1862

RIGA TECHNICAL
UNIVERSITY

Mārtiņš Mihaeljans

EVALUATION OF ADAPTIVE SOLUTIONS FOR MANAGEMENT OF VIRTUALIZED SOFTWARE-DEFINED NETWORKS

Doctoral Thesis

RTU Press
Riga 2025

RIGA TECHNICAL UNIVERSITY
Faculty of Computer Science, Information Technology and Energy
Institute of Photonics, Electronics and Telecommunications

Mārtiņš Mihaeljans
Doctoral Student of the Study Programme "Telecommunications"

EVALUATION OF ADAPTIVE SOLUTIONS FOR MANAGEMENT OF VIRTUALIZED SOFTWARE-DEFINED NETWORKS

Doctoral Thesis

Scientific supervisor
Professor Dr. sc. ing.
Jurģis Poriņš

Scientific consultant
Dr. sc. ing.
Andris Skrastiņš

Riga 2025

Mihaeljans, M. Evaluation of adaptive solutions for management of virtualized software-defined networks. Doctoral Thesis. Riga: RTU Press, 2025. 134 p.

Published in accordance with the decision of the Promotion Council “P-08” of 27 June 2025, Minutes No. 40.

DOCTORAL THESIS PROPOSED TO RIGA TECHNICAL UNIVERSITY FOR THE PROMOTION TO THE SCIENTIFIC DEGREE OF DOCTOR OF SCIENCE

To be granted the scientific degree of Doctor of Science (Ph. D.), the present Doctoral Thesis has been submitted for the defence at the open meeting of RTU Promotion Council on 14 November 2025, at the Faculty of Computer Science, Information Technology and Energy of Riga Technical University, 12 Āzenes Street, Room 201.

OFFICIAL REVIEWERS

Associate Professor Dr. sc. ing. Andis Supe,
Riga Technical University

Senior Researcher Dr. sc. comp. Atis Elsts,
Institute of Electronics and Computer Science, Latvia

Associate Professor Ph. D Oskars Java
Vidzeme University of Applied Sciences, Latvia

DECLARATION OF ACADEMIC INTEGRITY

I hereby declare that the Doctoral Thesis submitted for the review to Riga Technical University for the promotion to the scientific degree of Doctor of Science (Ph. D.) is my own. I confirm that this Doctoral Thesis had not been submitted to any other university for the promotion to a scientific degree.

Mārtiņš Mihaeljans ……………………………. (signature)
Date: ………………………

The Doctoral Thesis has been written in Latvian/English. It consists of an Introduction; 8 Chapters; Conclusion; 40 figures; 8 appendices; the total number of pages is 134, including appendices. The Bibliography contains 68 titles.

ANOTĀCIJA

Adaptīvo risinājumu novērtējums virtuāli realizēto programmatūras definētu tīklu valdībai – šis promocijas darbs – ir solis uz priekšu informācijas pasaules attīstībā. Šis darbs ir radies, izpētot vispārīgo tradicionālo tīklu principus, programmatūras definēto tīklu (SDN) un tā iespējas, lietu internetu (IoT) un tā ierobežoto (constrained) tīkla ierīču īpašības, un mūsdienīgo uz nolūku balstītu tīklu (IBN).

Promocijas darbs ir 8 pētījumu rezultātu apkopojums, kurā tika izstrādāti un novērtēti 7 adaptīvi risinājumi telekomunikāciju tehnoloģijām, kā arī, darba tapšanas brīdī, tas kalpo kā ieskats nozarē jaunajās un aktuālajās tehnoloģijās. Apspriestās problēmas ir tīkla ceļu ģenerēšana, sadrumstalotības mazināšana IoT tīklos, noteiktam mērķim būvēta tīkla aprīkojuma izmantošanas samazināšana, un automatizācija tīkla konfigurācijas izveidē un pārvaldībā.

Šajā promocijas darbā aplūkotie pētījumu rezultāti ir publicēti zinātniskajās publikācijās, kas pievienotas kā pielikumi darba pilnajam tekstam. Izvirzītie un novērtētie adaptīvie risinājumi ir neviendabīga servisa funkciju ceļa iekapsulēšana, reaktīva servisa ceļa atklāšana, servisa funkciju koplietošana, servisa funkcijas ceļa deklarēšana, ieteikumi IoT lietu specifikācijai, ieteikumi Thread režģtīkla blīvumam (savienojum skaitam starp iekārtām), un ieteikumi IBN mākslīgā intelekta uzdevumu vienkāršošanai, tādējādi veicinot IBN adaptāciju telekomunikāciju nozarē.

Šajā disertācijā izvirzītās hipotēzes ir balstītas uz autora veiktiem un publicētiem pētījumiem. Hipotēzes ir apstiprinājušās, jo adaptīvie risinājumi sniedz atbildes uz problēmu nostādnē izklāstītajām problēmām. Adaptīvo risinājumi savietojumam ar esošo globālā datortīkla infrastruktūru var būt nepieciešama to papildus modificēšana, atbilstoši vēlamajam darbības principam.

ABSTRACT

Evaluation of adaptive solutions for management of virtualized software-defined networks – this PhD thesis - is a step forward in the advancement of the information world. This research has originated from the exploration of general principles of conventional networking, software-defined networking (SDN) and its capabilities, the Internet of Things (IoT) with its constrained network device properties, and state-of-the-art intent-based networking (IBN).

The PhD thesis is a collection of 8 research results that not only contribute to the provision of 7 new adaptive solutions for telecommunication technologies, but also, at the time of writing, serve as insight into novel and actual technologies in the industry itself. Problems discussed are – network path generation, fragmentation mitigation in the IoT networks, cut on usage of specific-purpose built network equipment, and automation of human-managed network configuration.

Research results covered in this thesis have been published in scientific publications that are attached as appendices to the full-length text. The adaptive solutions proposed are: a service function path encapsulation used when necessary, a reactive service path discovery, service function share, service function path declaration, recommendations in specification for IoT things, recommendations for Thread mesh network density compliance, and recommendations for lift of IBN adaptation.

Hypotheses proposed in this thesis are all based on published research. Hypotheses have been proven to be true, as adaptive solutions do solve problems discussed in the problem statement. For compliance with existing global network infrastructure, the adaptive solutions may require modification according to the targeted working principle.

ACKNOWLEDGMENTS

Most importantly, I want to thank my family, who provided immense support, thus allowing me to focus on conducting this research and on the studies themself. It is all 10 years spent in the university together that counts here.

I must thank my mentor, Andris Skrastiņš. He not only took part in all aspects of conjuring the science, but also gave a significant impact and good example on how to approach and execute work in general.

I also get to thank my trainer, Alberts Bagojans. To be a part of the RTU team of athletes has been a positive experience throughout many years.

It is my supervisor Jurģis Poriņš and his RTU team who made these doctoral studies possible for me in the first place. I am grateful for that and the support that the university has put forward.

CONTENTS

Introduction 12
Background and topicality of the study 13
Problem statement 13
Objective and tasks of the study 14
Hypotheses 14
Authors Contributions 15
Scientific novelty 15
Practical significance 16
Publications and approbation of the Thesis 17
1. Summary of researched technologies 18
1.1. Software-defined network 18
1.2. Network function virtualization 23
1.3. Service function chaining 26
1.4. The Internet of Things 29
1.5. Thread mash-network and Matter protocol 31
1.6. Programmable protocol-independent packet processing 33
1.7. Segment routing with MPLS 35
1.8. Intent-based networking 36
2. Network topology-aware SF chaining 39
2.1. Expansion of the horizons 39
2.2. Design goals 40
2.3. Emulation of network topology-aware SF chaining 41
2.4. Results and conclusions 42
3. Reactive SF Path discovery 44
3.1. Rule-base update at runtime 44
3.2. Design goals 44
3.3. Emulations of reactive SF path discovery 46
3.4. Results and conclusions 47
4. SF share in inter-data center networks 50
4.1. Crossing the Bering Strait 50
4.2. Design goals 52
4.3. Emulation of multipath SFC 53
4.4. Results and conclusions 54
5. Segment routing with MPLS on reprogrammable data plane 55
5.1. Giving purpose to purposeless 55

5.2. Design goals 56
5.3. Emulation of SFC via SR-MPLS on P4 57
5.4. Results and conclusions 59
6. Analysis of the Internet of Things in action 60
6.1. A master key technology 60
6.2. Paradox of interconnectivity 61
6.3. Classification proposal 62
6.4. Structural placement 63
6.5. Conclusion 64
7. Thread mesh-network density considerations 65
7.1. Vertigo of Thread 65
7.2. Design goals 65
7.3. Simulation of the Thread network with various mesh densities 66
7.4. Results and conclusions 67
8. Analysis of Intent-based networking building blocks 68
8.1. Profiling of input 68
8.2. Translation of abstract 69
8.3. Resolution of policy 69
8.4. Activation of configuration 70
8.5. Assurance of everything 71
8.6. Lifting the burden 72
8.7. Conclusion 73
Conclusioner 74
References 78
Appendices 82
Appendix 1 83
Appendix 2 88
Appendix 3 93
Appendix 4 100
Appendix 5 107
Appendix 6 113
Appendix 7 118
Appendix 8 126

ABBREVIATIONS

Listed abbreviations are compliant with usage in thesis.

A

API – Application Programming Interface
AES-128 - Advanced Encryption Standard of 128 bits
AI - Artificial Intelligence
ALU - Arithmetical Logical Unit

B

BGP - Border Gateway Protocol
bmv2 - behavioral model version 2

C

CLI – Command Line Interface
CPU – Central Processing Unit
CVE - Common Vulnerabilities and Exposures

D

DAN - Data Aware Networking
DB - Database
DC – Data Center
DLP – Data Leak Prevention
DPI – Deep Packet Inspection

E

EAC – Event, Action, Condition
EGP - Exterior Gateway Protocol
EHF – Extra High Frequency
EIGRP - Enhanced Interior Gateway Routing Protocol

F

FEC – Forwarding Equivalence Class
FED – Full End Device (in context of Thread mesh network)
FIB – Forwarding Information Base
FW – Firewall

G

GAN – Generative Adversarial Network
GUI – Graphical User Interface

H

HF – High Frequency

I

IBN – Intent-Based Networking
IGP - Interior Gateway Protocol
IMT-2020 - International Mobile Telecommunications-2020 standard
INT - In-band Network Telemetry
IoT – the Internet of Things
IP – Internet Protocol
IPS – Intrusion Prevention System
IPv4 – IP version 4
IPv6 – IP version 6
ISO - International Organization of Standardization
IS-IS - Intermediate System-to-Intermediate System routing protocol
ITU - T - International Telecommunication Union Telecommunication Standardization Sector

L	
LAN – Local Area Network	
LB – Load Balancer	
LF – Low Frequency	

L
LAN – Local Area Network
LB – Load Balancer
LF – Low Frequency
M
MED – Minimal End Device (in context of Thread mesh network)
MF – Medium Frequency
ML – Machine Learning
MPLS - Multiprotocol Label Switching
N
NAT – Network Address Translation
NDO - Named Data Object
NF – Network function
NFV – Network Function Virtualization
NIC – Network Interface Card
NLP – Natural Language Processing
NSH – Network Service Header
O
OAM – Operation, Administration and Maintenance
ONOS - Open Network Operating System
OSI - Open System Interconnection
OSPF - Open Shortest Path First routing protocol
OVS – Open Virtual Switch
P
PAN – Personal Area Network
PBM – Policy-Based Management
PBNM – Policy-Based Network Management
PC – Personal Computer
PCEP - Path Computation Element Protocol
PDP – Policy Decision Point
PEP – Policy Execution Point
PIP – Policy Interpretation Point
PISA – Protocol-Independent Switch Architecture
PnP – Plug and Play
PoE - Power over Ethernet
POP - Points of Presence
PSA – Programmable Switch Architecture
P2P – Peer-to-peer communication
P4 – Programmable Protocol-independent Packet Processing
R
RAM – Random Access Memory
RFID – Radio Frequency Identifier
S
SDN – Software Defined Networking
SD-WAN – Software Defined Networking in Wide Area Network
SED – Sleepy End Device (in context of Thread mesh network)
SF – Service Function
SFF – Service Function Forwarder
SFC – Service Function Chaining
SHF – Super High Frequency

