defines an Integer Linear Programming (ILP) formulation of the problem and proposes greedy heuristics for iterative switching off of network nodes and links. An evaluation on the achievable energy savings on both real and randomly generated topology is presented. The paper, however, does not consider the impact on the QoS parameters and does not offer network control plane mechanism for implementation.

The main purpose of this project is to offer a practical solution for resource consolidation and switching off network elements. The proposed architecture is based on centralized SDN controller, which allows modular implementation of different network functionalities in the form of software applications. This allows great flexibility in the choice of traffic matrix estimation, network elements ranking, routing and traffic engineering techniques. Furthermore, the project offers an algorithm for resource consolidation, based MPLS traffic engineering techniques that aims to offer real-life implementation solution. The project estimates the impact of the QoS constraints on the achievable energy savings.

**Chapter 6**

**Proposed Solution**

The current project proposes an energy saving methodology for resource consolidation and turning off network elements in MPLS networks. The methodology focuses on core networks, where the resource overprovisioning is quite high. It is based on the observation that the network utilization experiences considerable variations, based on time of the day, time of the week, etc. For example the high traffic demand in the peak hours of the day is opposed to the much lower demand during the night [7]. During the low demand time intervals, it is then theoretically possible to concentrate the traffic in a small subset of nodes or links, which will allow the unused nodes/links to be switched off. The current project describes an SDN-based scheme for resource consolidation, proposes an algorithm for traffic rerouting and turning off network elements in MPLS-based networks and evaluates the algorithm’s performance in terms of the achieved energy conservation in accordance with predefined connectivity and quality of service constraints.



## Proposed architecture

A general scheme of the proposed architecture is presented on figure 6.1.

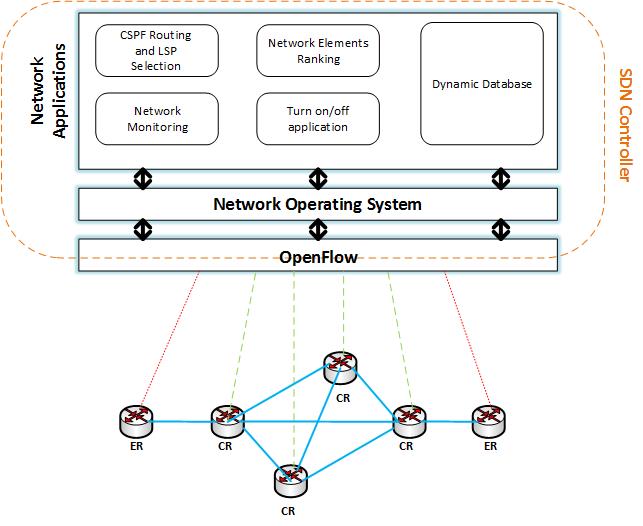


Fig. 6.1 Software Defined Networking Controller Scheme

The proposed approach is based on Software Defined Networking scheme with reconfigurable centralized controller. The scheme represents a hybrid SDN MPLS architecture, described in section 3.5.3. Controller’s functionalities will be implemented as network applications on top of the Network Operating System (NOS) and will communicate with the underlying network elements via OpenFlow protocol. The controller should perform the following actions:

* **Monitor the network traffic**
* **Rank network nodes** - based on traffic demand matrix estimation
* **Create and manage a dynamic database** – the database identifies the network elements that can be switched off and is based on:
  + Statistics for customer consumption – the utilization of the network resources varies depending on the time of the day, time of the week, etc.
  + Real-time network statistics – critical parameters should be monitored in order to react to unexpected traffic demand
* **Perform Constrained Shortest Path First (CSPF) routing decisions**
* **Control the MPLS edge routers** – which in turn perform the RSVP signaling accurately at the required time
* **Turn on and off the appropriate network elements**

## General overview of the switch off/on process

The main purpose of the centralized controller is to coordinate the switch off/on process and react immediately to changes in the customer demands. The process of turning on and off network nodes/port should meet certain connectivity and QoS constraints. The final goal is to minimize the total power consumption of a large core network, in which the resource overprovisioning is quite big.

According to [11] core networks are characterized with high bandwidth links and highly aggregated traffic. As a result the SLA requirements can be translated into bandwidth requirement, i.e. SLA parameters can be ensured if enough bandwidth is provided. The network utilization, however, experiences considerable variations, based on time of the day, time of the week, etc. For example the high traffic demand in the peak hours of the day is opposed to the much lower demand during the night [7]. During the low demand time intervals, it is then theoretically possible to concentrate the traffic in a small subset of nodes or links, which will allow the unused nodes/links to be switched off.

A general flow diagram of the switch off/on process is presented on figure 6.2.

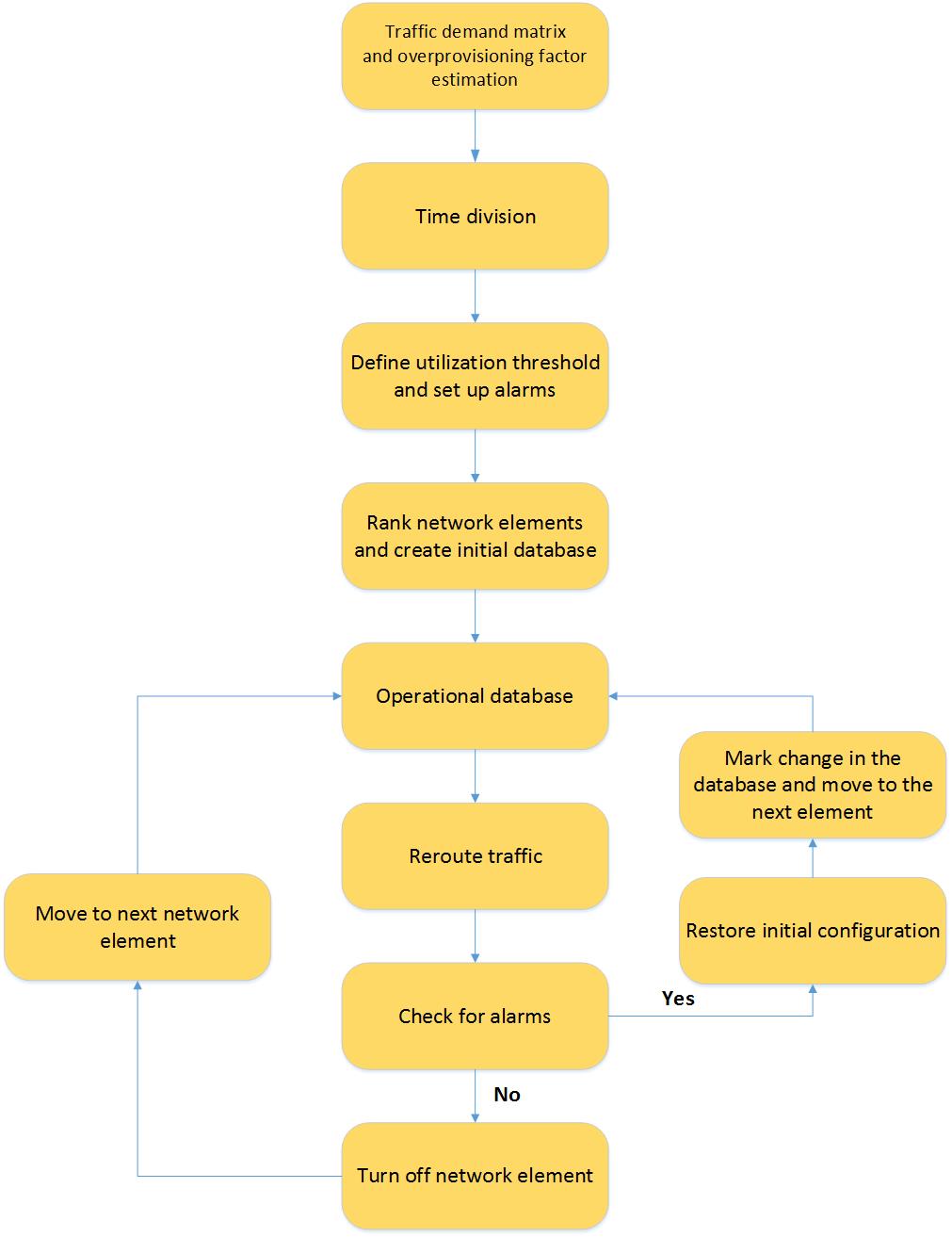


Fig. 6.2 Flow diagram of the switch off/on process

The switch on/off process should go through the following steps:

1. **Estimate traffic demand matrix** – in order to estimate the bandwidth requirements for every time period, an internal traffic demand matrix is calculated. A basic overview of traffic demand matrix calculation techniques was presented in section 4.3. One possible example is calculation, based on MPLS LSP traffic accounting statistics [11]. For more accurate estimation there are various commercial tools that can be used. A good example is Cisco MATE Collector that uses demand deduction technique [41], as it is suggested to be particularly accurate for predicting the overall utilization after a failure, a topology change, or a metric change. From the resulting traffic demand matrix the appropriate overprovisioning factor can be determined.
2. **Determine the overprovisioning factor** – this describes the amount of bandwidth that has to be overprovisioned in order for the SLA requirements for delay, jitter and loss to be met in case of unexpected traffic bursts. The process of determining the overprovisioning factor is described in [11] and summarized in section 4.3.
3. **Time division** - the time periods of the day/week that are characterized with significant variations in traffic demand are determined. This can be determined on the basis of the obtained traffic demand matrix or manually determined by the network operator.
4. **Threshold and alarms** – based on the traffic demand matrix and the overprovisioning factor the SLA requirements for each time period can be translated into bandwidth requirements. The maximum link utilization threshold can therefore be calculated. Based on the results alarms can be configured by the network monitoring system. In case the link utilization exceeds the predetermined threshold for a certain period of time, the nonoperational network elements will be turned back on.
5. **Node/link ranking** – based on the data from the traffic demand matrix the network elements can be ranked according to different parameters. The nodes can be classified according to parameters such as number of links, CPU and memory utilization. The link can be classified according to interface metrics such as throughput, average utilization, etc. Different network elements classification techniques are presented in Chapter 5. The process is performed by the application module, implemented on top of the NOS. The ranking can be based on so called local centrality measure [46]. The method estimates the node significance by calculating the number of nearest and next nearest neighbors. The elements are then ranked based on local centrality measure and the history of utilization.
6. **Initial database** – based on the node ranking results the order in which the network elements will be switched off is determined. A database containing information about the node and link priority is then created. Different tables will be created for every time period. Once more the purpose of the database is to mark the order in which the element will be switched off and will be used as a base for future operation in order to speed up the process. Nodes and links with higher priority will be switched off first. An example is presented in table 1.

Table 6.1 Node ranking database

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time period** | **Ranking** | | | |
| **Priority** | **Node** | **Priority** | **Port** |
| 23h-06h | **1** | Node 3 | **1** | *Eth 2* |
| **2** | *Eth 0* |
| **3** | *Eth 1* |
| **2** | Node 1 | **1** | *Eth 0* |
| **2** | *Eth 1* |
| **3** | … | … | *…* |
| 06h-09h | **1** | Node 2 | **1** | *Eth 1* |
| **2** | *Eth 0* |
| **2** | Node 1 | **1** | *Eth 2* |
| **2** | *Eth 0* |
| **3** | *Eth 1* |

1. **Operational database** - a second instance of the database is created. This instance will be dynamically changed during the switching off process in order to store any resulting changes in the order in which the element have been switched off. The changes are expected to occur after the traffic from a switched off element is rerouted.
2. **Reroute traffic off the least utilized links** - in order to avoid packet loss, the traffic is rerouted before the links/nodes can be switched off. Traffic engineering (TE) techniques are utilized. In case of MPLS based core networks this project suggests resource reservation protocol (RSVP) with extension for traffic engineering to be used. Once the traffic is rerouted, the links/nodes can be switched off. The process of traffic rerouting will be explained in details in the following chapter. Essentially, alternative label switching paths (LSP) will be established, excluding the node that has been marked for switching off.
3. **Check for alarms –** the new LSP will be established by the edge routers and signaled via RSVP. At this point the traffic will be rerouted through the new path. The path will be chosen by CSPF module of the controller and is supposed to have enough available bandwidth. However, the utilization threshold might still be exceeded. If this is the case the original path will be reestablished and the switch off process will move to the next network element. The change will be recorded in the operational database.
4. **Switching off –** in case there are no alarms indicating excessive link utilization, the switch off process will move to the next element in the operational database.