SID – Segment Identifier
SRGB – Segment Routing Global Block
SRLB – Segment Routing Local Block
SR-IPv6 – Segment Routing with IPv6 data plane
SR-MPLS – Segment Routing with MPLS data plane
SSH - Secure Shell
T
TE – Threat emulation/extraction (in context of network function virtualization)
TE – Traffic Engineering (in context of segment routing)
TLS - Transport Layer Security
TM - Traffic manager
TMF - Thread Management Framework
U
UE - User Equipment
UHF – Ultra High Frequency
V
VHF – Very High Frequency
VLAN – Virtual Local Area Network
VLF – Very Low Frequency
VoIP – Voice over IP
VXLAN - Virtual eXtensible LAN
W
WAN – Wide Area Network
WLAN - Wireless Local Area Network
WNAN – Wireless Neighborhood Area Network
WPAN - Wireless Personal Area Network
WWAN - Wireless Wide Area Network

INTRODUCTION

The General Assembly of the United Nations has a resolution of sustainable development goals where it lists man-made resource management and distribution mechanisms - power grid, water supply, food distribution, sanitary, military, monetary, and healthcare system – which are all locked in an information world's global network of physical and logical nodes. It's interconnected systems' operation and development maintains the well-being of all to the network attached ecosystems and among their residents [1].

Since the dawn of digitalization, a cardinal component of the information world is packet-switched networking. Communication, enabled by the network, allows its users to exchange information, goods, and services [2]. Evaluation of adaptive solutions for management of virtualized software-defined networks (this PhD thesis) is based on multiple research studies in packet-switched networks. These researches result in the development and evaluation of 7 new adaptive solutions.

Multihoming (a single target node is simultaneously reachable through multiple connections) gets enabled for service function chaining. This functionality allows for the usage of service function sharing for synchronous processing of multipath data flows.

An alternative to the usage of full encapsulation for the length of the service function path is introduced. This alternative method is the use of encapsulation only in the segments of the network where it is necessary.

Predefined service function path policy gets replaced with a mechanism for the automation of path discovery via the use of default path and network traffic re-classification. This functionality mitigates human intervention in network management, therefore, eliminating the possibility of having a network disruption that is a result of misconfiguration.

A recommendation is developed for combat of missing things' categorization in the Internet of Things. It allows grouping devices according to their physical attributes. Categorization enables one sort of things to have a joint attribute policy (reachability, identifiability, software support, energy consumption type), thus increasing the compatibility among them.

An analysis is conducted for Thread mesh density (Thread is a close-distance communication technology for IoT devices in small-sized personal area networks). The analysis results in a recommendation that allows easing the network availability issue caused by personal area link regeneration due to changes in network topology.

A programmable packet processing solution model is developed for service function chaining over segment routing with MPLS transport. This model identifies the relation between the segment identifier and MPLS label, and it proposes for service path description with the use of unified identification.

A working principles modification is proposed for an artificial intelligence and machine learning run state-of-the-art intent-based networking. Modification simplifies tasks that are delegated to the artificial intelligence by a split of management types.

This thesis is structured in 4 major parts as follows. First, an explanation of covered technologies is given. Second, the research done is outlined with illustrations tailored to the context of this thesis. Third, conclusions about the results of made work are drawn. Fourth, background research as appendices is attached to the thesis. The research and its results discussed in this thesis have been published in 8 scientific articles – all indexed in Scopus - to improve the operation of telecommunications and computer network infrastructure.

Background and topicality of the study

This research has originated from the exploration of general principles of conventional networking, software-defined networking (SDN) and its capabilities of service function chaining, the Internet of Things (IoT) with its constrained network device properties, and state-of-the-art intent-based networking. All mentioned technologies are pertinent to the communication and information technology field.

As the research base is a collection of work carried out from the fall of 2020 till the spring of 2025, it not only showcases the rapid development and innovation in the field itself, but also delivers solutions that are in synchronization with the telecommunication industry.

And even more so, it surpasses the commercial off-the-shelf product offer as research goes into a showcase of technology implementation that is not yet widely available. Such is programmable switch architecture (PSA). It's a technology intended for data packet processing customization according to individual requirements. On PSA, a service function chaining domain had been constructed and evaluated.

Problem statement

It is typical for the information world, also known as the digital medium, to be unpredictable and in constant change. The changes tend to result in various network problems. Some examples of these problems are – remotely located network node link disjoint due to unavailable network segments, live stream connection errors due to uneven dispersion of content load, voice signal jams due to low network quality, etc. All mentioned problems are directly addressable as their solution resides in the analysis of network nodes participating in the communication. Meanwhile, not all problems are directly addressable. Problems for which solutions can be achieved in various ways, and those that are not directly related to the misbehavior of a specific network entity, are discussed in this work:

1. Enhancement of network path generation and coordination of transmitted information in them with the use of service function chaining.
2. Fragmentation mitigation of the Internet of Things by the use of unified communication standards for the increase of compatibility among devices in use.
3. Cut on the use of task-specific or proprietary devices by advancement of the feature set for re-programmable units.
4. Automation of human-managed network configuration, maintenance, and telemetry analysis by provision of intent-based networking.

The mentioned problems are in the scope of the technology list given in the study's background and explored under section No. 1, Summary of researched technologies. With acknowledgment of existing problems study's tasks and hypotheses are defined.

Objective and tasks of the study

The study's objective is an analysis of existing solutions and the development of new adaptive solutions set for software-defined networking and complementary technologies in compliance with future network and IoT frameworks, including IMT-2020 non-radio aspects, defined in ITU-T Y recommendations. Studies scope is packet-switched networks in data centers, edge, access networks, and the Internet of Things. Solutions are made in a way to ensure that they are compatible with existing technologies and incorporate recommendations or standardization of the working principles of the mentioned technologies. Synchronization is also evident by the fact that the references of doctoral theses hold multiple recommendations and technology standards.

Defined tasks for the achievement of the set objective are as follows:

1. Perform analysis of software-defined networking and related technologies to define the problem statement.
2. Perform experiments and emulate working principles of software-defined networks and complementary technologies, and the Internet of Things, to give recommendations about proposed and evaluated solutions for the elimination of given problems.
3. From the output of analysis and experiments define a solution set in accordance with the intended telecommunication framework.
4. Evaluate the scenarios of enhancement possibilities of the developed solution according to technologies' evolvement trends, such as a spike in device count, a move towards packet forwarding, or a push for virtualization and automation.

Tasks define guidelines for actions performable on all in this work discussed problems. **The scientific method used in 6 of 8 works is the conduct of an experiment by network topology emulation. In 2 other works, the scientific method used is a literature analysis.**

Hypotheses

7 solutions are produced in this dissertation to answer the following hypotheses:

1. Reactive service function path discovery and coordination of transmitted information along multiple administrative domains is achievable via the use of partial-encapsulation and re-classification methods in service function chaining.
2. Fragmentation of the Internet of Things can be mitigated with the use of a unified transmission medium, and device compatibility can rise with the use of a unified application layer, if more specific characteristics of things are defined.
3. Service function chaining via segment routing can be built on top of programmable devices, thus serving as an alternative to the use of purpose-specific or proprietary devices with similar capabilities.
4. Computer network creation and management can be automated by the use of intent-based networking if artificial intelligence functionality is delegated separately to different network management types.

The first hypothesis holds in multiple studies, as service function chaining is the most explored technology in this work. This technology is mostly used in data center networks.

The third hypothesis also deals with service function chaining (SFC), but in a different way. Here, SFC is a subject, and the object is re-programmable devices. This refers to experiments with programmable switch architecture.

Authors Contributions

The PhD thesis is a collection of 8 research results that not only contribute to the provision of new adaptive solutions for telecommunication technologies, but also, at the time of writing, serve as insight into novel and actual technologies in the industry itself.

The most studied technology is service function chaining (SFC). In 5 of 8 studies, adaptive solutions are proposed for overcoming limitations of SFC technology. These solutions do not require their architecture modification as they are defined in a way of reimagining the use of existing elements.

Work also touches on well-studied but not well-rounded technology of the Internet of Things. A retrospective of both standardization and conducted studies revealed a causal relation between inadequately general recommendations and interoperability issues.

The strive for focus on the coverage of industry's actual research topics reveals itself most prominently through the more recent studies of programmable protocol-independent packet processing and intent-based networking technologies.

Scientific novelty

As a result of conducted research an applicable to software defined networking and related technologies, and the internet of things novelties have been produced:

- 2 new service function chaining packet encapsulation methods [3].
- New service function chaining path policy creation approach [4,5].
- New categorization concept for physical devices of the internet of things to overcome the existing device interoperability issue [6].
- New use case for devices within the service function domain for service of multipath data flows [7].
- New causal relation between mesh networks link count and networks regeneration occurrence ratio in Thread technology [8].
- New service function path declaration approach in segment routing [9], and programs for network devices of programmable switch architecture [10].
- 2 new intent-based networking models that are aimed at ease of tasks meant for artificial intelligence, with models customized in accordance with a manageable network [11].

Practical significance

PhD studies do not have a single definitive problem or single hypothesis to be proven. There is a multiple of them. Also, the tasks put in place are written to give guidance on how the study cases were approached. It is to be in such a way that the background of the research is each on its own standing scientific publication. Therefore, the research outcomes addressed within the context of this PhD thesis, as adaptive solutions, are as follows:

1. A service function path encapsulation according to the necessity for service function chaining, in opposition to encapsulation throughout.
2. A reactive service path discovery with the use of default path and data flow reclassification, in opposition to proactive (defined before the arrival of the ingress traffic classification event) manual path policy.
3. Service function share in inter-datacenter networks that enables the ability to expand the service function path's set of available network functions, or enables redundancy for existing network functions.
4. Service function path declaration with designated segment identifier in segment routing built on top of reprogrammable devices.
5. Recommendation on the implementation of specific characteristics for IoT things definition.
6. Recommendation on IoT Thread mesh network density (count of connections) compliance, to avoid regeneration of the whole network by defining the minimal required number of connections that must form between a routing-capable device and to it attached end-device.
7. Recommendation for modification of intent-based networking that envisions ease of tasks meant for artificial intelligence according to a manageable network.

All seven adaptive solutions are a valuable contribution to the information world as they are thought out and constructed in compliance with existing technology in use. Especially the service function chaining rooted solutions, which do not require modification in SFC domains standards.

Solutions that are expressed as recommendations carry their significance within the conducted studies context itself. For example, the IoT study of the 5th solution holds multiple paragraphs of arguments pointing out the flaws in existing standardization.

All aspects of the introduction section are revisited in later sections in detail and in overall summarized under the conclusions section.

Publications and approbation of the Thesis

Research results covered in this thesis have been published in scientific publications that are attached as appendices and listed below. Conferences where results were presented are also listed.