## Rerouting algorithm representation

The proposed algorithm’s main function is to steer traffic off the network node that has been marked for switching off in order to prevent packet loss and to enable safe disconnection procedure. The algorithm is implemented within the “CSPF Routing and LSP selection” module, which communicates with the MPLS edge router (ER) via the OpenFlow protocol. The algorithm takes as an input the ordered set of IP addresses, produced by the CSPF routing, which defines the new LSP. The node under consideration is determined by the corresponding entry in the operational database. A flow diagram of the procedure is presented on figure 6.3.

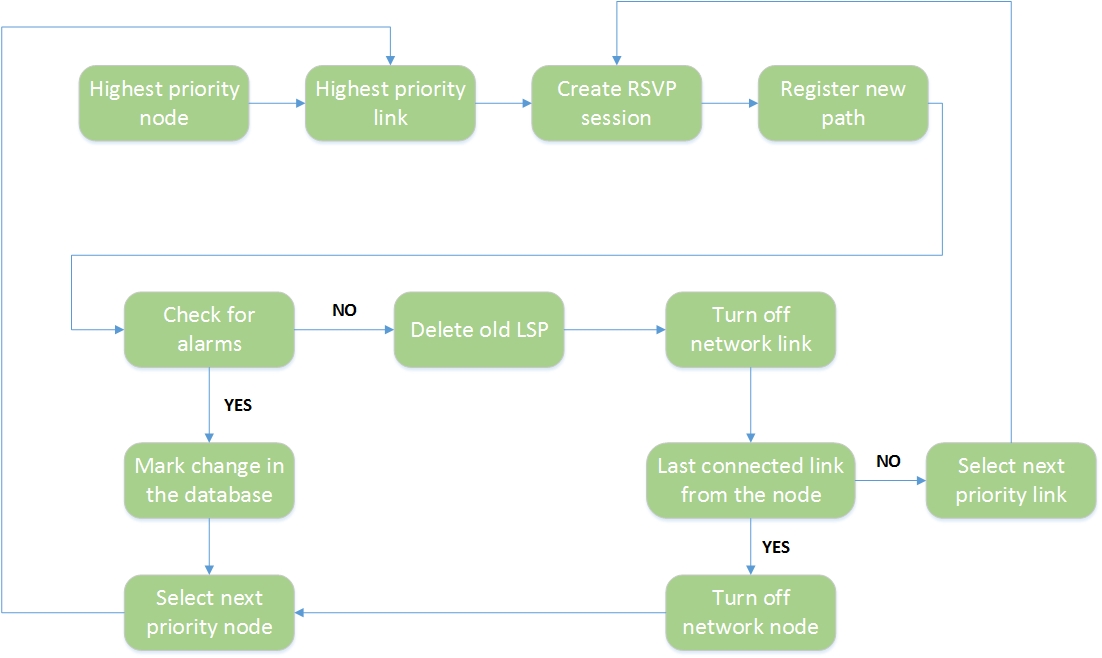


Fig. 6.3 Re-routing algorithm flow diagram

The algorithm follows the following steps:

1. **Select highest priority node –** the node determined by the “Network elements ranking” application module as the least utilized node and is indicated by the corresponding entry in the Operational Database.
2. **Select highest priority link –** the least utilized link connected to the selected node.
3. **Create RSVP sessions –** the LSPs that pass through the selected link are identified and alternative LSPs are calculated for each path, using the implemented CSPF algorithm. The selected node is excluded from the calculation by setting the advertised bandwidth to minimum. The alternative LSPs are identified as ordered sets of IP addresses and an RSVP-TE session is initiated by each corresponding ER.
4. **Register paths -** the new paths are registered in the routers’ FEC tables. After the RSVP path reservation messages are propagated, the traffic is forwarded through the new paths.
5. **Check for alarms –** it is checked if the utilization requirements are fulfilled after the traffic rerouting. In case the utilization threshold is exceeded the old path is reestablished and the next priority node is selected from the corresponding database entry. If no alarms are detected for a defined period of time, the old LSP is deleted and the link is switched off.
6. **Last connected link –** the next step is to check if there are more active ports on the selected node. If there are any, the next priority link is selected from the corresponding entry of the operational database.
7. **Turn off node –** if there are no more active ports on the selected node, the node is switched off and the process moves to the next priority node.
8. **End –** the process ends when no alternative LSPs are identified by the CSPF algorithm.

The algorithm will be evaluated in terms of the achieved energy conservation in accordance with predefined connectivity and quality of service constraints. The simulation tool that will be used is OMNET++. The procedure will be explained in details in Chapter 7.

**Chapter 7**

**Simulation Setup**

The project aims to develop a practical approach for energy saving by resource consolidation and turning off network elements, based on an SDN MPLS network architecture. An algorithm for resource consolidation and turning off network nodes, based on MPLS traffic engineering techniques, is proposed. The achieved energy conservation under predefined connectivity and quality of service constraints will be examined. The simulation tool that will be used is OMNET++ Integrated Development Environment (IDE) 4.3.1, with INET 2.3.0 open-source communication networks package. The algorithm’s operation will be implemented via an xml script with the use of the existing *Scenario Manager* block in OMNeT++.



## OMNeT++ 4.x Integrated Development Environment

OMNeT++ 4.x IDE is an Eclipse based platform with added functionality, editors and views. It offers the possibility for creating and configuring models (NED and ini files), performing batch executions, and analyzing simulation results. [1] The main components of the platform include:

* **NED Editor** –NED files can be edited in both graphical and source code editing mode. The graphical user interface allows the user to build the chosen topology by creating compound modules and submodules, channels and other component types. Visual and non-visual properties can be edited both from the Properties View and by direct source code commands. The text mode uses Eclipse C++ editing and offers context-aware completion of keywords and module types, syntax highlighting, automatic indentation and source code validation as the user is typing.
* **Ini File Editor** – allows configuration of the simulation model both in graphical and source code editing mode. The editor examines the NED declarations and can provide automatic generation of all NED parameters. The user will only have to provide the desired values. A Problem View that displays all the errors, warning and info messages is also offered.
* **Simulation Launcher –** the simulation can be directly executed form the IDE as normal C/C++, as standalone applications (under Tkenv or Cmdenv) or as batches of simulations where every runs differs in module parameters. In order to offer enhanced debugging and traceability opportunities. The Tkenv environment uses three methods: automatic animation, module output window and object inspector. Automatic animation method animates the flow of messages and reflects the state changes of the nodes. The module output window displays textual debugging or tracing information, generated by the individual modules or module groups. The object inspector displays the state or contents of every individual object.
* **Scave (Result Analysis) –** it is an OMNet++ tool, used to visualize simulation results saved into vector and scalar files. The tool allows the user to group the data into various datasets and perform various processing, filtering and charting operations. Various chart and graph types are available.
* **INET Framework –** INET is a simulation model suite for TCP/IP and Internet-related protocols for OMNeT++. In includes various protocol implementations (such as IPv4, IPv6, TCP, SCTP, UDP), several application models (such as MPLS with RSVP-TE and LDP signaling, link layers models – PPP, Ethernet, 802.11), support for mobile and wireless communications.

## Simulation Setup

The evaluation of the proposed approach is based on a test case, which considers a simplified version of real network topology of a national telecommunications service provider. The evaluation focuses on the MPLS based super core part of the provider’s network. The topology is implemented using OMNeT++ 4.3.1 IDE, equipped with INET 2.3.0 open-source communication networks package.

## Network topology

In order to obtain realistic and relevant test results, an actual network topology of a national telecommunications service provider is chosen. The network is built over a DWDM transmission network and follows the hierarchical design of four levels – super core, core edge, metro and access. The high performance super core consists of 8 sites in four major cities. The core edge layer consists of 40 sites, situated in major cities around the country that aggregate the traffic from the metro and access layer. The project focuses on the super core part of the topology for two key reasons:

* **Huge resource overprovisioning –** the core network planning takes into account the future rise of the traffic demand and ensures high redundancy of the links and nodes in order to prevent losses in case of failure.
* **High power consumption –** the high-performance network elements and long high-bandwidth links consume considerably larger amount of electrical power and temporal turning off such elements can ensure large power savings [7].

A simplified scheme of the network topology is presented on figure 7.1.

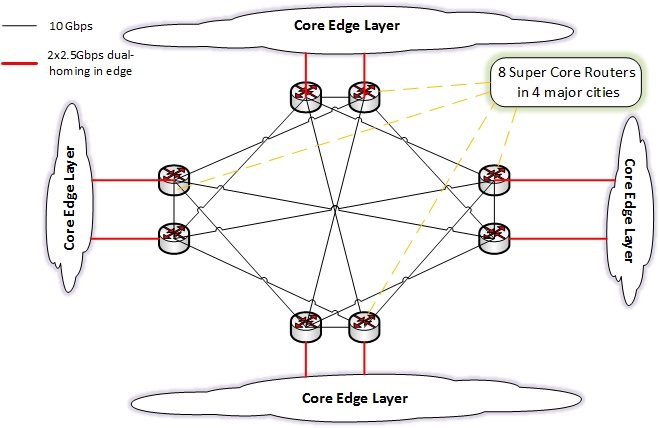
****

Fig. 7.1 Super core network topology

The super core layer consists of 8 sites in 4 major cities. A full mesh topology is constructed between the major cities and partial mesh between the super core sites, connected by 10Gbps trunk optical channels, implemented over a DWDM transmission network. The core edge layer is not examined in the current project and therefore is represented as a cloud on figure 1. The core edge routers are situated in different big cities and each of them is dual-homed to two independent super core sites. The connection channels are typically with capacity of 2.5Gbps or 1Gbps.

## Simulation Setup

The chosen network topology is implemented using the NED editor of OMNeT++ 4 IDE (figure 7.2).

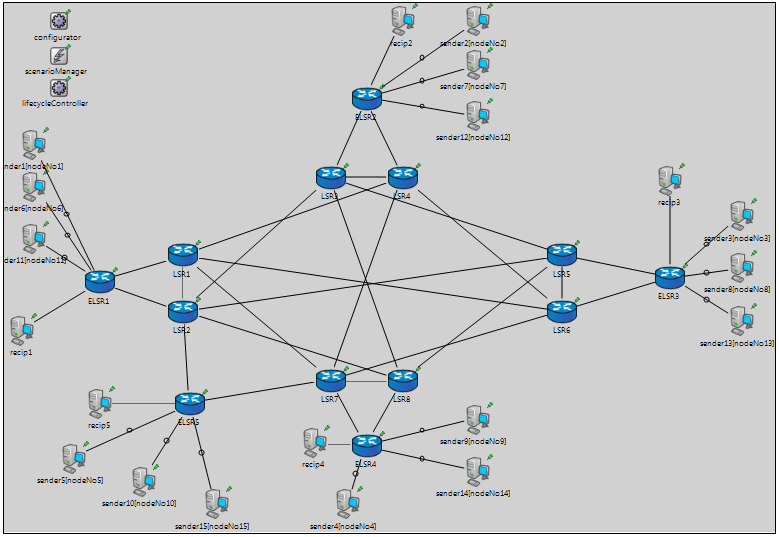


Fig. 7.2 OMNeT++ implementation of the super core network topology

The super core layer of the provider’s network is implemented using *Label Switching Router (LSR)* modules, provided by the INET framework. The channels between the nodes are characterized by two parameters – datarate and delay. As the edge core layer is not examined in the current project, only 5 edge core routers are used for simplicity. Each edge core router aggregates the outgoing traffic for a set of hosts that generate IP packets. The incoming traffic is destined for separate hosts, that act as traffic sinks. The edge routers communicate with the controller and are responsible for RSVP signaling. The controller’s operation is implemented via xml script with the use of the *Scenario Manager* module. The simulation consists of the following blocks that will be examined in details in the following sections:

* 8 *RSVP LSRs* - **LSRx** at figure 2
* 5 edge *RSVP LSRs* - **ELSRx**  at figure 2
* 5 Burst Hosts (sinks) – **recipx** at figure 2
* Varying number of Burst Hosts – **senderx**  at figure 2
* *IPv4NetworkConfigurator* – **configurator** at figure 2
* *ScenarioManager*
* *lifecycleController*
* connections

## RSVP-TE capable routers (LSRx and ELSRx)

The RSVP-TE capable router module instances represent either the super core MPLS switching routers that will be targeted by the algorithm for switching off or the MPLS edge routers. The super core routers’ main function is to perform fast packet switching by simple lookup of the packet labels. The information required by routers’ LIBs that identify the way the packets are forwarded is propagated via RSVP sessions initiated by the edge routers. The MPLS core router are key elements in the proposed architecture. They are responsible for initiating the RSVP sessions according to the instructions received by the SDN controller.