Scientific publications

1. M. Mihaeljans and A. Skrastins, "Network Topology-aware Service Function Chaining in Software Defined Network," 2020 28th Telecommunications Forum (TELFOR), Belgrade, Serbia, 2020, pp. 1-4, doi: 10.1109/TELFOR51502.2020.9306554.
2. M. Mihaeljans and A. Skrastins, "Reactive Service Function Path Discovery Approach in Software Defined Network," 2021 29th Telecommunications Forum (TELFOR), Belgrade, Serbia, 2021, pp. 1-4, doi: 10.1109/TELFOR52709.2021.9653356.
3. M. Mihaeljans and A. Skrastins, "Evaluation of reactive service function path discovery in symmetrical environment," Telfor Journal, vol. 14, no. 1, pp. 2-7, 2022, doi: 10.5937/telfor2201002M.
4. M. Mihaeljans and A. Skrastins, "IoT concept and SDN fusion in consumer products: Overview," 2023 3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), Tenerife, Canary Islands, Spain, 2023, pp. 1-6, doi: 10.1109/ICECCME57830.2023.10252518.
5. M. Mihaeljans and A. Skrastins, "Openthread Network Density Evaluation: Quantitative Analysis," 2023 Symposium on Internet of Things (SIoT), São Paulo, Brazil, 2023, pp. 1-5, doi: 10.1109/SIoT60039.2023.10390236.
6. M. Mihaeljans and A. Skrastins, "Efficient Multipath Service Function Chaining in Inter-Data Center Networks," 2023 31st Telecommunications Forum (TELFOR), Belgrade, Serbia, 2023, pp. 1-4, doi: 10.1109/TELFOR59449.2023.10372615.
7. M. Mihaeljans, A. Skrastins and J. Porins, "Parmounts of Intent-Based Networking," 28th International Conference ELECTRONICS 2024, Palanga, Lithuania, September, 2024
8. M. Mihaeljans, A. Skrastins and J. Porins, "Service Function Chaining via SR-MPLS over P4: A rudimentary analysis," The International Conference on Intelligent Computing, Communication, Networking and Services (ICCNS2024), Dubrovnik, Croatia, September, 2024

Result presentation in conferences

1. 28th Telecommunications Forum (TELFOR), Belgrade, Serbia, 2020
2. 29th Telecommunications Forum (TELFOR), Belgrade, Serbia, 2021
3. 3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), Tenerife, Canary Islands, Spain, 2023
4. Symposium on Internet of Things (SIoT), São Paulo, Brazil, 2023
5. 31st Telecommunications Forum (TELFOR), Belgrade, Serbia, 2023
6. 28th International Conference ELECTRONICS 2024, Palanga, Lithuania, September, 2024
7. The International Conference on Intelligent Computing, Communication, Networking and Services (ICCNS2024), Dubrovnik, Croatia, September, 2024

1. SUMMARY OF RESEARCHED TECHNOLOGIES

This chapter is a glimpse into the technologies that are discussed and with which experiments were carried out over the course of this thesis. As many of the adaptive solutions share the same technological background, a combined overview has been put up front covering, if not all, but close to most of the acronyms used later in the thesis. But make no mistake, information given ahead is extracted from coverage of adaptive solutions themselves and, for the most part, should have no repeat later. Therefore, it is suggestable for this section not to be skipped over.

1.1.Software-defined network

In a self-respectable manner, a section on software-defined networking (SDN) cannot start with SDN itself. A conventional network, aka legacy network, is shown in Figure 1. (created by the PhD author). A simplistic arrangement is illustrated where only key elements of interest are in color. The Network consists of three remote areas, each of which is directly connected to the cloud via a router.

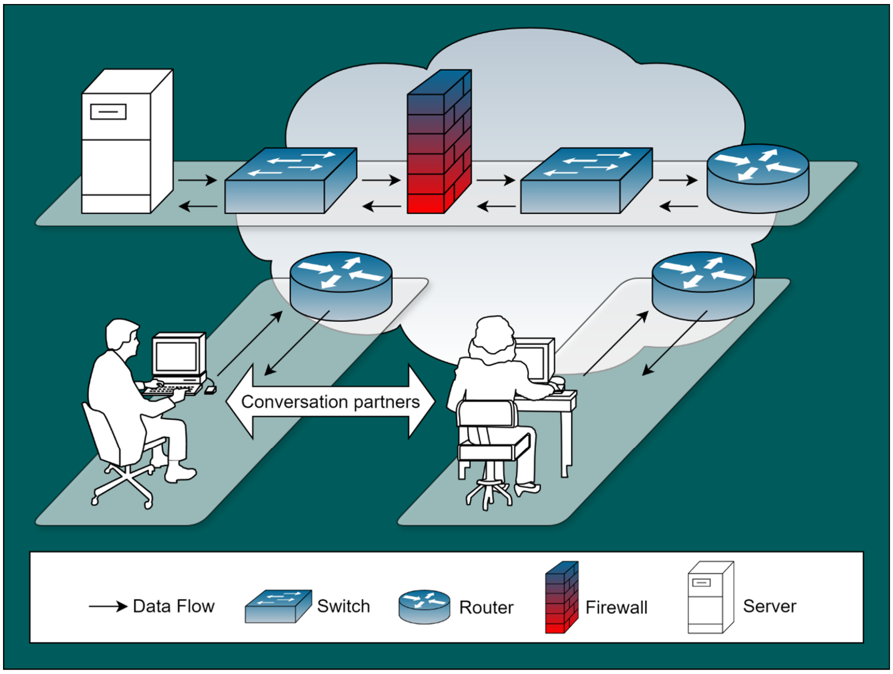


Fig. 1. Conventional network

While two plates are home to the network’s users who are engaged in a dialog, the third plate houses additional network equipment with which this dialog is facilitated. Surely all sorts of interconnection can be in place, but in its most basic form, network traffic can be represented by data flows. These flows traverse the cloud and reach all end-devices – computers and the server.

Colored network equipment in Figure 1. got its tones not by chance but with considerations. Switches and routers whose primary function is packet transportation are colored grey (network traffic forwarding device) and blue (control). Firewall is red (network traffic security device) and blue (control). The important thing to note is that each network equipment device also hosts local control of how it handles the network traffic that passes through.

In a nutshell, a network device is an entity that receives packets on its ports and performs one or more network functions (NF) on them. For example, the network device could forward a received packet, drop it, or alter the contents, and so on. A network device is an aggregation of multiple resources, such as ports, CPU, memory, and queues. Resources are either simple or can be aggregated to form complex structures that can be viewed as a single entity. The network device is in itself a complex resource. Examples of network devices include switches and routers. Additional examples include network elements that may operate at a layer above IP, such as firewalls, load balancers, and video transcoders, or below, like Layer 2 switches and optical or microwave network elements [12].

Conventional networking differs quite a lot from contemporary solutions. In contrast to conventional, aka legacy devices, where everything (transmission, control, management) was squashed into each network device itself, the Software-defined network (SDN) consists of three planes, each of which represents a different objective – application, control, forwarding, as shown in Figure 2. (created by the PhD author).

Originally, in 2014, the networking model was introduced in the ITU-T recommendation [13] with the fourth plane – external management, which would span across all other three, but it did not stand the test of time and ceased to exist. The three planes are just enough to fulfill SDN's purpose of dividing network control from packet forwarding.

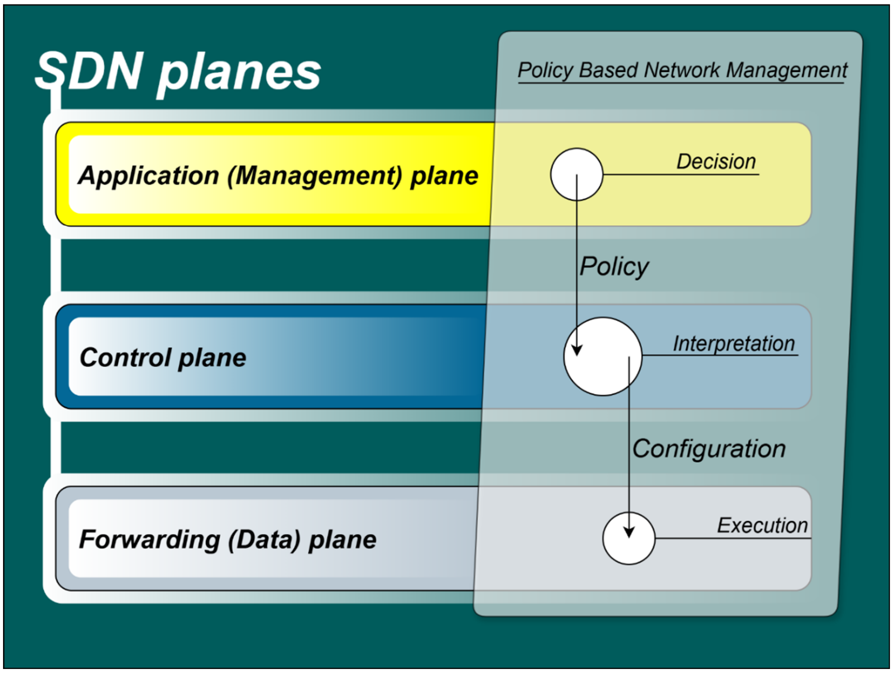


Fig. 2. Planes of Software Defined Networking

With a color scheme borrowed from figure 1. for forwarding plane (gray) and control plane (blue), figure 2. adds the application plane in yellow. Application in some literature, also referred to as the management plane, is the representation of an operator's (power-user who works for network existence) involvement with the network. Also depicted is the arrangement of policy-based network management (PBNM) elements – policy decision point (PDP), policy interpretation point (PIP), and policy execution point (PEP) placement in the SDN setup.

Policy-Based Management (PBM) is a management paradigm that separates the rules governing the behavior of a system from its functionality. It promises to reduce maintenance costs of information and communication systems while improving flexibility and runtime adaptability [14].

A quest for a secure data transmission is covered under SDN's umbrella. It is stated that SDN does provide new possibilities to combat security breaches. In a case of such cases, the affected resources may be easily and quickly isolated, malicious traffic may be safely terminated, sensitive flows can be identified and separately transferred in a more secure manner, by use of dedicated equipment and security protocols. All these processes may be automated due to SDN for improved availability. The logically centralized control of SDN enables operators to have a broader and in some cases even a global view of the current status of networks, which makes security operations easier and more efficient [13].

SDN planes

Let's start with the application plane. It provides human (network administrator, operator, or in the case of self-service even the end user, etc.) interaction with the network via a graphical user interface (GUI) or command line interface (CLI). This plane consists of software referred to as applications, facilitating the interaction. The interaction itself is a set of tasks that are man-imposed and machine-committed. A non-exhaustive list of these tasks is as follows:

1. Network policy creation – this policy defines how the network should operate. The policy does not limit itself to the definition of how the forwarding of data packets is carried out. It also regulates communication among network elements and their duties.
2. Network resource allocation – this allocation at the network initialization is done by a human, expressing the criteria for proper network data flow management, but later can be committed to automation.
3. Network monitoring – a feedback mechanism to provide information about the current network state, allowing for proper action to be taken when errors, disruptions, or even outages are faced. Monitoring usually consists of triggers, alerts, and indicators, all of which go off at the event of network state changes.

Mentioned tasks in the form of instructions are delegated to the control plane via application programming interface (API).

The control plane consists of network elements that are in place for the governance of the forwarding plane. These network elements, namely controllers, can be placed locally or remotely from the forwarding plane. Their main goal is to provide a centralized control of all forwarding plane elements and a single point of view of network state. There can be put in us a single controller or multiple ones in a cluster, or multiple controllers that are decentralized and oversee a network slice of their own. Tasks fulfilled by a network controller are as follows:

1. Enforce man-made instructions in the form of network equipment appropriate configuration onto network elements that reside in the forwarding plane.