The RSVP-TE capable router module consists of various submodules presented on figure 7.3

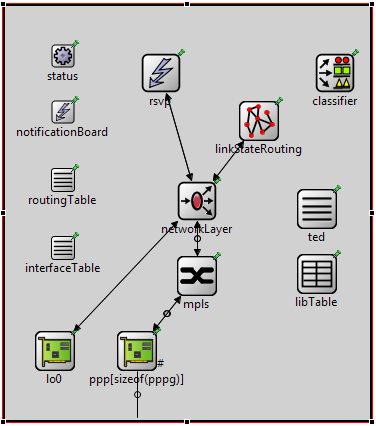


Fig. 7.3 RSVP-TE router module

The modules consist of standard submodules such as *IP routing table*, *interface table* and *network layer* submodule that provide interfaces to transport layer protocols - TCP, UDP, echo/ping, RSVP. In addition to this there are *MPLS* submodule, *RSVP* submodule, *linkStateRouting* submodule, *Traffic Engineering Database (TED)* and *LIB* tables, *and classifier* submodule. The *classifier* submodule provides an interface for static FEC table definitions via separate xml file. The *MPLS* submodule implements the MPLS protocol. It is responsible for label operations (PUSH, POP and SWAP) and packet forwarding. The *RSVP* submodule processes RSVP-TE messages, installs labels and does the reservation along LSP paths. It communicates with the *MPLS* submodule, *TED table* and can receive commands from the *ScenarioManager* modules. In the current project the RSVP sessions are determined by the SDN controller and distributed to the MPLS edge routers (ELSRx) by the *ScenarioManager* module. The *linkStateRouting* submodule implements simple link state routing protocol. It is responsible for distributing information about the current link usage. The *TED table* is filled by the implemented link state routing protocol and contains information about the network topology and about the total and reserved bandwidth of the links.

## Burst Hosts (recipx and senderx)

Burst hosts act like traffic sources (senderx) and traffic sinks (recipx). The traffic sources are responsible for the IP traffic generation and the traffic sinks represent aggregated traffic destination behind each edge router. The *BurstHost* module and the relevant submodules are presented on figure 7.4.

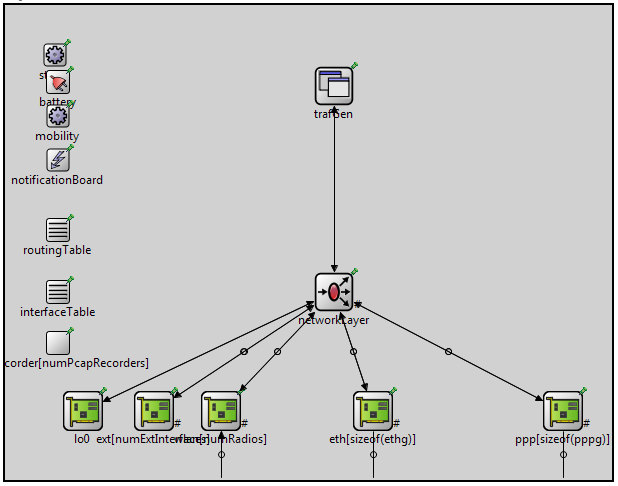


Fig. 7.4 Burst Host module

The module again contains standard submodules such as *IP routing table*, *interface table* and *network layer* submodule, similarly to the router module previously described. A Key element of the burst host is the *IIPvXTrafficGenerator (trafGen)* submodule. The traffic generator connects directly to IP (without using TCP or UDP) and generates data packets according to several user-defined parameters. This approach is appropriate for the current simulation, as here only the behavior of the super core network devices is examined, where the traffic is highly aggregated. The parameters that are specified by the user are start time, send interval, packet length, number of packets and destination address. Received packet bytes and end to end delay statistics are generated.

## IPv4NetworkConfigurator (configurator)

This module assigns IP addresses, configures static routes and performs route optimization. The module communicates with the network layer submodule and the interface tables of the other network elements. IP addresses are assigned to every element listed in the interface tables. The module allows both manual and automatic address assignment and supports manual and automatic routes.

## LifecycleController

This module is designed to control operations like shutdown/restart, suspend/resume, crash/recover for network nodes, interfaces and protocols. The operations are initiated via *ScenarioManager* script. In the current simulation this module implements the SDN controllers switch off/on commands.

## ScenarioManager

*ScenarioManager* is a key module as it implements the functionality of the SDN controller in the current simulation via an xml script. Generally, the *ScenarioManager* module is designed to control the simulation experiments. It allows the user to schedule certain events at certain times. The module executes either built up commands or dispatches commands that are to be executed by other simple modules. The commands are specified by the user in xml format. The module can schedule events such as changing a parameter value, changing the bit error rate of a connection, removing or adding connections, removing or adding routes in a routing table. In the current simulation the module defines certain RSVP sessions that signal the parameters of the new LSPs. The traffic is then rerouted through the newly established LSPs.

## Simulation Scenario

The proposed algorithm is evaluated in terms of achievable energy saving with respect to bandwidth utilization percentage. Packet loss, latency and utilization variations, resulting from the switching off procedure are examined. The experiments were realized on the test simulation setup, described in the previous chapter. Several simulation setup parameters should be taken into consideration:

1. **Traffic generators** – in order to simulate realistic traffic patterns in the *IIPvXTrafficGenerator (trafGen)* submodule, part of the *BurstHost* module (described in section 7.2.2.2) is chosen. It communicates directly with IP, (without the use of transport protocols such as TCP or UDP). The traffic patterns are shaped by defining start time, send interval, packet length, number of packets and destination address.

Three different groups of hosts are distinguished, based on the packet length parameter – small packets with uniformly distributed size between 40B and 45B, packets with uniformly distributed size between 1290B and 1310B and large packets uniformly distributed size between 1450B and 1500B. These packet sizes are identified in [10] as predominant in Internet traffic.

The amount of generated traffic is controlled by varying the send interval. The send interval is either fixed or exponentially distributed around a certain value. The *TrafGen* submodule parameters are defined in the *omnetpp.ini* file.

1. **Channel parameters** – the channels in the current simulation are characterized by two main parameters – datarate and delay. The datarate of the channels, connecting the super core nodes is fixed to 10Gbps. The datarate of the links, connecting the edge nodes to the core nodes is not limited in order to avoid congestion. The latency parameter for each node is calculated based on its real physical length (propagation delay), allowing 5% increase, based on routing and switching delay [11]. Single mode fibers (1550nm) are considered in the calculations.
2. **Edge routers configuration** – the edge routers are implemented via the *RSVP-TE capable router* module. These are the modules that communicate directly with the centralized controller and initiate the RSVP sessions in the specified time. They received information about the correct paths by the controller in the form of ordered set of IP addresses. In the current simulation FEC information and RSVP sessions information are specified in xml form.
3. **Queuing** – in the current simulation *“Drop Tail Queue”* type with frame capacity 100 frames is chosen. The frame capacity is selected empirically to guarantee no packet loss at 20 ms maximum delay.
4. **Scenario Manager** – the *ScenarioManager* module implements the functions of the SDN controller and manages the switching off of networks element. The commands are defined in the form of xml script. The module:

* Reroutes the traffic by initiating a new RSVP session
* Binds the newly defined path to the FEC, linked to the corresponding destination
* Deletes the old path’s LSP
* Disconnects the chosen network element

The operation is described in the sequence chart, presented on the following figure:

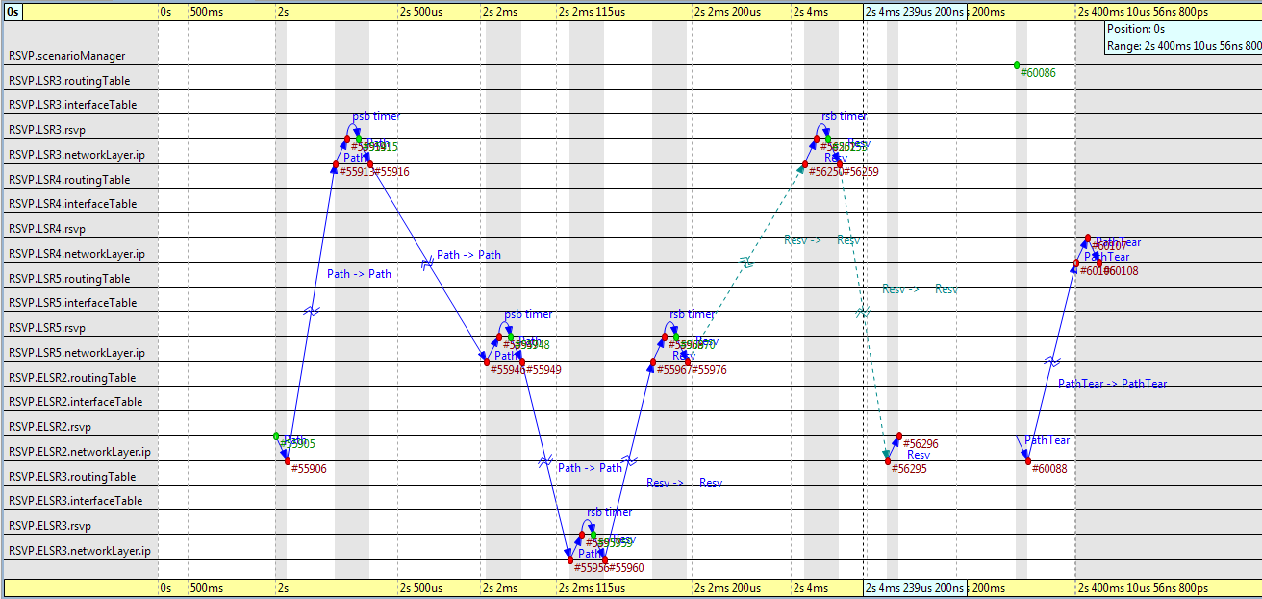
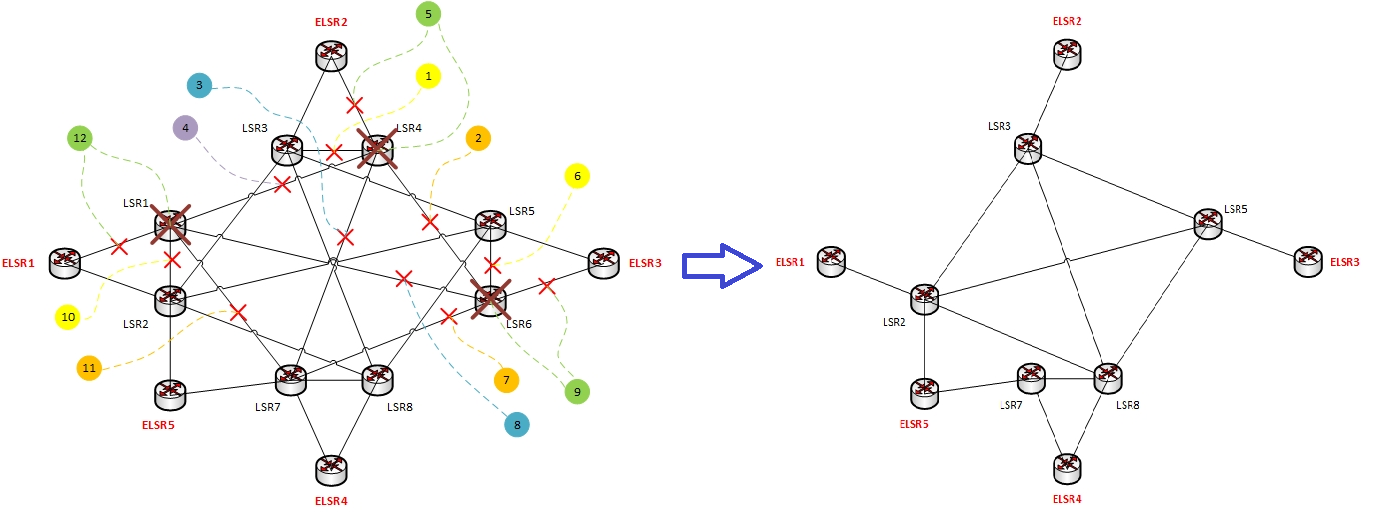


Fig. 7.5 RSVP session sequence chart

In the current simulation *RSVP.ELSR2* edge router initiates the session by sending RSVP path message to *RSVP.LSR3*, which defines its own interface from the EXPLICIT\_ROUTE object, records information about the outgoing interface in the RECORD\_ROUTE object and forwards the message to the next hop (*RSVP.LSR3*). This is repeated until the last hop (*RSVP.ELSR3*) is reached. *RSVP.ELSR3* then initiates a RSVP reservation message, which is propagated back to *RSVP.ELSR2*, recording the route and the reserved bandwidth. After the new LSP is established, *RSVP.ELSR2* deletes the old path by sending a PATH TEAR message and *ScenarioManager* module disconnects the chosen network element.