2. Logically centralize network control for locally dispersed network elements.
3. Network resource abstraction for generalized reference in the application plane.

For communication between the controller and network elements residing in the forwarding plane, a special control channel and to it tailored protocol is used. This channel can be either outbound (a separate interface and means of communication are allocated for this communication only) or inbound (communication is in place via the same interface in use for end-user data flows). The most common protocol in use is OpenFlow.

The long-awaited explanation of what these forwarding plane network elements consist of is one simple word: switches. That's right, this plane consists mostly of the same kind of network equipment devices that perform data flow forwarding. On a side note, the statement is true only if we exclude a special kind of packet processing from this section, which will be covered under the network function virtualization section, and if we also exclude L2 and L3 routing processes, as their control also can be centralized via the use of SDN controllers.

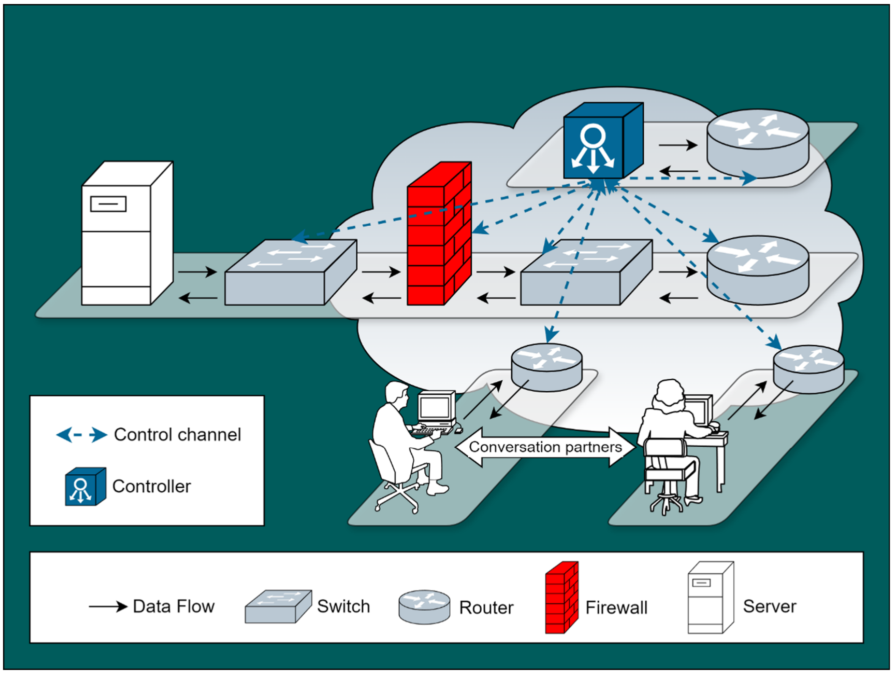


Fig. 3. Software-Defined Network

Shown in Figure 3. (created by the PhD author) is a software-defined network (SDN). Here control function (blue) is extracted from forwarding-capable (grey) and security (red) network equipment and delivered from a centralized controller remotely over a control channel.

In an SDN network, packets are not routed by local rules of International Organization of Standardization – Open System Interconnection Layer 3 (ISO OSI L3) routing protocols or L2 switched by media access control (MAC) addresses. They are forwarded according to the forwarding policy made up from the match-action pipeline shown in Figure 4. (created by the PhD author).

OpenFlow and semi-white switches

No introduction of SDN is complete without an explanation of the most common underlay standard – OpenFlow. Maintained by the Open Network Foundation, the OpenFlow itself not only represents a communication channel between controller and switches, but it also defines the target architecture of the switches themselves.

Shown in Figure 4. is the logical target architecture of semi-white switches. Mostly in the corresponding literature, these switches are referred to as white switches, but times have changed, and this architecture has been surpassed by reprogrammable network equipment (which can be considered as truly white) mentioned in later sections. White color is chosen due to its ability to customize data flow forwarding according to a match-action pipeline defined by the implementer and not bound to L3 and L2 rules. SDN originated for use in data center (DC) networks, where the original premise was – We know where the information is, all we need to do is reach it.

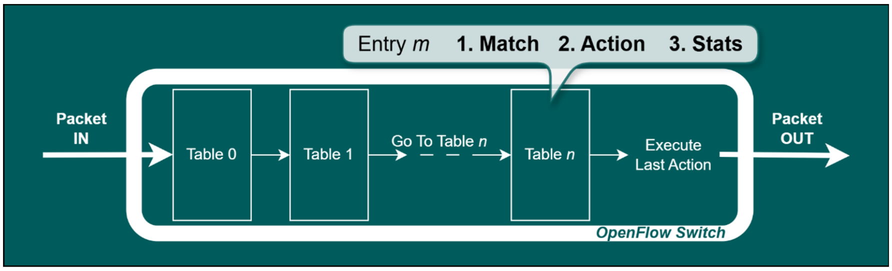


Fig. 4. Per-table packet processing in match-action pipeline

Shown in Figure 4. is the OpenFlow switch working principle. An incoming packet enters the match-action pipeline, which essentially is a list of tables. These tables consist of entries where each entry has three types of fields:

1. Match field – consists of a clause that defines what part of the packet needs to be a match with the entry to apply it. A match can be made against any packet header (differs among OpenFlow versions in use), incoming port, or other metadata about the packet itself.
2. Action field – tells the switch what to do with the packet, for example, forward it further in the network through some physical interface, send it to the controller, drop it, or give it to the next table for continuation of its processing there.
3. Stats (short for statistics) field – tracks entry usage – processed packet count, bit count, time how long entry is in table, etc.

Sending a packet to the controller signals a need for modification in the forwarding policy.

The packet processing process itself is quite simplistic. So, when processed by a flow table, the packet is matched against the flow entries of the flow table to select a flow entry. If a flow entry is found, the instruction set included in that flow entry is executed. These instructions may explicitly direct the packet to another flow table, where the same process is repeated. A flow entry can only direct a packet to a flow table number that is greater than its flow table number; in other words, pipeline processing can only go forward and not backward. Once the matching flow entry does not direct packets to another flow table, pipeline processing stops at this table. When pipeline processing stops, the packet is processed with its associated action set and usually forwarded [15].

1.2.Network function virtualization

If only data transfer across the network were as easy as sending and receiving actions. But it's not. Well, not in general. Of course, there are peer-to-peer (P2P) networks, but you can't really get far within those.

Wherever there's a solution, a problem must have existed prior. The complexity of the physical network is the culprit for a call for virtualization. It has been stated that network protocols tend to be defined in isolation, however, with each solving a specific problem and without the benefit of any fundamental abstractions. This has resulted in one of the primary limitations of today's networks: complexity. For example, to add or move any device, operators must touch multiple switches, routers, firewalls, Web authentication portals, etc., and update access lists (ACLs), virtual local area networks (VLANs), quality of services (QoS), and other protocol-based mechanisms using device-level management tools. In addition, network topology, vendor switch model, and software version all must be taken into account. Due to this complexity, modern-day networks are still relatively static as operators seek to minimize the risk of service disruption [16]. The Internet, for example, consists of many types of interconnected networks (public and private), facilitating a myriad of communication ways. It operates due to continuous development and use of all sorts of network functions (NF).

Network Functions

Network functions are all and every action performed upon transferred data except the transference itself (switching, routing, forwarding). A non-exhaustive list of network functions is as follows:

1. Firewall (FW) – a network function used to limit access between different parts of a network. There are many types of firewalls, but the most common are ISO-OSI L3 - limits access by use of Internet protocol (IP) addressing, and L7 - limits access by end-user application recognition via pattern match or other techniques.
2. Network address translation (NAT) – a network function that is used to mask original IP addressing with one that is targeted node or network acceptable. NAT also comes in different types to provide various solutions. For example, port address translation (PAT) is a type of NAT, where many translatable local IP addresses communicate with a remote network via the same source IP address. The trick here is the router remembering the source L4 port in use for the communication for the reverse packet to reach the appropriate communication party in the local network.
3. Load balancing (LB) – a network function aimed to provide sharing of the ingress network flows among multiple service providers. For example, in DC networks, LB is used to balance ingress traffic among multiple servers to ease the resource demands of a single server and increase the average throughput.
4. Deep packet inspection (DPI) – a network function that is used when there is a need to look into the contents of the packet's payload. In many cases, especially for web, for inspection to be possible, DPI must be able to decrypt the incoming data flow, as most of the network flows on the Internet are encrypted these days.
5. A proxy – a network function that is in use for services where none of the communication parties can manage the communication dialog. Similar to previous network functions, proxies also have different types, all of which are tailored to a specific service that is provided. One example might be a voice over IP (VoIP) proxy server that allows the end-users to make a voice call over the Internet.

6. Antispam – a network function that targets e-mail exchange for the elimination of context that is massively shared with multiple recipients at a scale that holds no regard for the content's actual value to each of the recipients individually. An example of spam is commercial advertising via the use of promotional e-mails.
7. Antivirus and antibot protection – a network function that leverages publicly available information of malicious actors, their current or last spotted online whereabouts, and tools in their arsenal to put up a blockage of harmful and illegal activity.
8. Intrusion prevention system (IPS) – a network function that is in place to stop unwanted and unauthorized access to resources that are not made public. IPS uses records of found week spots in software in the form of a common vulnerabilities and exposures (CVE) database (DB) to match the access request to a possible use case found in the DB, and if true, deny access to resources.
9. Data leak prevention (DPL) – a network function used for tracking of private information that is not supposed to be made available publicly or exposed to individuals operating from the outside of the network. At first glance, DPL might seem similar to IPS, but they are two completely different functions.
10. Threat emulation/extraction (TE) – a network function used to examine the data that is to be transmitted under a microscope, if we are speaking figuratively. Literally, it means that a user-like action is emulated upon the data in a sandbox, which reveals any potential harm that the recipient might face upon operation with the data.

The list of NFs is endless. The aspect that gives a focal point for the appropriate NF application is the determination of the underlay communication technology in use and the overlay service provided to the end-user. For example, in a household access network, a gateway from a local area network (LAN) to a wide area network (WAN) requires NAT, as private addresses are not routable over a public network.

Another mighty word that could ring a bell is service. Here it is an absolute necessity to distinguish a difference among the following two. The Network Service contributes to the behavior of the higher-layer service, which is characterized by at least performance, dependability, and security specifications. The end-to-end network service behavior is the result of the combination of the individual network function behaviors as well as the behaviors of the network infrastructure composition mechanism [17]. So, to conclude the thought, Internet service provider's (ISP) provided end-to-end network connectivity is an end-user-oriented service that consists of many, many network services in the underlay. But the underlay is made up of countless network functions.

Virtualization

All functions are equal, but some functions are more equal than others. While NAT is a sort of logical function and does not require a physical device built for its performance, others, like a firewall, do. The difference lies in resource consumption. In the old days, NFs were mostly carried out on purpose-specific network equipment that, all in all, was proprietary and closely tied to vendor-specific technologies.

A turning point in DC maturity was the introduction of general-purpose hardware. It's a hardware designed to accommodate virtualized services on top of a shared central processing unit (CPU), random access memory (RAM), storage, and other resources. Show in Figure 5. (created by the PhD author) is the network function virtualization (NFV). In NFV, for specific-purpose built network equipment is replaced by general-purpose devices on which the NFs are emulated.