The sequence in which the network elements are switched off and the resulting topology are presented on figure 7.6.



1. b)

Fig. 7.6 Switching off scenario: a) switching off sequence; b) resulting topology

The simulation process goes through the following steps:

1. *RSVP.LSR4* is chosen as the least critical node for the performance of the network.
2. The traffic is rerouted off the least utilized network link through alternative path excluding *RSVP.LSR4.* The link is then switched off.
3. The process is repeated until all links, connected to *RSVP.LSR4* are switched off, and then the node is turned off.
4. The next node is identified.
5. **SLA parameters –** two key parameters are selected for the current simulation:
   1. **Latency –** the latency is measured by the *BurstHost* traffic sink modules (recipx at Fig. 2). As both the sender and the receiver are directly connected to the edge core routers, the measured value represents the delay through the core network. The chosen maximum value is 20ms. It is defined as the maximum delay from provider edge router to provider edge router.
   2. **Packet loss –** it is defined as the average percentage of lost packets over the simulation period. Packet loss is measured for every interface as ratio between transmitted packets and dropped packets. The maximum packet loss value is set to 0.15%.

**Chapter 8**

**Result Analysis**

The performance of the proposed algorithm is evaluated in this section. As a starting point the initial values of maximum network utilization, packet loss and the variations in packet delay are presented. The switch off process is then activated and the changes in network performance are examined. At every switch off step, the packet loss and maximum delay values are verified. If the values are above the acceptable level, the amount of generated traffic is reduced until the SLA-required performance is restored. In the end the achieved energy conservation under predefined connectivity and SLA constraints is evaluated.

## Initial network state

The simulation is activated for 20s without switching off network elements. The amount of generated traffic is not varied. Packet loss, maximum utilization and delay variations are examined.

## Maximum utilization

The maximum utilization scalar value is measured on every network interface and the results are presented in the form of a bar chart. Maximum values are chosen as the core network is scaled to handle traffic burst without degrading the SLA parameters.

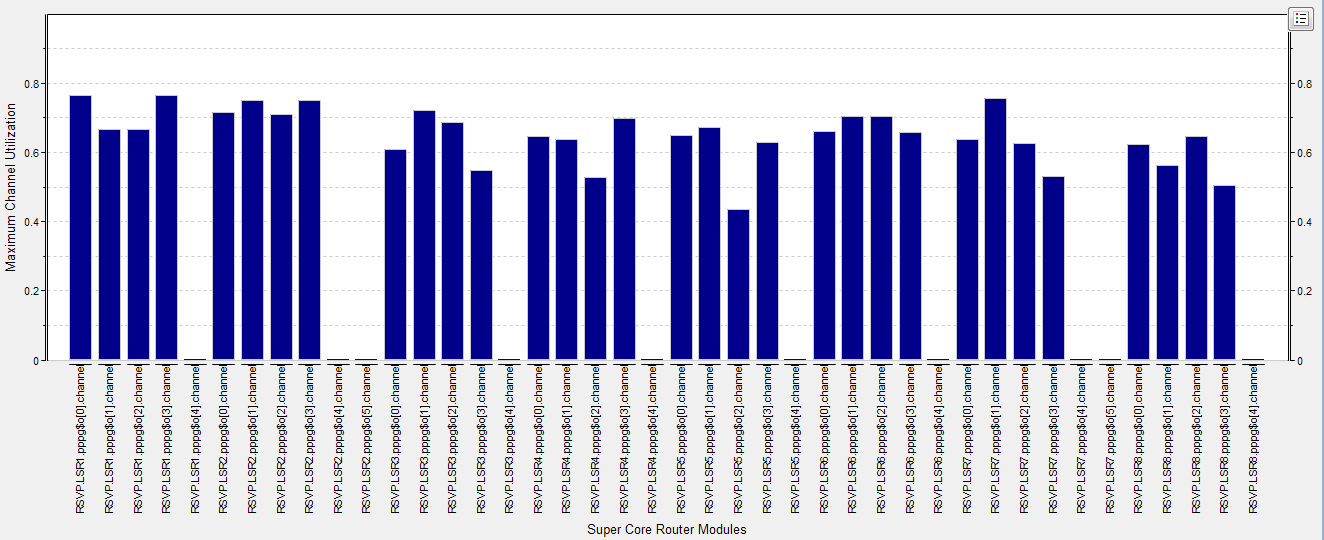


Fig. 8.1 Maximum Utilization Bar Chart

The maximum utilization values vary for every interface due to the different number of traffic generating hosts behind every edge core router, the uneven distribution of the send intervals and the bursty nature of the generated traffic. The values are distributed around 65%, with highest values of close to 79% and minimum values around 50%. The network is scaled to meet future traffic growth.

## Latency

The end to end delay value is measured by every *BurstHost* traffic sink module and essentially represents the delay between the ingress edge core router and the egress edge core router. The maximum scalar values as well as time variations of the delay values are measured.

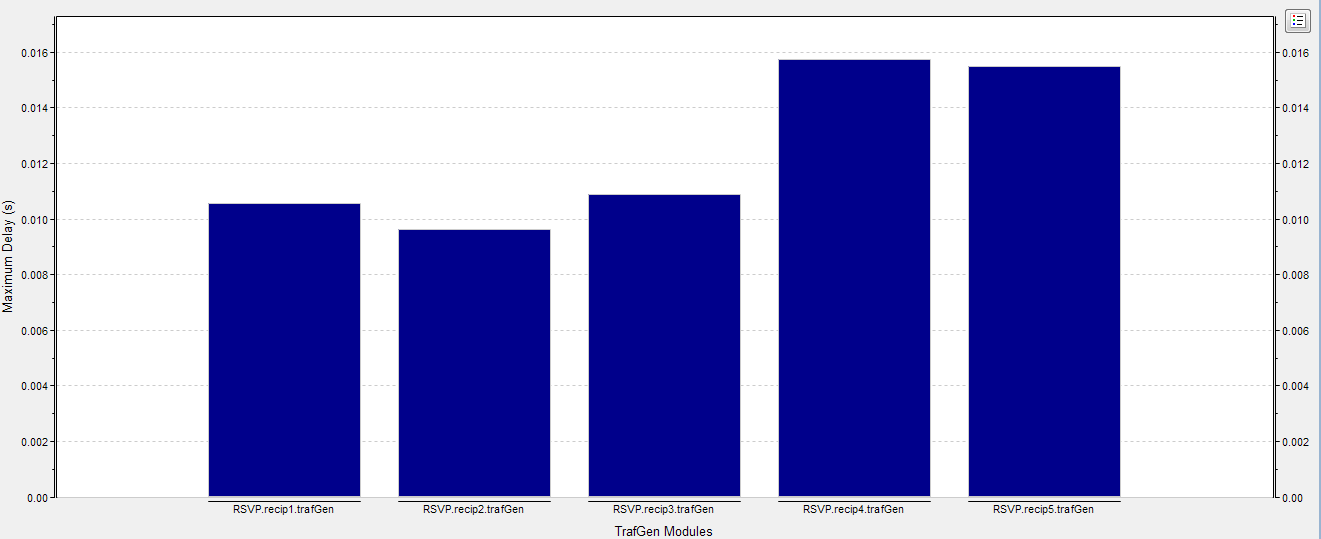


Fig. 8.2 Maximum end to end delay values for every BurstHost traffic sink

The maximum delay values are under 16ms, which is well below the SLA threshold of 20ms.



Fig. 8.3 End to end delay vector line chart

The chart shows that the majority of data packets experience delay that is lower than 8ms. However, there are distinct spikes that reach and exceed 16ms due to the bursty nature of the generated traffic.

## Packet Loss

No packet loss is measured on any of the network interfaces.

## Network performance after switching off process

The influence of the switching off process on the network performance parameters is examined in this chapter. An expected increase in the channel utilization, end to end delay and number of dropped packets is observed.

## Network Utilization

The simulation time is set to 40s. Switching off of network elements is scheduled at predefined simulation times. An alternative LSP is configured for every LSP that passes through the network element before it is turned off. Network elements are iteratively switched off at 10s, 15s, 20s, 25s, 30s, and 35s. The alternations in mean channel utilization of the network elements that carry the newly established LSPs are examined.

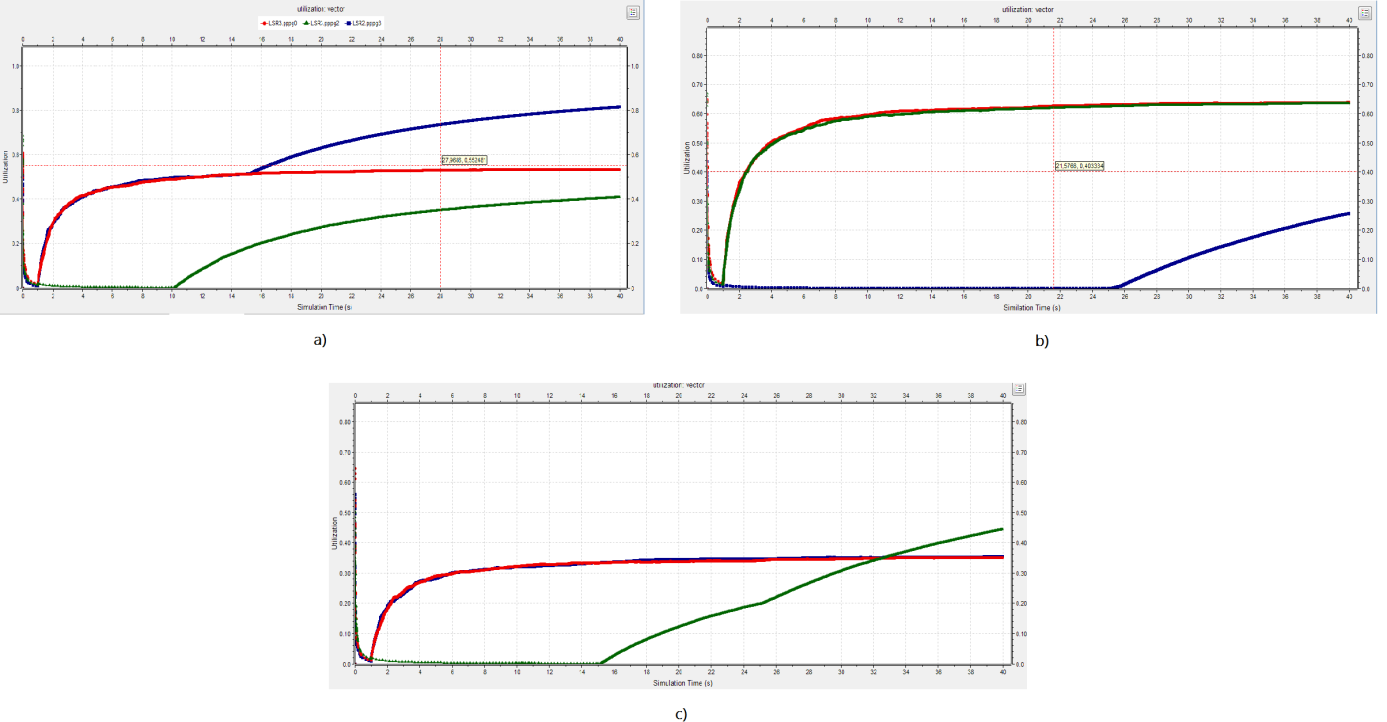


Fig. 8.4 Mean utilization vector: a) RSVP.LSR3 b) RSVP.LSR5 c) RSVP.LSR8

At 10s, a new LSP is created between *RSVP.LSR3* and *RSVP.LSR5*. Steady rise in the mean utilization of LSR3.pppg[2] (fig.8.4a - green) is observed. At 15s , a new LSP is created between *RSVP.LSR3* and *RSVP.LSR8*, resulting in rise in the utilization of LSR3.pppg[3] (fig.8.4a - blue) and LSR8.pppg[1] (fig.8.4c - green). At 25s, two LSPs are created in both directions between *RSVP.LSR5* and *RSVP.LSR8* and a rise of utilization of LSR5.pppg[2] (fig.8.4b - blue) and LSR8.pppg[1] (fig.8.5c - green) is observed.

## Latency

The observed rise in the utilization of certain elements described in section 8.2.1 reflects on the end to end delay experienced by the packets within the core.

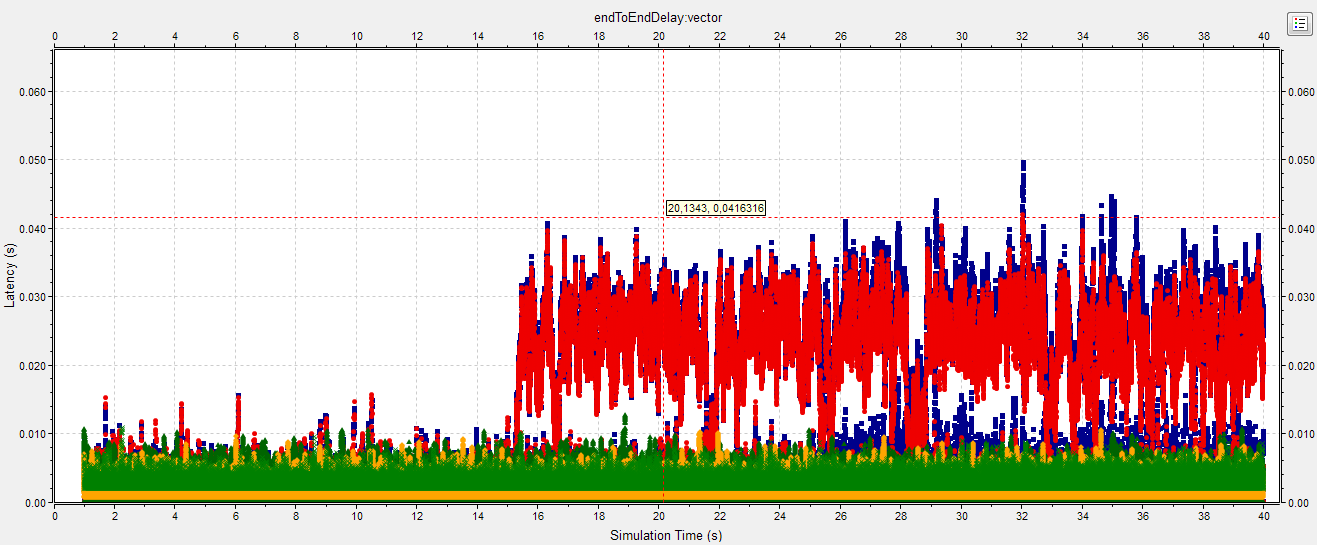


Fig. 8.5 End to end delay vector line chart

From the line chart presented on fig. 8.5 it can be noticed that the rise of the end to end delay values begins at time 15s or when the second network element is switched off. From fig.8.5a it can be noticed that the decay in the network performance begins when the mean utilization of the link rises above 60%. At 20s, 25s and 30s further rise in the end to end delay values can be observed.

## Packet loss

The number of dropped packets is also examined after each consecutive element is switched off. The dropping of data packets also begins at 15s, when the second network element is switched off, the mean utilization of some elements rises above 60% and the rise of the end to end delay parameter is observed. It should be pointed out that the packet loss parameter can be controlled by changing the queue length in the core routers. If the queue length is increased, the packet drop rate falls, but the packet delay increases, which is still not acceptable in the examined scenario.

A key observation that verifies the successful operation of the proposed algorithm is the fact that no data packets are dropped due to the interface switching off. The only packets that are dropped are the RSVP hello messages from the neighboring nodes, which give them an indication that these interfaces are not operational anymore.

## Achieved energy savings

The selected network elements are switched off in the order described in section 7.3. After each consecutive switching off it is verified if the SLA requirements for packet loss and end to end delay are fulfilled. If the measured values exceed the predefined threshold values (delay 20ms, packet loss 0.15%), the amount of generated traffic is reduced. The achieved energy savings vs. the percentage reduction of traffic is plotted on fig. 8.6. The achieved energy conservation is calculated as a ratio of the number of switched off elements vs. the total number of network elements. Each switched off node is accounted as 16.67 links. The ratio is derived from the energy consumption values for core network elements, discussed in [3].

Fig. 8.6 Achieved energy conservation chart

Two main observations can be derived from the obtained results. First, considerable energy savings can be obtained only from switching off entire network nodes. The energy conservation obtained by powering off network links is very small. Second, safe switching off can be performed after the traffic demand decreases by at least 33%.

The first network link that was turned off was the link between *RSVP.LSR3* and *RSVP.LSR4.* This link is a redundant back up link, which is not utilized under normal operating conditions. Turning off this link naturally does not affect the network performance. The second link is an operational link between *RSVP.LSR4* and *RSVP.LSR6*. Its turning off does not affect severely the SLA parameters. The turning off of the third link (between *RSVP.LSR4* and *RSVP.LSR1*) results in high increase in the end to end delay value. The normal network performance is restored after the traffic decreases with more than 33%. Switching off of subsequent network elements required much smaller decrease in generated traffic. An entire network node and the five links connected to it can be turned off at 34% decrease. The following two links can be turned off at 39% and the second node and the remaining connected links – at 42% decrease in traffic generation. The last possible third network node is safely turned off when the traffic decreases by at least 47%. The total achieved energy savings reach 42.52%.

## A glance at the big picture

The current project focused on analyzing the achievable energy savings in the core part of the network. In order to attain network resilience, however, other parts of the provider’s network (edge, access) are also overprovisioned. Furthermore, it is widely discussed that the rise in bandwidth demand causes providers to migrate also their mobile backhaul to IP-based MPLS solutions [3, 4]. The above mentioned key points make the presented solution applicable also to the edge and access part of the network. This might lead to even larger energy savings and more efficient operation of the entire provider’s network. The approach, described in [13], proved the above statement by examining the potential for switching off of both the core and the edge part and achieving considerable reduction in the total energy consumption.

**Chapter 9**

**Conclusion and future work**

An energy saving methodology, that is based on resource consolidation and turning off underutilized network elements, was proposed. The methodology focused on MPLS core networks, where the resource overprovisioning is considerable. The concept was based on the observation that the network utilization experiences considerable variations, based on time of the day, time of the week, etc. For example the high traffic demand in the peak hours of the day is opposed to the much lower demand during the night [7]. It was demonstrated that during the low demand time intervals, it was possible to concentrate the traffic in a small subset of nodes or links, which allowed the unused nodes/links to be switched off. The project described centralized architecture, with an SDN controller and underlying MPLS enabled core routers. An algorithm for redirecting traffic off the least utilized network elements, based on MPLS TE engineering techniques, was proposed. Its performance was examined on a realistic network topology, based on the core network architecture of a national telecommunications services provider. It was demonstrated that the turning off procedure results in degradation of network performance in terms of latency and packet loss parameters due to increased utilization of some links. It was observed, however, that the proposed algorithm allowed safe turning off of networks element without loss of data packets due to interface switch off. The algorithm’s performance was then evaluated in terms of the achieved energy conservation in accordance with predefined connectivity and quality of service constraints. The elements were iteratively switched off and the generated traffic was decreased until SLA requirements were covered. The results showed achievable conservation up to 45.52% when traffic was decreased by 47%.

The performed experiment demonstrated that considerable energy savings are possible in periods of low utilization. Better performance, however, can be achieved if load balancing and traffic distribution techniques are utilized when the traffic is rerouted off the networks elements of interest. That might be realized, for example, by implementing better routing techniques, as SDN application modules. Furthermore, fault recovery mechanisms in case of link or node failure should be examined in future research.

**References**

[1] Thomas Wehmeier, *Understanding today’s smartphone user: An analysis of data-usage patterns in the world’s most advanced 4G LTE markets*, Informa Telecoms & Media, Mobidia, 2013

[2] Peter Croy, *LTE Backhaul Requirements: A Reality Check*, Aviat Networks, 2011

[3] White paper, *Managing LTE IP Transport Networks with Route Analytics*, Packet design, Inc., 2013

[4] White paper, *MPLS-TP: The Perfect Fit for Aggregation Networks*, <http://www.otn.be/MPLS-TP>, OTN Systems, 2014

[5] White paper, *Architectural Considerations for Backhaul of 2G/3G and Long Term Evolution Networks,* http://www.cisco.com/c/en/us/solutions/collateral/service-provider/mobile-internet/white\_paper\_c11-613002.html , Cisco, 2010

[6] S. Lambert, W. van Heddeghem, W. Vereecken, B. Lannoo, D. Colle, and M. Pickavet, *Worldwide electricity consumption of communication networks*, Opt. Express 20, B513-B524, 2012

[7] Utpal Paul, Anand Prabhu Subramanian, Milind Madhav Buddhikot, SamirR.Das, *Understanding Traffic Dynamics in Cellular Data Networks,* INFOCOM, 2011 Proceedings IEEE , vol., no., pp.882,890, 10-15 April 2011

[8] Reviriego, P.; Hernandez, J.A.; Larrabeiti, D.; Maestro, J.A., *Performance evaluation of energy efficient ethernet*, Communications Letters, IEEE , vol.13, no.9, pp.697,699, Sept. 2009

[9] Chengchen Hu; Chunming Wu; Wei Xiong; Binqiang Wang; Jiangxing Wu; Ming Jiang, *On the design of green reconfigurable router toward energy efficient internet*, Communications Magazine*, IEEE* , vol.49, no.6, pp.83,87, June 2011

[10] Le Rouzic, E.; Bonetto, E.; Chiaraviglio, L.; Giroire, F.; Idzikowski, F.; Jimenez, F.; Lange, C.; Montalvo, J.; Musumeci, F.; Tahiri, I.; Valenti, A.; Van Heddeghem, W.; Yabin Ye; Bianco, A.; Pattavina, A., *TREND towards more energy-efficient optical networks*, Optical Network Design and Modeling (ONDM), 2013 17th International Conference on , vol., no., pp.211,216, 16-19 April 2013

[11] White paper, *Best Practices in Core Network Capacity Planning,* <http://www.cisco.com/c/en/us/solutions/collateral/service-provider/quantum/white_paper_c11-728551.html>, Cisco, 2013

[12] L. Chiaraviglio, M. Mellia, F. Neri, [*Energy-aware networks: Reducing power consumption by switching off network elements*](http://scholar.google.com/citations?view_op=view_citation&hl=bg&user=-qZy1fwAAAAJ&citation_for_view=-qZy1fwAAAAJ:qjMakFHDy7sC), FEDERICA-Phosphorus tutorial and workshop (TNC2008) Bruges (Belgium), 2008

[13] S. Gao, J. Zhou, T. Aya, N. Yamanaka, *Reducing Network Power Consumption Using Dynamic Link Metric Method and Power Off Links*, June 2009

[14] Bianzino, A.P.; Chaudet, C.; Rossi, D.; Rougier, J.; Moretti, S., *The Green-Game: Striking a balance between QoS and energy saving*, Teletraffic Congress (ITC), 2011 23rd International, vol., no., pp.262,269, 6-9 Sept. 2011

[15] White paper, *Software-Defined Networking: The New Norm for Networks*, Open Network Foundation, April 2012

[16] Nunes, B.; Mendonca, M.; Nguyen, X.; Obraczka, K.; Turletti, T., *A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks*, Communications Surveys & Tutorials, IEEE , vol.PP, no.99, pp.1,18

[17] Kamite, Y.; Ikejiri, Y., *Software-Defined Networking: The Current Picture and Future Expectations,* IETF journal, July 2013

[18] Handigol, N.; Seetharaman, S.; Flajslik, M.; McKeown, N.; Johari, R., *Plug-n-serve: Load-balancing web traffic using OpenFlow*, ACM SIGCOMM Demo, 2009

[19] Wang, R.; Butnariu, D.; Rexford, J., *Openflow-based server load balancing gone wild*, In Workshop of HotICE, volume 11, 2011

[20] Nayak, A.K.; Reimers, A.; Feamster, N.; Clark, R., *Resonance: Dynamic access control for enterprise networks*, In Proceedings ofthe 1st ACM workshop on Research on enterprise networking, pages11–18. ACM, 2009

[21] Reitblatt, M.; Foster, N.; Rexford, J.; Schlesinger, C.; Walker, D., *Abstractions for network update*, In Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication, SIGCOMM ’12, pages 323–334, New York, NY, USA, 2012. ACM

[22] Heller, B.; Seetharaman, S.; Mahadevan, P.; Yiakoumis, Y.; Sharma, P.; Banerjee, S.; McKeown, N., *Elastictree: Saving energy in data center networks*, In Proceedings of the 7th USENIX conference onNetworked systems design and implementation, pages 17–17. USENIXAssociation, 2010

[23] *Inter-datacenter WAN with centralized TE using SDN and OpenFlow*, In Open Networking Summit, April 2012

[24] Jain, S.; Kumar, A.; Mandal, S.; Ong, J.; Poutievski, L.; Singh, A.; Venkata, S.; Wanderer, J.; Zhou, J.; Zhu, M., *et al. B4: Experience with a globally-deployed software-defined WAN*, In Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM, pages 3–14. ACM, 2013

[25] Suresh, L.; Schulz-Zander, J.; Merz, R.; Feldmann, A.; Vazao, T., *Towards programmable enterprise WLANs with Odin*, In Proceedings of the first workshop on Hot topics in software-defined networks, HotSDN ’12, pages 115–120, New York, NY, USA, ACM, 2012

[26] Optical transport working group OTWG, In Open Networking Foundation ONF, 2013

[27] [U. Lakshman](http://www.ciscopress.com/authors/bio.aspx?a=ef33a61d-651e-465c-a041-b31a168cd3a8), [L. Lobo](http://www.ciscopress.com/authors/bio.aspx?a=412e6124-f9d3-4c69-b093-cab844cd60ab), *MPLS Configuration on Cisco IOS Software, Chapter 9:* [*MPLS Traffic Engineering*](http://www.ciscopress.com/articles/article.asp?p=426640), Pearson Education 2006, Cisco Press, USA, ISBN:978-1-58705-199-9

[28] A. Sharafat, S. Das, G. Parulkar, N. McKeown, [*MPLS-TE and MPLS VPNs with OpenFlow*](http://conferences.sigcomm.org/sigcomm/2011/papers/sigcomm/p452.pdf), demonstration at SIGCOMM, Toronto, August 2011.