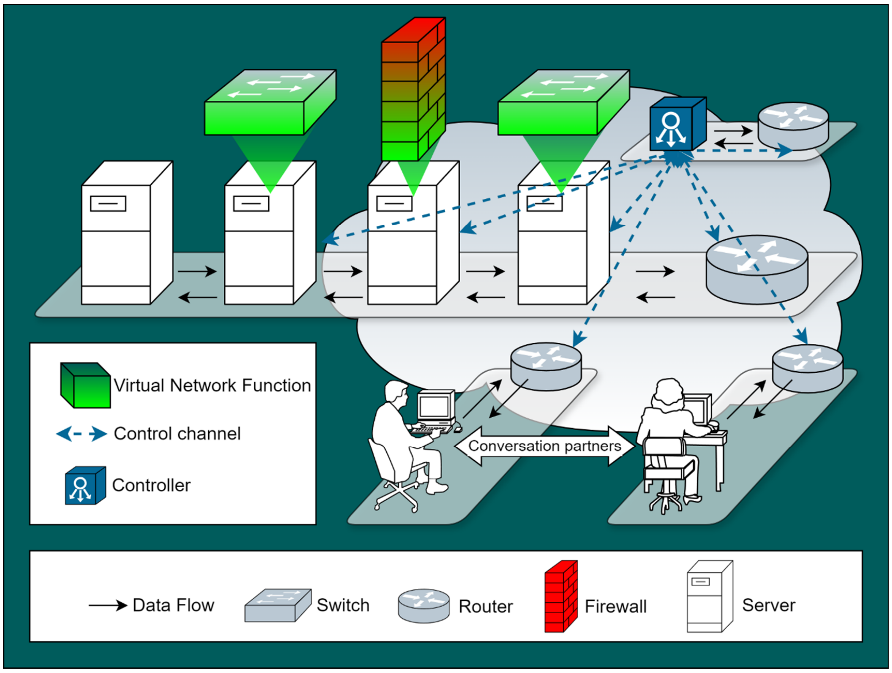


Fig. 5. Network Function Virtualization

Network Function Virtualization (NFV) shown in Figure 5. depicts a similar network topology previously illustrated in figures 1. and 3. The central plate, which can constitute for role of a data center, has its local network functions virtualized (light green) on physical servers. In such a configuration, data flows that previously were forwarded by physical network equipment now traverse physical servers instead. Their processing is facilitated by services that are internal to these physical servers. These services perform the functionality of network functions. Even though virtualized, network functions (NF) can use a control. This controller would both manage network functions and monitor physical server stats to an extent required for assurance of NF workflow. Surprisingly, NFV can be considered as a problem too. One of the long-standing problems of the industry has been silos, separate areas with separate expertise, separate staffing, separate business, service, etc. A non-exhaustive list of examples is as follows:

1. Transport vs packet.
2. Wireline vs wireless.
3. Local vs long-haul (access vs core).
4. Data center, cloud vs dedicated and dispersed physical elements.
5. Network functions virtualization (NFV) vs SDN.

There will continue to be important differences along these and other dimensions. However, SDN ought never provide a technical justification for the perpetuation or creation of silos. The goal of the architecture is to open possibilities, and especially to expose common ground, such that silos can be collapsed whenever and however it makes sense [18]. So, in regard to NFV, virtualization of the physical network creates a group of network devices that is logically isolated to an extent from other network devices that may or may not be a part of a separate virtualization. Silos of virtualization are created.

1.3.Service function chaining

Service function chaining (SFC) is a technology explored throughout almost all the practical research works of this thesis. SFC is a network data flow steering method that is used to modify the packet forwarding path, altering the packet processing without making changes in the underlay transport topology. In other words, the path is templated for the service that is to be provided. In the context of SFC, network functions (NFs) are referred to as service functions (SFs).

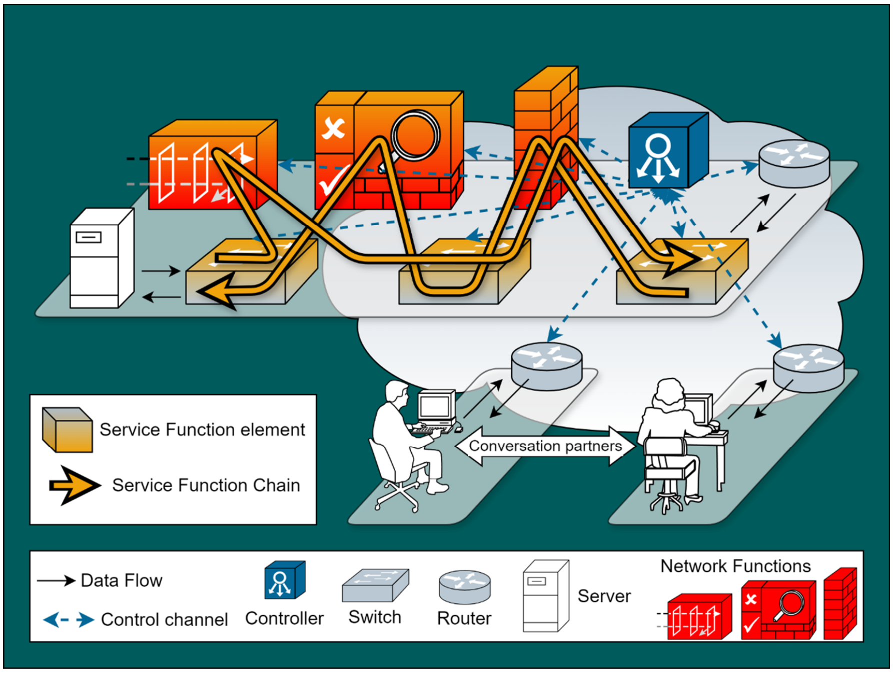


Fig. 6. Service Function Chaining

Network Functions (NFs) have gained a new orange tint in Figure 6. (created by the PhD author) as they are a part of Service Function Chaining (SFC), therefore, they can be referred to as Service Functions (SFs). At first glance concept behind the orange arrows might seem hard to grasp, but in essence, what they do represent is no longer a point-to-point (P2P) connection, but a path taken by the same data within an SFC domain. Mentioned data is the same data flows coming to and from end users as they communicate via the server, but within the SFC domain, these data flows are encapsulated for enablement of alteration in their processing.

The alternate processing reveals itself in Figure 6. where the reverse path from the server traverses a different set of service functions than one that is directed towards the server. If examined further, it is an interesting case where, for one of the SFs a stateful packet processing is possible, while for the two others, only a single direction of communication dialog is visible, therefore making only stateless processing available.

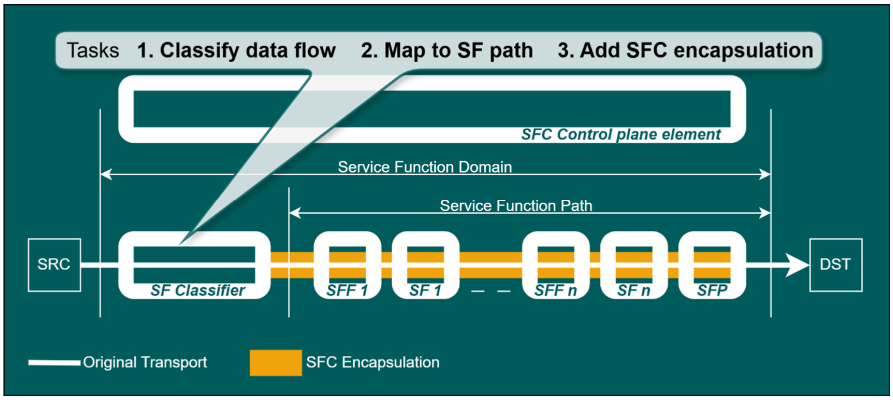


Fig. 7. Service Function Chaining Domain

SFC domain is a collection of network nodes under a single administration, and it consists of SFC elements that are service function (SF) encapsulation aware. Show in Figure 7. (created by the PhD author) is the SFC domain and its elements as follows:

1. Service function (SF) – a network function that is service function encapsulation aware. In other words, this network function makes use of and has visibility over the encapsulation imposed on the data packets of the original flow.
2. SF forwarder (SFF) – an intermediate element placed in the SFC domain among all other elements (service functions, classifiers, and proxies) with the sole purpose of data packet forwarding along their specific service function path.
3. SF classifier (CL) – an element that is placed at the ingress for application of encapsulation onto incoming data flows in accordance with the required SF path.
4. SF proxy (SFP) – an element that is placed at the egress of the SFC domain for the removal of encapsulation for packet forwarding towards its destination. SF proxies can also be placed between the SFC domain and a network function (NF) that does not support the encapsulation.
5. SF encapsulation – is a data packet header commonly placed between L2 and L3 headers for alteration of packet processing within the SFC domain. This encapsulation should provide enough information for all SFC elements to properly steer the data flow along the SF path. There can be various protocols used for this task, some of which are Multiprotocol Label Switching (MPLS), Network Service Header (NSH), Virtual Local Area Network (VLAN), etc.
6. SF path – is an actual forwarding path which data flow took while being steered through the SFC domain. The SF path begins at the ingress SF classifier and ends with the egress SF proxy. The SF path's length is limited to either the constraints of the service provided for the end-user or the SF encapsulation itself.
7. SF chain – is the required logical path of data flow that is described in the network flow classification policy. The SF chain describes which SFs need to be visited and in what order, but does not define specific network elements to steer data flow to. For example, if the SFC domain has two equivalent firewalls (FW) and the SF chain consists of a requirement to visit any firewall, then it is the SF path where the specific FW visited is listed.
8. SF control element (CTRL) – is a network element that handles operational, administrative, and management (OAM) operations for all SFC domain elements. It is the control element from which network-administrator-imposed SF chaining policies are distributed throughout the network.

9. SF control channel (C*n*) – is a communication channel managed by the SF control element and used for communication between SFC elements.

The encapsulation used should be able to represent the network policy in use. Therefore, regardless of the source, the metadata it contains reflects the result of classification. The granularity of classification may vary. For example, a network switch, acting as a Classifier, might only be able to classify based on a 2-tuple or based on a 5-tuple, while a service function may be able to inspect application information. Regardless of granularity, the classification information can be represented in the network service header (NSH) [19].

The original SF encapsulation

A protocol proposed by telecommunication conglomerate Cisco – Network Service Header (NSH) is an add-on to SFC. It is shown in Figure 8. (created by the PhD author) NSH has three types, where one version of protocol has no context value, another one has a fixed-length context value, but the third type has a variable-length context value. Context value is information about the service provided in the form of metadata. Aside from that, all three NSH types hold a service path identifier value and a service index value.

Similar to Multiprotocol Label Switching (MPLS), NSH is also positioned between Ethernet and IP headers. Around 2012, it seemed that NSH was superior to MPLS, as from any point in the network, a packet processing network device could track where in the SF path the packet is. But this has proven to be false with the rise of Segment Routing with MPLS (SR-MPLS), which provided equivalent clarity by utilization of a label stack rather than a single label. If NSH is used to realize service routing, it can carry policy information from an uplink classifier (CL) to a downlink CL. In this case, no interface between the policy control and the downlink CL is needed. If SDN is used to realize the service routing, an interface between the policy control and the downlink CL is needed, which allows the downlink CL to obtain policy for downstream traffic [20].