[29] S. Das, A.Sharafat, G. Parulkar, N. McKeown, [*MPLS with a Simple OPEN Control Plane*](http://archive.openflow.org/wk/images/d/d6/OFC11_symposia_MPLS_final.pdf), invited talk at Packet Switching Symposium at OFC/NFOEC'11, Los Angeles, March 2011

[30] White paper, *Multi-Protocol Label Switching (MPLS) Conformance and Performance Testing,* IXIA, January 2014

[31] Reed, B., *What's next for MPLS*, Network World, December 21, 2009, <http://www.networkworld.com/news/2009/122109-mpls-future.html?page=1>

[32] Medved, J. ; McLachlan, A. ; Meyer, D., *MPLS-TP Pseudowire Configuration using OpenFlow 1.3,* IETF Internet draft - draft-medved-pwe3*-of-config-01,* July 10, 2012

[33] White paper, *MPLS Basics Introduction,* H3C Technologies Co, <http://www.h3c.com/portal/Products___Solutions/Technology/MPLS/Technology_Introduction/200702/201197_57_0.htm>, Accessed on 25 April 2014

[34] Perros, H., *Connection-oriented Networks: SONET/SDH, ATM, MPLS and Optical Networks, Chapter 7: Label Distribution Protocols,* pp. 149-179, John Wiley & Sons, 2005, ISBN:0470021632

[35] XiPeng Xiao; Telkamp, T.; Fineberg, V.; Cheng Chen; Ni, L.M., *A practical approach for providing QoS in the Internet backbone*, Communications Magazine, IEEE , vol.40, no.12, pp.56,62, Dec 2002

[36] Garcia, J.-M.; Hamouda, M. Z. B.; Brun, O., *Network Planning: Traffic Matrices Estimation and Demand Uncertainty*, Stud. Inform. Univ. 8 (2) , 235-262 , 2010

[37] Evans, J., *Best Practices for Determining the Traffic Matrix in IP Networks,* PLNOG 3, Sep 2009

[38] Betgé-Brezetz, S.; Martinot, O.; Delégue, G.; Marilly, E., *Pro-Active SLA Assurance for Next Generation Network*, World Telecommunication Congress, 2002

[39] Telkamp, T.; Maghbouleh, A.; Sharma, V.; Gordon, S., *Internet Traffic is not Self-Similar at Timescales Relevant to Quality of Service*, February 2004

[40] Gunnar, A.; Johansson, M.; Telkamp, T., [*Traffic Matrix Estimation on a Large IP Backbone - A Comparison on Real Data*](http://www.ltcm.net/%7Etelkamp/papers/p149-gunnar.pdf), Sept. 2004, Published in Proceedings of the [2004 ACM Internet Measurement Conference](http://www-net.cs.umass.edu/imc-2004/), Taormina, Sicily, Oct. 25-27, 2004

[41] White paper, *Building Accurate Traffic Matrices with Demand Deduction,* http://www.cisco.com/c/en/us/solutions/collateral/service-provider/quantum/white\_paper\_c11-728552.html, Cisco, 2013

[42] Zhang, Y.; Roughan, M.; Duffield, N.; Greenberg, A., *Fast Accurate Computation of Large-Scale IP Traffic Matrices from Link Loads*, ACM SIGMETRICS, 2003

[43] Farrel, A.; *The Impact of SDN On MPLS Networks*, 15th annual conference –MPLS 2012, Juniper Networks

[44] Casado, M.; Koponen, T.; Shenker, S.; Tootoonchian, A.; [*Fabric: A Retrospective on Evolving SDN*](http://conferences.sigcomm.org/sigcomm/2012/paper/hotsdn/p85.pdf), HotSDN 2012

[45] Arsenio, A. M.; Silva, T., *Energy Efficiency and Failure Recovery Mechanisms for communication Networks*, Chapter 18 in Green Networking and Communications: ICT for Sustainability, pp. 438-455, CRC Press, October 29, 2013

[46] Chen, D.; Lü, L.; Shang, M.; Zhang, Y.; Zhou, T*., Identifying influential nodes in complex networks, Physica A: Statistical Mechanics and its Applications*, Volume 391, Issue 4, 15 February 2012, Pages 1777-1787, ISSN 0378-4371.

[47] Chiaraviglio, L.; Mellia, M.; Neri, F., Minimizing ISP Network Energy Cost: Formulation and Solutions, IEEE/ACM Transactions on Networking (TON), 2012

[48] Varga , A.; Hornig, R., *An overview of the OMNeT++ simulation environment*, Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops, page 60, 2008

[49] *Welcome to the INET Framework!*, [*http://inet.omnetpp.org/*](http://inet.omnetpp.org/)*,* Accessed on 28.5. 2014

[50] Sinha, R.; Papadopoulos, C.; Heidemann, J., *Internet Packet Size Distributions: Some Observations*, Technical Report ISI-TR-2007-643, USC/Information Sciences Institute, May, 2007

[51] White paper, *What is Network Latency and Why Does It Matter?,* 03b Networks, November 2008

**Appendix A: *omnetpp.ini***

[General]

network = RSVP

sim-time-limit = 20s

*#cpu-time-limit= 5800000s*

total-stack = 256MiB

tkenv-plugin-path = ../../../etc/plugins

*#number of hosts*

\*\*.nodeNo1 = 1

\*\*.nodeNo2 = 1

\*\*.nodeNo3 = 1

\*\*.nodeNo4 = 1

\*\*.nodeNo5 = 1

\*\*.nodeNo6 = 4

\*\*.nodeNo7 = 4

\*\*.nodeNo8 = 3

\*\*.nodeNo9 = 4

\*\*.nodeNo10 = 3

\*\*.nodeNo11 = 4

\*\*.nodeNo12 = 4

\*\*.nodeNo13 = 3

\*\*.nodeNo14 = 3

\*\*.nodeNo15 = 4

*# app config*

*# packet size 1300B*

\*\*.sender1[\*].trafGenType = "IPvXTrafGen"

\*\*.recip1.trafGenType = "IPvXTrafSink"

\*\*.sender1[\*].trafGen.startTime = 1s

\*\*.sender1[\*].trafGen.sendInterval = 1.4\*15\*0.03s

\*\*.sender1[\*].trafGen.numPackets = 100000

\*\*.sender1[\*].trafGen.protocol =17

\*\*.sender1[\*].trafGen.packetLength = uniform(1290B, 1310B)

\*\*.sender1[\*].trafGen.destAddresses = "recip2 recip3 recip4 recip5"

\*\*.recip1.trafGen.protocol = 17

\*\*.sender2[\*].trafGenType = "IPvXTrafGen"

\*\*.recip2.trafGenType = "IPvXTrafSink"

\*\*.sender2[\*].trafGen.startTime = 1s

\*\*.sender2[\*].trafGen.sendInterval = 1.4\*exponential (15\*0.04ms)

\*\*.sender2[\*].trafGen.numPackets = 100000

\*\*.sender2[\*].trafGen.protocol =17

\*\*.sender2[\*].trafGen.packetLength = uniform(1290B, 1310B)

\*\*.sender2[\*].trafGen.destAddresses = "recip1 recip3 recip4 recip5"

\*\*.recip2.trafGen.protocol = 17

\*\*.sender3[\*].trafGenType = "IPvXTrafGen"

\*\*.recip3.trafGenType = "IPvXTrafSink"

\*\*.sender3[\*].trafGen.startTime = 1s

\*\*.sender3[\*].trafGen.sendInterval = 1.4\*exponential (15\*0.05ms)

\*\*.sender3[\*].trafGen.numPackets = 100000

\*\*.sender3[\*].trafGen.protocol =17

\*\*.sender3[\*].trafGen.packetLength = uniform(1290B, 1310B)

\*\*.sender3[\*].trafGen.destAddresses = "recip1 recip2 recip4 recip5"

\*\*.recip3.trafGen.protocol = 17

\*\*.sender4[\*].trafGenType = "IPvXTrafGen"

\*\*.recip4.trafGenType = "IPvXTrafSink"

\*\*.sender4[\*].trafGen.startTime = 1s

\*\*.sender4[\*].trafGen.sendInterval = 1.4\*exponential (15\*0.06ms)

\*\*.sender4[\*].trafGen.numPackets = 100000

\*\*.sender4[\*].trafGen.protocol =17

\*\*.sender4[\*].trafGen.packetLength = uniform(1290B, 1310B)

\*\*.sender4[\*].trafGen.destAddresses = "recip1 recip2 recip3 recip5"

\*\*.recip4.trafGen.protocol = 17

\*\*.sender5[\*].trafGenType = "IPvXTrafGen"

\*\*.recip5.trafGenType = "IPvXTrafSink"

\*\*.sender5[\*].trafGen.startTime = 1s

\*\*.sender5[\*].trafGen.sendInterval = 1.4\*exponential (15\*0.07ms)

\*\*.sender5[\*].trafGen.numPackets = 100000

\*\*.sender5[\*].trafGen.protocol =17

\*\*.sender5[\*].trafGen.packetLength = uniform(1290B, 1310B)

\*\*.sender5[\*].trafGen.destAddresses = "recip1 recip2 recip3 recip4"

\*\*.recip5.trafGen.protocol = 17

*# packet size 1500B*

\*\*.sender6[\*].trafGenType = "IPvXTrafGen"

\*\*.sender6[\*].trafGen.startTime = 1s

\*\*.sender6[\*].trafGen.sendInterval = 1.4\*15\*0.014s

\*\*.sender6[\*].trafGen.numPackets = 1000000

\*\*.sender6[\*].trafGen.protocol =17

\*\*.sender6[\*].trafGen.packetLength = uniform(1450B, 1500B)

\*\*.sender6[\*].trafGen.destAddresses = "recip2 recip3 recip4 recip5"

\*\*.sender7[\*].trafGenType = "IPvXTrafGen"

\*\*.sender7[\*].trafGen.startTime = 1s

\*\*.sender7[\*].trafGen.sendInterval = 1.4\*uniform (15\*0.013s, 15\*0.017s)

\*\*.sender7[\*].trafGen.numPackets = 1000000

\*\*.sender7[\*].trafGen.protocol =17

\*\*.sender7[\*].trafGen.packetLength = uniform(1450B, 1500B)

\*\*.sender7[\*].trafGen.destAddresses = "recip1 recip3 recip4 recip5"

\*\*.sender8[\*].trafGenType = "IPvXTrafGen"

\*\*.sender8[\*].trafGen.startTime = 1s

\*\*.sender8[\*].trafGen.sendInterval = 1.4\*exponential (15\*0.015s)

\*\*.sender8[\*].trafGen.numPackets = 1000000

\*\*.sender8[\*].trafGen.protocol =17

\*\*.sender8[\*].trafGen.packetLength = uniform(1450B, 1500B)

\*\*.sender8[\*].trafGen.destAddresses = "recip1 recip2"

\*\*.sender9[\*].trafGenType = "IPvXTrafGen"