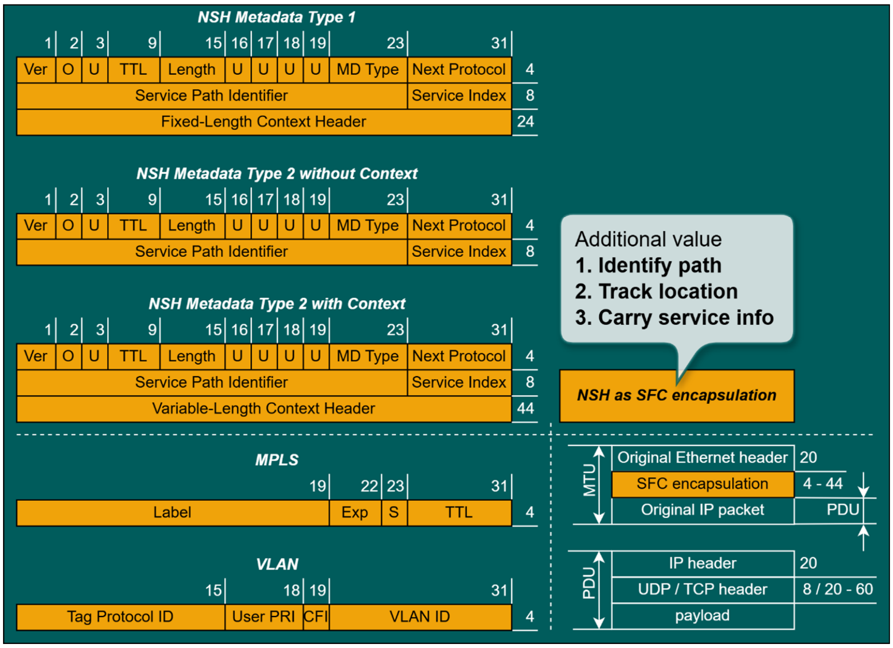

NSH Metadata Type 1
1 2 3 9 15 16 17 18 19 23 31
Ver O U TTL Length U U U U MD Type Next Protocol 4
Service Path Identifier Service Index 8
Fixed-Length Context Header 24
NSH Metadata Type 2 without Context
1 2 3 9 15 16 17 18 19 23 31
Ver O U TTL Length U U U U MD Type Next Protocol 4
Service Path Identifier Service Index 8
NSH Metadata Type 2 with Context
1 2 3 9 15 16 17 18 19 23 31
Ver O U TTL Length U U U U MD Type Next Protocol 4
Service Path Identifier Service Index 8
Variable-Length Context Header 44
Additional value
1. Identify path
2. Track location
3. Carry service info
NSH as SFC encapsulation
MPLS
19 22 23 31
Label Exp S TTL 4
VLAN
15 18 19 31
Tag Protocol ID User PRI CFI VLAN ID 4
MTU
Original Ethernet header 20
SFC encapsulation 4 - 44
Original IP packet PDU
PDU
IP header 20
UDP / TCP header 8 / 20 - 60
payload

Fig. 8. Service Function Chaining Encapsulation

1.4. The Internet of Things

Although there is a common misconception out there that nobody can definitively clarify what exactly the Internet of Things (IoT) is, the secret to understanding it lies in the granularity of the explanation that is to be expected. A brilliant warm-up case can be a device called a sensor. If focusing on electrical sensors out of all other sensors, an Electrical Sensor device measures the electrical power and/or energy being imported and/or exported [21].

Similarly, IoT can be reasoned. The global network is nothing more than a set of interconnected devices. In this network among human-operated devices, there are also those devices, like actuators and sensors, and specific purpose-built equipment that do not necessarily require human-assisted input for their task accomplishment. Being autonomous and connected to the same global infrastructure of interconnected devices, they form a unity that can be referred to as the Internet of Things (IoT).

And like all technologies, IoT brings a problem with it. The Internet must be able to accommodate IoT devices, as the decreasing cost of devices and increasing number of application demands are the main factors driving the increased number of deployed devices. The decreasing cost of devices allows new use cases, which are not profitable at a higher device cost, to be deployed commercially. There is a clear trend that the number of deployed devices continues to increase. Therefore, the network ultimately needs to support a large number of devices [22].

IoT characteristics

On the surface, it might seem that everything connected to the Internet is an IoT device, but there is more than meets the eye. An IoT device must fit in the design shown in Figure 9. (created by the PhD author).

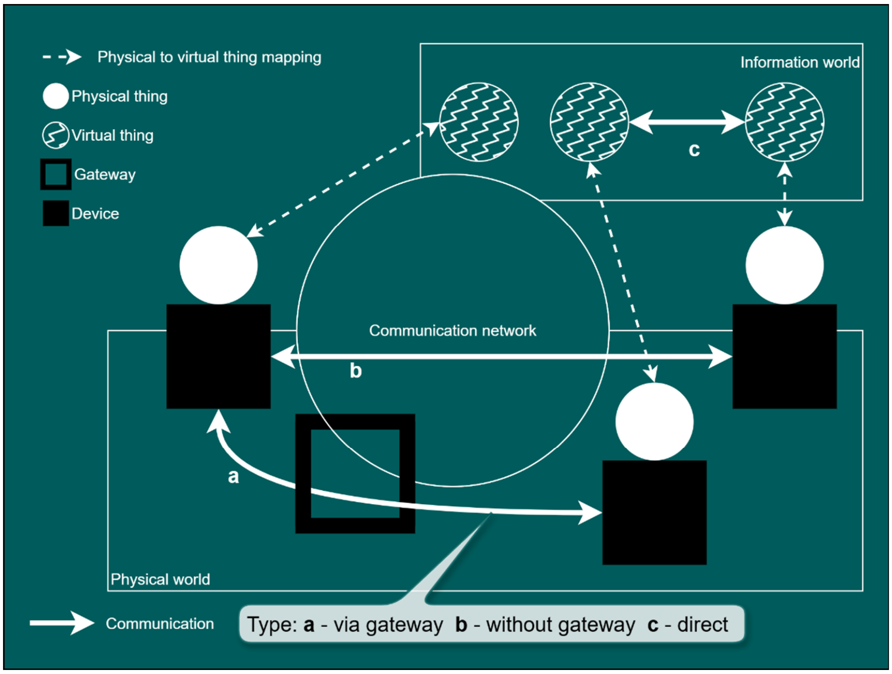


Fig. 9. Design of the Internet of Things

Notice in Figure 9. a device is represented as a black box, as it can be anything, starting from home appliances to industrial or environmental objects. An interchangeable term to physical-world and information-world within the context of IoT are real-world and digital-world. Figure 9. shows also how the IoT devices can communicate with one another and their placement in the network.

The communication ways shown are as follows:

- Via gateway and across the network – in this case, an intermediate device facilitates the connection over the transmission medium provided by the network. For example, a dialog over a wide area network (WAN).
- Without a gateway and across the network – in this case, communication peers speak directly but still use the transmission medium provided by the network. For example, a dialog between devices in child-parent relationships within a personal area network (PAN).
- Directly – in this case, communication peers have a direct communication medium that is in no part related to the communication network. For example, a dialog between two applications using a localhost connection.

These communication ways can be used for both unidirectional and bidirectional communications among physical and digital devices, aka things. To enable either or all three communication ways, each thing in use must tick all the following properties:

- Identification – a thing needs to be identifiable within the network, therefore, it must hold a unique identifier attached to it either physically or digitally.
- Location – a thing needs to be traceable within the network, therefore, it must be mapped either directly via its positioning or indirectly via its parenting structure relation.
- Reachability – a thing needs to be reachable over the communication network. Meaning, things task's fulfillment of actuation or sensing can be triggered by access from a communication network via the use of previous properties of identification and location.

Typically, things with embedded sensors observe physical environments and acquire information about their surroundings. Based on this information, some devices are actuated (actuators) and the physical surroundings can be controlled. Some applications, such as radio frequency identifiers (RFID) applications, for example, use data exchanges between things. In this type of application, data that the thing acquires from outside and/or holds inside are essential to provide the IoT services [23].

Constrained devices

A personal computer (PC) does not constitute to being a constrained device. Even a laptop or smartphone wouldn't count. As these are exceptionally capable devices built to produce maximum performance with regard to having unlimited availability to resources. Requirements for a device to be considered as a constrained device are that it has a limit on one or more parameters from the following:

- Data transmission – The transmission can be limited by the direction, meaning the device is only capable of transmitting data or receiving it, or doing both one at a time. Another aspect is the characteristics of transmission itself. Maximum bandwidth, range of reach, and supported transmission medium.
- Processing power – commonly, autonomous devices are built to last on their own for a significant amount of time, like a period from three months to a year to a couple of years. Therefore, by design, they are equipped with less energy-consuming hardware. That includes the chipset. Central processing units (CPU)

are clocked for maximum efficiency, so the amount of data that can be processed at a given time is far less than on a regular non-constrained device.

- Energy – Disconnection from a constant source of energy, such as a power grid, is to be faced when equipment must be placed in rural areas, or its working state requires it to be in constant motion by itself. This is where an alternative power supply comes in play. It can be a battery or a solar panel, etc. Their produced energy is a limited resource. Therefore, it must be spent optimally. One example of a power economy is putting device transceivers to sleep while they are not in use. Another is putting the whole device to sleep during periods of inactivity.

1.5.Thread mash-network and Matter protocol

An alliance of tech industry leading manufacturers has tackled the problem of IoT device interoperability's inexistence. For starters, it is not that manufacturers themselves produced incapable devices or commercialized undeveloped products. The design of all things sold at the local marketplace is well-rounded and heavily certified to be highest of quality and value for the consumer. However, once these things are put on the store shelves, another attribute comes into play. Which is the price. So, the actual need for interoperability comes from the buyers themselves, as they wish to cut the best deal possible by taking a mix of available goods. Once the product bucket is brought home, the end user faces another existential dilemma of how to connect the things in a way where a single device could take the role of being a universal remote for others.

The answer to both issues – the interoperability at the physical layer and the interoperability at the application layer is a two-fold solution. A separate communication technology named Thread and an application translation layer named Matter bridges the interoperability gap. Additionally, one can work without the other, therefore allowing older devices to use only the functionality that they can lift.

Matter uses a common application layer and data model that delivers interoperability between devices, allowing them to communicate with each other across multiple IP network technologies. Since its launch, Matter runs on Wi-Fi, Thread, and Ethernet network layers and uses Bluetooth Low Energy for commissioning [24].

Thread

Thread technology, similarly to Wireless Fidelity (Wi-Fi), uses a personal area network (PAN) as its playground at a 2.4GHz band, but instead of the star topology used in Wi-Fi, Thread uses a two-tier mesh. Mash allows for escaping single-point-of-failure scenarios.

Thread makes full use of the ability of the MAC layer to provide addressing based on short addresses (16-bit length) to further reduce the information bits needed to be sent over-the-air to provide efficient packet forwarding. This saves processing cycles and improves power consumption at the same time while still using an IP-based routing protocol [25].

Under the two-tier preposition lies the parent-child relationship model usage. A sort of hierarchy among devices is required to introduce graduation in expected functionality, allowing for less capable devices to do bare minimal in relation to communication, but still be a part of the PAN. Thread devices are ordered as follows:

- Full end device (FED) – Maintains IP address mapping and routing alongside its functions as an IoT thing.

- Minimal end device (MED) – Only communicates with its parent device and does not maintain IP address mapping alongside its functions as an IoT thing.
- Sleepy end device (SED) – Alongside its functions as an IoT thing, it communicates similarly to MED but also preserves energy by switching off its radio during the idle periods.

There is a more granular division between these devices, for example, not all devices that are routing-capable function as routers. A single router is elected to be the leader of the PAN and so on. The important thing to notice is that the structural complexity of this network allows for variation in physical device capabilities, enabling weaker by specs devices to be reachable via more capable devices.

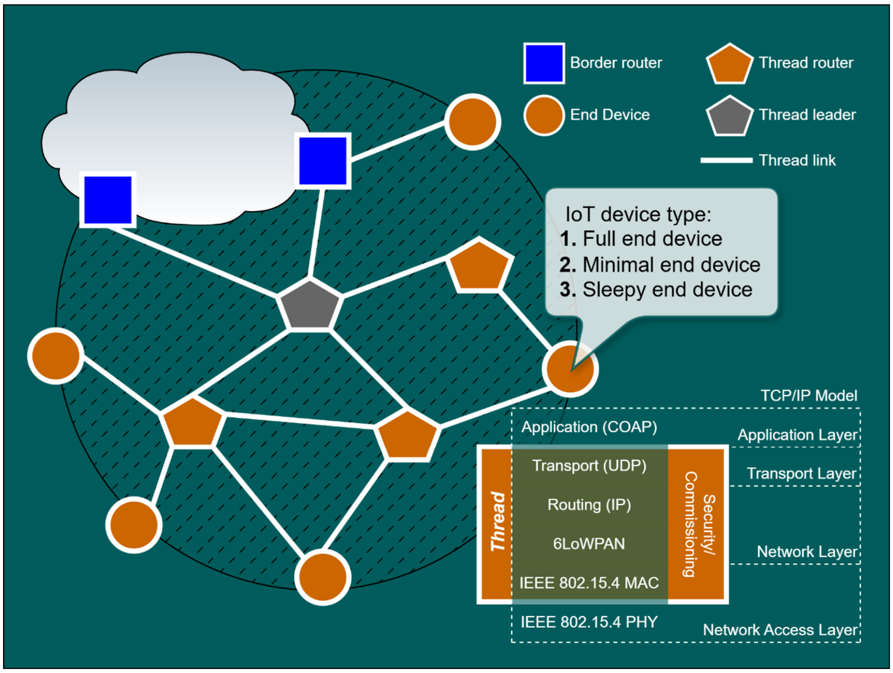


Fig. 10. Thread personal area network

Personal area network (PAN) created with the use of Thread and Thread's placement in TCP/IP model is shown in Figure 10. (created by the PhD author). In this figure Thread network is connected to the cloud via the use of a border router. Even though Thread is an IP-based communication technology, it is meant to be used in local networks.

Security-wise, Thread uses versions of the same types of security technologies that make applications across the Internet safe, the ones that keep financial transactions, purchases, and social media sessions secure. These security technologies have been adapted to the unique situation of consumer devices that must automatically maintain network security with little or no user interaction. Thread uses the advanced encryption standard of 128 bits (AES-128) encryption, which closes holes that are present in other networking protocols. Although Zigbee and Z-Wave also use AES-128 encryption for data communication, Thread was designed to always require all data frames to have security enabled [26].

Matter

In general, the goal of Matter is to allow two or more devices from different vendors to exchange their data and control information. This exchange not only allows for simplifying users' experience while interacting with them, but also greatly increases automation capabilities of such concepts as smart homes, digital assistants, and others.

Matter treats networks as shared resources. It makes no stipulation of exclusive network ownership or access. As a result, it is possible to overlay multiple Matter networks over the same set of constituent IP networks [27].

Matter protocol sits at the application layer of the ISO-OSI model and consists of the following structure:

- Application Layer – actual functionality of the thing (light, climate control, etc.).
- Data model – translates functionality into commonly understandable data.
- Interaction model – specifies client-server relations for operation on interoperable data.
- Action framing – constructs an action from an interaction formed in upper layers.
- Security – transferable information is secured via encryption.
- Message framing and routing – specifies packet payload fields and routing information.
- IP framing and transparent management – delivers a constructed message to the underlay transport network for it to be sent across the network.

The capability of each Matter-compatible device is tightly locked in a database of descriptors that define what clickable actions are, how many buttons a TV remote has, and so on. For example, one of such descriptors is a cluster that provides an interface for managing low power mode on a device that supports the Wake On LAN or Wake On Wireless LAN (WLAN) protocol. This cluster would be supported on IP devices that have a low power mode and support the ability to be woken up using the Wake on LAN or Wake on WLAN protocol. This cluster provides the device MAC address, which is a required input to the Wake on LAN protocol. Besides the MAC address, this cluster provides an optional link-local IPv6 address, which is useful to support "Wake on Direct Packet" used by some Ethernet and Wi-Fi devices [28].

1.6. Programmable protocol-independent packet processing

Programmable protocol-independent packet processing in short P4 is a concept of enabling re-programmability capability on network equipment devices. It can be a device such as a multiport switch and even a network interface card (NIC) by itself. P4 language is used on network devices whose architecture corresponds to that of a Protocol-independent switch architecture (PISA). Together, P4 and PISA form a framework for white-switches and network functions (NF) that can be run on top of them.

P4 is also a language for expressing how packets are processed by the data plane of a programmable forwarding element such as a hardware or software switch, network interface card, router, or network appliance.

In general, P4 programs are not expected to be portable across different architectures. For example, executing a P4 program that broadcasts packets by writing into a custom control register will not function correctly on a target that does not have the control register. However, P4 programs written for a given architecture should be portable across all targets that faithfully implement the corresponding model, provided there are sufficient resources [29].

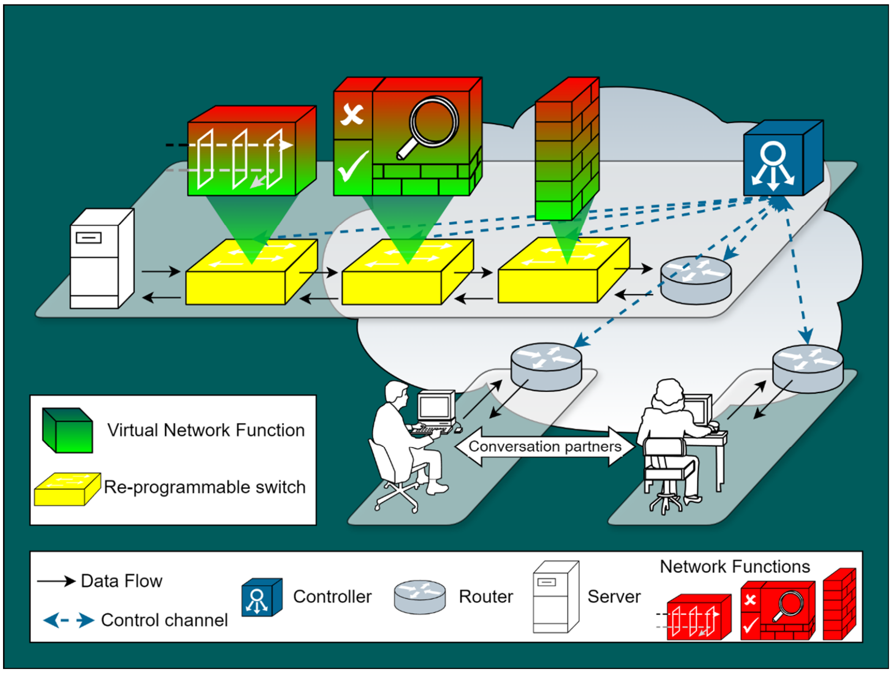


Fig. 11. Programmable protocol-independent packet processing

Shown in Figure 11. (created by the PhD author) is the use of re-programmable switches. These switches are in yellow, which is supposed to represent direct influence on packet processing from the SDN management layer. This new functionality allows for network functions to be executed on the switches themselves, creating a sort of virtualization solution.

PISA architecture

Protocol-independent switch architecture (PISA) is a switch architecture that allows for surpassing semi-white, for example, open virtual switch (OVS) capabilities by allowing the user to programmatically add functionality to it. Another prism to look at it is through the eyes of the user. Proprietary devices have a bottom-up capability provision, while P4 devices have a top-down capability provision. The difference between them is that the first has built-in functionality that can only be upgraded through the vendor's update cycle, but in the second, functionality can be adjusted, modified, or even remodeled on the user's demand.

A problem that arises in such modularity is the need for an explanation of custom implementations. While a P4 program does provide a precise description of the data plane behavior, and this can prove invaluable in writing correct control plane software, in some cases, it is enough for a control plane software developer to have the control plane API, plus good documentation of the data plane behavior. Some device vendors may wish to keep their P4 source code private. The minimum requirement for the controller and device to communicate properly is a P4Info file that can be loaded by a controller in order to render the correct P4Runtime API [30].

1.7.Segment routing with MPLS

When using conventional routing strategies, it is the routers that determine packet routing paths via the use of routing protocols. When using source routing, it is the source host that determines the necessary packet route and attaches it to the packet upon sending it towards its destination, therefore, routers do not make routing decisions but simply forward the packet according to the information that came with it. Segment routing is a packet routing method that combines source routing with packet encapsulation. In segment routing source host or the first router in the path pushes an encapsulation on top of the original packet. This encapsulation is used by transit routers to make decisions on how to route the packet.

In general, packet headers contain considerably more information than is needed simply to choose the next hop. MPLS, however, is designed with this point in mind. Choosing the next hop can therefore be thought of as the composition of two functions:

1. A function that partitions the entire set of possible packets into a set of forwarding equivalence classes (FECs).
2. A function that maps each FEC to a next hop.

Insofar as the forwarding decision is concerned, different packets that get mapped into the same FEC are indistinguishable. All packets that belong to a particular FEC and which travel from a particular node will follow the same path (or if certain kinds of multi-path routing are in use, they will all follow one of a set of paths associated with the FEC) [31].

In segment routing, the encapsulation contains a finite-length number that represents a segment. A segment is a descriptor of a network, node, or instruction. If MPLS is used as a data-plane encapsulation, an MPLS label fills the role of the descriptor. There are three types of control possible for SR-MPLS, which are:

1. Decentralized – by use of modified versions of link state routing protocols like Open Shortest Path First (OSPF) or Intermediate System-to-Intermediate System (IS-IS), or Border Gateway Protocol (BGP), MPLS labels are distributed. While OSPF and IS-IS are Interior Gateway Protocols (IGP) and route network traffic among routers within a single autonomous system (AS), BGP is an Exterior Gateway Protocol (EGP) and is used to route network traffic among multiple autonomous systems.
2. Centralized – in software-defined networks, a network controller oversees MPLS label distribution. To communicate with network nodes controller can utilize the Path Computation Element Protocol (PCEP).
3. Hybrid – a combination of decentralized and centralized control. For example, centralized control over remotely separated decentralized networks.

Whichever control mechanism is in use, each routing-capable network node must maintain its own forwarding information base (FIB). In a case of having two or more available paths to a destination, as in all link-state routing solutions, the chosen one will be the path with the lowest link cost. In case of collision of incoming labels, the situation is resolved via the use of tiebreaking rules and not first-come, first-served.

One MPLS label equals one segment identifier (SID). SID identifiers are allocated from the segment routing global block (SRGB) for packets that must travel along the network and the segment routing local block (SRLB) for local use within a routing instance. There are three types of SIDs:

1. Network SID – an identifier that represents a network subnet towards which to forward network flow.
2. Node SID – an identifier that represents a target node that should receive the network flow.
3. Adjacency SID – an identifier that is used for packet path specification to steer the network flow forwarding in a more gradual way than via the use of network SID.

An alternative to MPLS is IPv6. The difference between both is that MPLS does not have to be modified anyhow for segment routing support, while IPv6 uses new headers.

Segment Routing enables Service Providers to support the realization of Network Slicing in an IP/MPLS transport network. The network as a whole, in a distributed and entirely automated manner, can share a single infrastructure resource along multiple virtual services (slices). For example, one network slice is optimized continuously for low-cost transport, but a second one is optimized continuously for low-latency transport, a third one is orchestrated to support disjoint services, etc. The optimization objective of each of these slices is programmable by the operator [32].