\*\*.sender9[\*].trafGen.startTime = 1s

\*\*.sender9[\*].trafGen.sendInterval = 1.4\*15\*0.015s

\*\*.sender9[\*].trafGen.numPackets = 1000000

\*\*.sender9[\*].trafGen.protocol =17

\*\*.sender9[\*].trafGen.packetLength = uniform(1450B, 1500B)

\*\*.sender9[\*].trafGen.destAddresses = "recip1 recip2 recip3 recip5"

\*\*.sender10[\*].trafGenType = "IPvXTrafGen"

\*\*.sender10[\*].trafGen.startTime = 1.8s

\*\*.sender10[\*].trafGen.sendInterval = 1.4\*exponential(15\*0.015s)

\*\*.sender10[\*].trafGen.numPackets = 1000000

\*\*.sender10[\*].trafGen.protocol =17

\*\*.sender10[\*].trafGen.packetLength = uniform(1450B, 1500B)

\*\*.sender10[\*].trafGen.destAddresses = "recip1 recip2 recip3 recip4"

*# packet size 40B*

\*\*.sender11[\*].trafGenType = "IPvXTrafGen"

\*\*.sender11[\*].trafGen.startTime = 1s

\*\*.sender11[\*].trafGen.sendInterval = 1.4\*15\*0.00002s

\*\*.sender11[\*].trafGen.numPackets = 1000000

\*\*.sender11[\*].trafGen.protocol =17

\*\*.sender11[\*].trafGen.packetLength = uniform(40B, 45B )

\*\*.sender11[\*].trafGen.destAddresses = "recip2 recip3 recip4 recip5"

\*\*.sender12[\*].trafGenType = "IPvXTrafGen"

\*\*.sender12[\*].trafGen.startTime = 1s

\*\*.sender12[\*].trafGen.sendInterval = 1.4\*15\*0.00004s

\*\*.sender12[\*].trafGen.numPackets = 1000000

\*\*.sender12[\*].trafGen.protocol =17

\*\*.sender12[\*].trafGen.packetLength = uniform(40B, 45B )

\*\*.sender12[\*].trafGen.destAddresses = "recip1 recip3 recip4 recip5"

\*\*.sender13[\*].trafGenType = "IPvXTrafGen"

\*\*.sender13[\*].trafGen.startTime = 1s

\*\*.sender13[\*].trafGen.sendInterval = 1.4\*exponential(15\*0.00002s)

\*\*.sender13[\*].trafGen.numPackets = 1000000

\*\*.sender13[\*].trafGen.protocol =17

\*\*.sender13[\*].trafGen.packetLength = uniform(40B, 45B )

\*\*.sender13[\*].trafGen.destAddresses = "recip1 recip2"

\*\*.sender14[\*].trafGenType = "IPvXTrafGen"

\*\*.sender14[\*].trafGen.startTime = 1s

\*\*.sender14[\*].trafGen.sendInterval = 1.4\*15\*0.00005s

\*\*.sender14[\*].trafGen.numPackets = 1000000

\*\*.sender14[\*].trafGen.protocol =17

\*\*.sender14[\*].trafGen.packetLength = uniform(40B, 45B )

\*\*.sender14[\*].trafGen.destAddresses = "recip1 recip2 recip3 recip5"

\*\*.sender15[\*].trafGenType = "IPvXTrafGen"

\*\*.sender15[\*].trafGen.startTime = 1s

\*\*.sender15[\*].trafGen.sendInterval = 1.4\*exponential (15\*0.00004s)

\*\*.sender15[\*].trafGen.numPackets = 1000000

\*\*.sender15[\*].trafGen.protocol =17

\*\*.sender15[\*].trafGen.packetLength = uniform(40B, 45B )

\*\*.sender15[\*].trafGen.destAddresses = "recip1 recip2 recip4"

*# LSR configuration*

*#\*\*.ELSR{3,4,5}.classifier.conf = xmldoc("\_fec.xml")*

*#\*\*.ELSR{3,4,5}.rsvp.traffic = xmldoc("\_traffic.xml")*

\*\*.ELSR\*.rsvp.helloInterval = 0.2s

\*\*.ELSR\*.rsvp.helloTimeout = 0.5s

\*\*.ELSR\*.libTable.config = **xmldoc**("\_lib.xml")

\*\*.LSR\*.classifier.config = **xmldoc**("\_fec.xml")

\*\*.LSR\*.rsvp.traffic = **xmldoc**("\_traffic.xml")

\*\*.LSR\*.rsvp.helloInterval = 0.2s

\*\*.LSR\*.rsvp.helloTimeout = 0.5s

\*\*.LSR\*.libTable.config = **xmldoc**("\_lib.xml")

\*\*.ELSR1.classifier.config = **xmldoc**("ELSR1\_fec.xml")

\*\*.ELSR1.rsvp.traffic = **xmldoc**("ELSR1\_rsvp.xml")

\*\*.ELSR2.classifier.config = **xmldoc**("ELSR2\_fec.xml")

\*\*.ELSR2.rsvp.traffic = **xmldoc**("ELSR2\_rsvp.xml")

\*\*.ELSR3.classifier.config = **xmldoc**("ELSR3\_fec.xml")

\*\*.ELSR3.rsvp.traffic = **xmldoc**("ELSR3\_rsvp.xml")

\*\*.ELSR4.classifier.config = **xmldoc**("ELSR4\_fec.xml")

\*\*.ELSR4.rsvp.traffic = **xmldoc**("ELSR4\_rsvp.xml")

\*\*.ELSR5.classifier.config = **xmldoc**("ELSR5\_fec.xml")

\*\*.ELSR5.rsvp.traffic = **xmldoc**("ELSR5\_rsvp.xml")

\*\*.routerId = "auto"

\*\*.routingFile = ""

*# NIC configuration*

\*\*.ppp[\*].queueType = "DropTailQueue" *# in routers*

\*\*.ppp[\*].queue.frameCapacity = 100 *# in routers*

\*.configurator.dumpConfig = "ipv4config.xxx"

*# scenario*

\*\*.scenarioManager.script = **xmldoc**("empty.xml")

\*\*.vector-recording = **true**

\*\*.scalar-recording = **true**

\*\*.**channel**.throughput.result-recording-modes = vector

\*\*.**channel**.utilization.result-recording-modes = **default**, vector, min, max, timeavg, stats, histogram

\*\*.**channel**.channelBusy.vector-recording = **true**

**Appendix B: *package.ned***

**package** rsvp\_te2;

**import** inet.applications.generic.IIPvXTrafficGenerator;

**import** inet.applications.generic.IPvXTrafGen;

**import** inet.applications.tcpapp.TCPSrvHostApp;

**import** inet.base.LifecycleController;

**import** inet.examples.inet.routerperf.BurstHost;

**import** inet.examples.inet.tcptimestamps.NormalPath;

**import** inet.networklayer.autorouting.ipv4.HostAutoConfigurator;

**import** inet.networklayer.autorouting.ipv4.IPv4NetworkConfigurator;

**import** inet.networklayer.ospfv2.OSPFRouting;

**import** inet.nodes.inet.StandardHost;

**import** inet.nodes.mpls.RSVP\_LSR;

**import** inet.world.scenario.ScenarioManager;

**@license**(LGPL);

*//*

*// TODO documentation*

*//*

**network** RSVP

{

**parameters**:

**int** nodeNo1;

**int** nodeNo2;

**int** nodeNo3;

**int** nodeNo4;

**int** nodeNo5;

**int** nodeNo6;

**int** nodeNo7;

**int** nodeNo8;

**int** nodeNo9;

**int** nodeNo10;

**int** nodeNo11;

**int** nodeNo12;

**int** nodeNo13;

**int** nodeNo14;

**int** nodeNo15;

**@display**("bgb=1028,707");

**submodules**:

LSR1: RSVP\_LSR {

**@display**("p=239,337");

peers = "ppp0 ppp1 ppp2 ppp3 ppp4";

**gates**:

pppg[5];

}

LSR2: RSVP\_LSR {

**@display**("p=239,412");

peers = "ppp0 ppp1 ppp2 ppp3 ppp4 ppp5";

**gates**:

pppg[6];

}

LSR3: RSVP\_LSR {

**@display**("p=436,234");

peers = "ppp0 ppp1 ppp2 ppp3 ppp4";

**gates**:

pppg[5];

}

LSR4: RSVP\_LSR {

**@display**("p=532,234");

peers = "ppp0 ppp1 ppp2 ppp3 ppp4";

**gates**:

pppg[5];

}

LSR5: RSVP\_LSR {

**@display**("p=745,337");

peers = "ppp0 ppp1 ppp2 ppp3 ppp4";

**gates**:

pppg[5];

}

LSR6: RSVP\_LSR {

**@display**("p=745,412");

peers = "ppp0 ppp1 ppp2 ppp3 ppp4";

**gates**:

pppg[5];

}

LSR8: RSVP\_LSR {

**@display**("p=532,504");

peers = "ppp0 ppp1 ppp2 ppp3 ppp4";

**gates**:

pppg[5];

}

LSR7: RSVP\_LSR {

**@display**("p=436,504");

peers = "ppp0 ppp1 ppp2 ppp3 ppp4 ppp5";

**gates**:

pppg[6];

}

ELSR1: RSVP\_LSR {

**@display**("p=129,373");

peers = "ppp0 ppp1";

**gates**:

pppg[3];

}

ELSR2: RSVP\_LSR {

**@display**("p=484,128");

peers = "ppp0 ppp1";

**gates**:

pppg[3];

}

ELSR3: RSVP\_LSR {

**@display**("p=889,367");

peers = "ppp0 ppp1";

**gates**:

pppg[3];

}

ELSR4: RSVP\_LSR {

**@display**("p=484,591");

peers = "ppp0 ppp1";

**gates**:

pppg[3];

}

ELSR5: RSVP\_LSR {

**@display**("p=249,535");

peers = "ppp0 ppp1";

**gates**:

pppg[3];

}

configurator: IPv4NetworkConfigurator {

**@display**("p=74,25");

}

sender1[nodeNo1]: BurstHost {

**parameters**:

**@display**("p=41,190,row,100");

}

recip2: BurstHost {

**parameters**:

**@display**("p=532,25,row,100");

}

recip1: BurstHost {

**parameters**:

**@display**("p=25,438,row,100");

}

sender2[nodeNo2]: BurstHost {

**parameters**:

**@display**("p=632,25,row,100");

}

sender3[nodeNo3]: BurstHost {

**parameters**:

**@display**("p=984,285,row,100");

}

sender4[nodeNo4]: BurstHost {

**parameters**:

**@display**("p=459,669,row,100");

}

sender5[nodeNo5]: BurstHost {

**parameters**:

**@display**("p=94,608,row,100");

}

recip3: BurstHost {

**parameters**:

**@display**("p=889,238,row,100");

}

recip4: BurstHost {

**parameters**:

**@display**("p=415,587,row,100");

}

recip5: BurstHost {

**parameters**:

**@display**("p=129,535,row,100");

}

scenarioManager: ScenarioManager {

**@display**("p=74,71");

}

lifecycleController: LifecycleController {

**@display**("p=74,114");

}

sender6[nodeNo6]: BurstHost {

**parameters**:

**@display**("p=41,246,row,100");

}

sender7[nodeNo7]: BurstHost {

**parameters**:

**@display**("p=632,83,row,100");

}

sender8[nodeNo8]: BurstHost {

**parameters**:

**@display**("p=984,354,row,100");

}

sender9[nodeNo9]: BurstHost {

**parameters**:

**@display**("p=632,556,row,100");

}

sender10[nodeNo10]: BurstHost {

**parameters**:

**@display**("p=187,639,row,100");

}

sender11[nodeNo11]: BurstHost {

**parameters**:

**@display**("p=41,315,row,100");

}

sender12[nodeNo12]: BurstHost {

**parameters**:

**@display**("p=632,151,row,100");

}

sender13[nodeNo13]: BurstHost {

**parameters**:

**@display**("p=984,425,row,100");

}

sender14[nodeNo14]: BurstHost {

**parameters**:

**@display**("p=632,626,row,100");

}

sender15[nodeNo15]: BurstHost {

**parameters**:

**@display**("p=284,669,row,100");

}

**connections**:

LSR1.pppg++ **<-->** { delay = 2.1ms; datarate = 10Mbps; } **<-->** LSR4.pppg++;

LSR1.pppg++ **<-->** { delay = 4ms; datarate = 10Mbps; } **<-->** LSR6.pppg++;

LSR1.pppg++ **<-->** { delay = 1.6ms; datarate = 10Mbps; } **<-->** LSR7.pppg++;

LSR2.pppg++ **<-->** { delay = 2.1ms; datarate = 10Mbps; } **<-->** LSR3.pppg++;

LSR2.pppg++ **<-->** { delay = 3.5ms; datarate = 10Mbps; } **<-->** LSR5.pppg++;

LSR2.pppg++ **<-->** { delay = 1.6ms; datarate = 10Mbps; } **<-->** LSR8.pppg++;

LSR4.pppg++ **<-->** { delay = 2ms; datarate = 10Mbps; } **<-->** LSR6.pppg++;

LSR4.pppg++ **<-->** { delay = 0.6ms; datarate = 10Mbps; } **<-->** LSR7.pppg++;

LSR4.pppg++ **<-->** { delay = 0.01ms; datarate = 10Mbps; } **<-->** LSR3.pppg++;

LSR3.pppg++ **<-->** { delay = 2ms; datarate = 10Mbps; } **<-->** LSR5.pppg++;

LSR3.pppg++ **<-->** { delay = 0.6ms; datarate = 10Mbps; } **<-->** LSR8.pppg++;

LSR5.pppg++ **<-->** { delay = 1.9ms; datarate = 10Mbps; } **<-->** LSR8.pppg++;

LSR5.pppg++ **<-->** { delay = 0.01ms; datarate = 10Mbps; } **<-->** LSR6.pppg++;

LSR6.pppg++ **<-->** { delay = 1.9ms; datarate = 10Mbps; } **<-->** LSR7.pppg++;

LSR7.pppg++ **<-->** { delay = 0.01ms; datarate = 10Mbps; } **<-->** LSR8.pppg++;

LSR2.pppg++ **<-->** { delay = 0.01ms; datarate = 10Mbps; } **<-->** LSR1.pppg++;

ELSR1.pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->**LSR1.pppg++;

ELSR1.pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->**LSR2.pppg++;

ELSR2.pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->**LSR3.pppg++;

ELSR2.pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->**LSR4.pppg++;

ELSR3.pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->**LSR5.pppg++;

ELSR3.pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->**LSR6.pppg++;

ELSR4.pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->**LSR7.pppg++;

ELSR4.pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->**LSR8.pppg++;

ELSR5.pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->**LSR2.pppg++;

ELSR5.pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->**LSR7.pppg++;

**for** i=0..nodeNo1-1 {

sender1[i].pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->** ELSR1.pppg++;

}

**for** i=0..nodeNo2-1 {

sender2[i].pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->** ELSR2.pppg++;

}

**for** i=0..nodeNo3-1 {

sender3[i].pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->** ELSR3.pppg++;

}

**for** i=0..nodeNo4-1 {

sender4[i].pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->** ELSR4.pppg++;

}

**for** i=0..nodeNo5-1 {

sender5[i].pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->** ELSR5.pppg++;

}

**for** i=0..nodeNo6-1 {

sender6[i].pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->** ELSR1.pppg++;

}

**for** i=0..nodeNo7-1 {

sender7[i].pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->** ELSR2.pppg++;

}

**for** i=0..nodeNo8-1 {

sender8[i].pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->** ELSR3.pppg++;

}

**for** i=0..nodeNo9-1 {

sender9[i].pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->** ELSR4.pppg++;

}

**for** i=0..nodeNo10-1 {

sender10[i].pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->** ELSR5.pppg++;

}

**for** i=0..nodeNo11-1 {

sender11[i].pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->** ELSR1.pppg++;

}

**for** i=0..nodeNo12-1 {

sender12[i].pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->** ELSR2.pppg++;

}

**for** i=0..nodeNo13-1 {

sender13[i].pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->** ELSR3.pppg++;

}

**for** i=0..nodeNo14-1 {

sender14[i].pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->** ELSR4.pppg++;

}

**for** i=0..nodeNo15-1 {

sender15[i].pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->** ELSR5.pppg++;

}

ELSR2.pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->** recip2.pppg++;

recip1.pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->** ELSR1.pppg++;

recip3.pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->** ELSR3.pppg++;

recip4.pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->** ELSR4.pppg++;

recip5.pppg++ **<-->** { delay = 0.01ms; datarate = 10Gbps; } **<-->** ELSR5.pppg++;

}

**Appendix C: *\_scenario.xml***

<?xml version="1.0"?>

<scenario>

<at t="10">

<disconnect src-module="LSR4" src-gate="pppg[3]" />

</at>

<at t="10s">

<add-session module="ELSR2.rsvp">

<endpoint>recip3</endpoint>

<tunnel\_id>26</tunnel\_id>

<paths>

<path>

<lspid>261</lspid>

<bandwidth>2000000</bandwidth>

<route>

<node>ELSR2%routerId</node>

<node>LSR3%routerId</node>

<node>LSR5%routerId</node>

<node>ELSR3%routerId</node>

</route>

<color>100</color>

</path>

</paths>

</add-session>

</at>

<at t="10.2s">

<bind-fec module="ELSR2.classifier">

<id>26</id>

<endpoint>recip3</endpoint>

<tunnel\_id>26</tunnel\_id>

<destination>recip3</destination>

<lspid>261</lspid>

</bind-fec>

</at>

<at t="10.4s">

<del-session module="ELSR2.rsvp">

<endpoint>recip3</endpoint>

<tunnel\_id>26</tunnel\_id>

<paths>

<path>

<lspid>260</lspid>

</path>

</paths>

</del-session>

</at>

<at t="10.4">

<disconnect src-module="LSR4" src-gate="pppg[1]" />

</at>

<at t="15s">

<add-session module="ELSR2.rsvp">

<endpoint>recip5</endpoint>

<tunnel\_id>28</tunnel\_id>

<paths>

<path>

<lspid>281</lspid>

<bandwidth>2000000</bandwidth>

<route>

<node>ELSR2%routerId</node>

<node>LSR3%routerId</node>

<node>LSR8%routerId</node>

<node>LSR7%routerId</node>

<node>ELSR5%routerId</node>

</route>

<color>100</color>

</path>

</paths>

</add-session>

</at>

<at t="15.2s">

<bind-fec module="ELSR2.classifier">

<id>28</id>

<endpoint>recip5</endpoint>

<tunnel\_id>28</tunnel\_id>

<destination>recip5</destination>

<lspid>281</lspid>

</bind-fec>

</at>

<at t="15.4s">

<del-session module="ELSR2.rsvp">

<endpoint>recip5</endpoint>

<tunnel\_id>28</tunnel\_id>

<paths>

<path>

<lspid>280</lspid>

</path>

</paths>

</del-session>

</at>

<at t="15.4">

<disconnect src-module="LSR4" src-gate="pppg[2]" />

</at>

<at t="20s">

<add-session module="ELSR1.rsvp">

<endpoint>recip1</endpoint>

<tunnel\_id>15</tunnel\_id>

<paths>

<path>

<lspid>151</lspid>

<bandwidth>2000000</bandwidth>

<route>

<node>ELSR1%routerId</node>

<node>LSR2%routerId</node>

<node>LSR3%routerId</node>

<node>ELSR2%routerId</node>

</route>

<color>100</color>

</path>

</paths>

</add-session>

</at>

<at t="20.2s">

<bind-fec module="ELSR1.classifier">

<id>15</id>

<endpoint>recip2</endpoint>

<tunnel\_id>15</tunnel\_id>

<destination>recip2</destination>

<lspid>151</lspid>

</bind-fec>

</at>

<at t="20.4s">

<del-session module="ELSR1.rsvp">

<endpoint>recip2</endpoint>

<tunnel\_id>15</tunnel\_id>

<paths>

<path>

<lspid>150</lspid>

</path>

</paths>

</del-session>

</at>

<at t="20.4">

<disconnect src-module="LSR4" src-gate="pppg[0]" />

</at>

<at t="20.4">

<disconnect src-module="LSR4" src-gate="pppg[4]" />

<tell module="lifecycleController" target="LSR4" operation="NodeShutdownOperation"/>

</at>

<at t="25">

<disconnect src-module="LSR6" src-gate="pppg[2]" />

</at>

<at t="25s">

<add-session module="ELSR3.rsvp">

<endpoint>recip5</endpoint>

<tunnel\_id>38</tunnel\_id>

<paths>

<path>

<lspid>381</lspid>

<bandwidth>2000000</bandwidth>

<route>

<node>ELSR3%routerId</node>

<node>LSR5%routerId</node>

<node>LSR8%routerId</node>

<node>LSR7%routerId</node>

<node>ELSR5%routerId</node>

</route>

<color>100</color>

</path>

</paths>

</add-session>

<add-session module="ELSR5.rsvp">

<endpoint>recip3</endpoint>

<tunnel\_id>57</tunnel\_id>

<paths>

<path>

<lspid>571</lspid>

<bandwidth>2000000</bandwidth>

<route>

<node>ELSR5%routerId</node>

<node>LSR7%routerId</node>

<node>LSR8%routerId</node>

<node>LSR5%routerId</node>

<node>ELSR3%routerId</node>

</route>

<color>100</color>

</path>

</paths>

</add-session>

</at>

<at t="25.2s">

<bind-fec module="ELSR3.classifier">

<id>38</id>

<endpoint>recip5</endpoint>

<tunnel\_id>38</tunnel\_id>

<destination>recip5</destination>

<lspid>381</lspid>

</bind-fec>

<bind-fec module="ELSR5.classifier">

<id>57</id>

<endpoint>recip3</endpoint>

<tunnel\_id>57</tunnel\_id>

<destination>recip3</destination>

<lspid>571</lspid>

</bind-fec>

</at>

<at t="25.4s">

<del-session module="ELSR3.rsvp">

<endpoint>recip5</endpoint>

<tunnel\_id>38</tunnel\_id>

<paths>

<path>

<lspid>380</lspid>

</path>

</paths>

</del-session>

<del-session module="ELSR5.rsvp">

<endpoint>recip3</endpoint>

<tunnel\_id>57</tunnel\_id>

<paths>

<path>

<lspid>571</lspid>

</path>

</paths>

</del-session>

</at>

<at t="25.4">

<disconnect src-module="LSR6" src-gate="pppg[3]" />

</at>

<at t="30s">

<add-session module="ELSR1.rsvp">

<endpoint>recip3</endpoint>

<tunnel\_id>16</tunnel\_id>

<paths>

<path>

<lspid>161</lspid>

<bandwidth>2000000</bandwidth>

<route>

<node>ELSR1%routerId</node>

<node>LSR2%routerId</node>

<node>LSR5%routerId</node>

<node>ELSR3%routerId</node>

</route>

<color>100</color>

</path>

</paths>

</add-session>

</at>

<at t="30.2s">

<bind-fec module="ELSR1.classifier">

<id>16</id>

<endpoint>recip2</endpoint>

<tunnel\_id>16</tunnel\_id>

<destination>recip3</destination>

<lspid>161</lspid>

</bind-fec>

</at>

<at t="30.4s">

<del-session module="ELSR1.rsvp">

<endpoint>recip3</endpoint>

<tunnel\_id>16</tunnel\_id>

<paths>

<path>

<lspid>160</lspid>

</path>

</paths>

</del-session>

</at>

<at t="30.4">

<disconnect src-module="LSR6" src-gate="pppg[0]" />

<disconnect src-module="LSR6" src-gate="pppg[4]" />

<tell module="lifecycleController" target="LSR6" operation="NodeShutdownOperation"/>

</at>

<at t="35">

<disconnect src-module="LSR1" src-gate="pppg[3]" />

</at>

<at t="35s">

<add-session module="ELSR1.rsvp">

<endpoint>recip3</endpoint>

<tunnel\_id>16</tunnel\_id>

<paths>

<path>

<lspid>161</lspid>

<bandwidth>2000000</bandwidth>

<route>

<node>ELSR1%routerId</node>

<node>LSR2%routerId</node>

<node>LSR5%routerId</node>

<node>ELSR3%routerId</node>

</route>

<color>100</color>

</path>

</paths>

</add-session>

</at>

<at t="35.2s">

<bind-fec module="ELSR1.classifier">

<id>16</id>

<endpoint>recip2</endpoint>

<tunnel\_id>16</tunnel\_id>

<destination>recip3</destination>

<lspid>161</lspid>

</bind-fec>

</at>

<at t="35.4s">

<del-session module="ELSR1.rsvp">

<endpoint>recip3</endpoint>

<tunnel\_id>16</tunnel\_id>

<paths>

<path>

<lspid>160</lspid>

</path>

</paths>

</del-session>

</at>

<at t="35.4">

<disconnect src-module="LSR6" src-gate="pppg[0]" />

<disconnect src-module="LSR6" src-gate="pppg[4]" />

<tell module="lifecycleController" target="LSR6" operation="NodeShutdownOperation"/>

</at>

</scenario>