1.8.Intent-based networking

Intent-based networking (IBN) has been introduced to the Society of Science twice. And still, it is in its infancy state. First time around, intents were used as direct references of executable configuration in Open Network Operating System (ONOS), playing the role of forwarding path descriptors. This time, intent has evolved into an abstraction of the user's (herein technical staff not end-user) request towards network operation. Quite a leap, but it might be something too far-fetched for available underlay technologies to be able to provide.

Intent as an abstract

Intents are users' requests, out of which a network policy can be generated. They are fed into IBN as abstractions, as the formation of a clear vision on how things need to be done by the user himself would defeat the whole purpose of IBN. It is taken as an exit point that the user only contributes to the matter of what the required goal is to achieve. Some examples of intents are as follows:

- Segment branch office networks from headquarters to ensure data leak prevention.
- Limit any possibility of excessive use of available bandwidth for media streaming.
- Make e-mail notifications of any spyware application in use to the security desk.
- Monitor daily use of the company's homepage and sort the results by most viewed sections.
- Tag with risk level 4 connections to the network for all employees newer than 3 months.
- Change data backup schedule from Sundays to the same time window on Saturdays.

Not every device in IBN is able to understand intents in this form. Even more so, not every network device will be able to understand a general network policy. It is important to understand that the critical point of interoperability with regard to network policy resides in a realized information model rather than in a transport protocol and its message semantics. Instances of the classes described in the core policy schema contain data that describes operational policies. To affect policies in the network, entities within the network must interpret prescribed policies. Not all entities within a network necessarily possess the ability to interpret policies directly. Such entities may require assistance in interpreting policies [33].

Functionality of the IBN concept

In its current form, intent-based networking is a network management model where the management itself is achieved via dialogue between human and artificial intelligence (AI), and with the use of AI subset technologies such as machine learning (ML) and big data. The need for such a dialogue has come up, as it has been reported that the number 1 root cause of system outages is human error. Therefore, supposedly, AI is to be put in the role of network manager and plaster those man-made contributions with its own ability to conjure network policy for control over different sorts of network equipment, including network controllers themselves. The network policy by itself can be of various use case scenarios as well. Ranging from dialog initialization among communication parties, data flow switching, routing or forwarding, or even network troubleshooting for problem mitigation.

Another aspect to consider is the proposal of IBN not to be focused only on physical network connectivity. It is stated that IBN can assist with business-related issues and coordinate business-specific application workflow. This ability is strongly tied to proper physical network resource distribution for optimal application performance. For example, a live (in-real-time) online event might require more resources than that of a cacheable video stream.

The design as provisioned by Cisco in the early days of development of the second reincarnation of the IBN concept was advertised and focused on end-user intents (i.e., arrange a meeting between sales and development company departments). This, however, did not gain any traction from the Society of Science as it did not solve any networking-related issue.

Shifting focus away from personal assistance, IBN succeeded in landing a seat in the ITU-T framework of Future networks. Intent-based networking model consists of five building blocks, which are as follows:

1. Profiling – This block is responsible for intent ingestion. For IBN, what is ingested is the request of the user. From the user request a usable information is extracted and an abstraction of it is formed.
2. Translation – This block is responsible for intent's abstract reinforcement into applicable policy. This policy must consist of three ingredients – event, action, and condition (EAC).
3. Resolution – This block is responsible for configuration creation. To be enforced, policy must be granulated according to underlay technology knobs and levers.
4. Activation – This block is responsible for intent absorption and feedback. Once the network farm is updated, the outcome (current network state) is reported back for evaluation.
5. Assurance - This block is responsible for intent conditioning and behavior forging. Whether the network farm will require more finetuning or will be content with what it got is up to this block. If IBN can keep functioning and does not feel downgraded by the ingestion of the input, it does save the current configuration and snapshots the previous one. This is also how intent drift tracking is maintained.

In IBN, the intent application does require human intervention and assistance as the process itself might require additional input and a decision-making of a favorable outcome from one predicted by artificial intelligence.

Underlay technologies

Artificial intelligence (AI), machine learning (ML), and big data are technologies that emulate reasoning and decision-making processes. There are many more processes that need to be done for any operation (i.e., data transmission) to be carried out. Some of which are those already discussed technologies like service function changing, segment routing, network device programmability, etc.

In fact, IBN is not put in place to replace or surpass any of the existing networking technologies. Its duty is to make use of them, imitating the management of an actual user (herein, technical staff) behind the wheel.

Intent processing and network control underlay technologies are as follows:

- Artificial Intelligence (AI) – assists in real-time decision-making and enablement of intelligent human-like planning, whether it is for new network configuration or troubleshot of an issue. Many subdivisions of AI can be used. For example, in intent input, natural language processing (NLP) can be used. For latency, jitter, or other physical issues with a measurable margin, a generative adversarial network (GAN) can be used for the generation of solutions from sampled cases.
- Machine Learning (ML) is a tool for data-based reaction or prediction generation. This tool is a subset of AI and primarily focuses on forecasting intent drift from the original result, anomaly detection in network usage, and traffic classification.
- Big data – this tool focuses on processing of massive amount of information or information that needs to be processed in a relatively short period of time. For example, network snapshots should be gathered from multiple points in the network fabric to make decisions about the network's health.

It has been said that AI applications in network management in the past have largely focused on classification problems. Examples include analysis by intrusion protection systems (IPS) of network traffic flow patterns to detect suspicious network traffic, classification of encrypted network traffic for improved QoS treatment based on suspected application type, and prediction of performance parameters based on observations. In addition, AI has been used for troubleshooting and diagnostics, as well as for automated help and customer support systems. However, AI-based solutions for the automated planning of actions, including the automated identification of courses of action, have to this point not been explored much [34].

When put simply, machine learning (ML) provides a way to teach computational systems to gain knowledge from data without necessarily being explicitly programmed, in order to realize complicated tasks such as the detection of characteristics or the prediction of behaviors. As ML becomes an important technical trend in the industry, operators are searching for cost-effective ways to incorporate ML into future networks, including IMT-2020 [35].

2. NETWORK TOPOLOGY-AWARE SF CHAINING

This section is an overview of research done in Appendix 1 and a continuation of previously made statements, which are as follows:

- A generalized problem in networking is addressed - Enhancement of network path generation and coordination of transmitted information in them with the use of service function chaining.
- An adaptive solution proposed and covered in this section - A heterogeneous service function path encapsulation for service function chaining.
- Some excerpts of the technology introduction from sections 1.2. and 1.3. - Network functions (NF) are all actions performed upon transferred data except the transference itself. In network function virtualization (NFV), for specific-purpose built network equipment is replaced by general-purpose devices on which the NFs are emulated. Service function chaining (SFC) is a network data flow steering method that is used to modify the packet forwarding path, altering the packet processing. In SFC, NFs are referred to as service functions (SFs). The SF path's length is limited by the SF encapsulation itself.

2.1. Expansion of the horizons

It has been stated that, for the migration and co-existence of legacy and compatibility with existing platforms it is for NFV to be susceptible to network heterogeneity. Implementations of NFV must co-exist with network operators' legacy network equipment and be compatible with their existing network management and resource orchestration elements. The network functions virtualization (NFV) architecture must support a migration path from today's proprietary physical network appliance-based solutions to more open standards-based virtual network appliance solutions. In other words, NFV must work in a hybrid network composed of classical physical network appliances and virtual network appliances [36].

To deliver such heterogeneity, SFC can be used. Both physical and virtualized network entities can manage to process SFC encapsulation. However, in the model of service function chaining (SFC), it is stated that the length of a service function chain can be no longer than that of a size of service function (SF) domain, as it cannot cross it. In fact, an SF domain must be within the bounds of a single administrative domain.

These limitations greatly reduce the capability of the packet steering technique, as it does not allow cross-network interconnection or overlay topology administration, where underlay topology would have multiple tenancies in use. The limitation on SF chain length limits the ability of multi-tier or even hierarchical service function alignment in the topology.

Another important aspect is that within the SF domain, a network traffic encapsulation was put in place throughout the entire SF chain. Meaning that even for portions of the network, where the path taken by the rules of encapsulation policy, including those in which the SF path does not differ anyhow from the underlay transport topology, the encapsulation would still be in use.

To address all three issues, network topology-aware service function chaining is proposed. The basic principle of it is to remove the encapsulation wherever it is not necessary to use it. In this case, the original design of service function changing does not require any modification, but all three limitations are abolished. The difference between full and partial SF encapsulation is shown in Figure 12. (created by the PhD author) where in partial SF encapsulation, only path indifference is encapsulated.

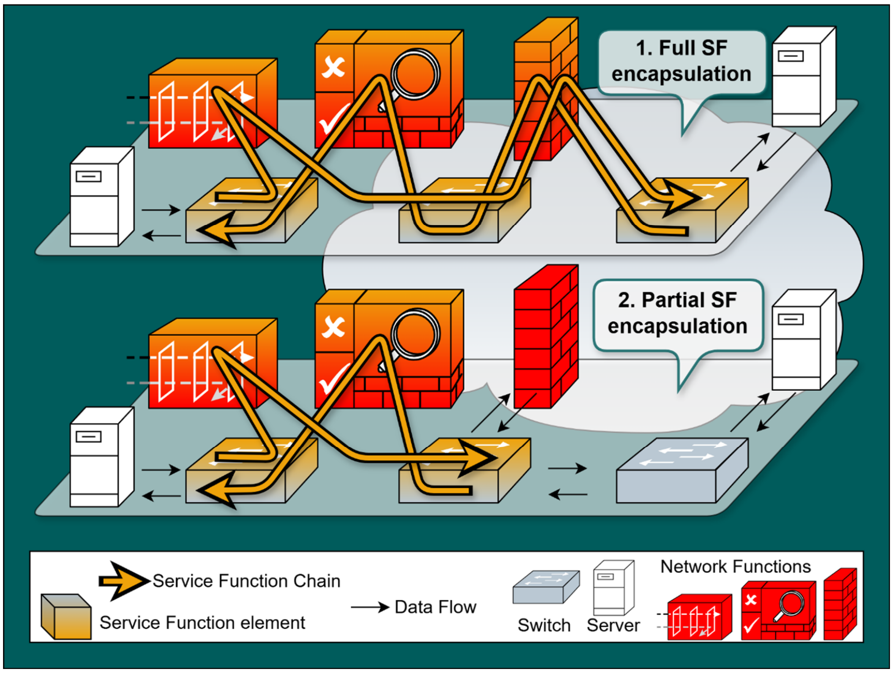


Fig. 12. Full SF encapsulation versus partial SF encapsulation

2.2. Design goals

First, the limitation of the SF chain length is lifted. From the SF forwarding plane viewpoint, traffic that had encapsulation removed and later in the path is to be put back on can be treated as a new encapsulation iteration, while from the perspective of the control element, backtracking of similar encapsulation value usage throughout the network can take place, therefore enabling mapping the whole path taken.

Second, the limitation on tenancy is removed, and underlay can span over multiple administrative domains, as at the network border, encapsulation can be lifted and put back in place once network traffic must enter the SF chain on the receiving network.

Third, allowing for encapsulation removal in places where the network traffic path needed to be taken did not differ from the underlay topology defined path would reduce overall overhead and ease network configuration management.

Two different network topology-aware SFC encapsulation methods were proposed:

- **Initial-partial encapsulation** – a method where, on network traffic arrival at the ingress, a classification is required, and initial encapsulation is imposed on all network traffic. This encapsulation is partially continued as necessary for the SF chain to be fulfilled.
- **As Required encapsulation** – a method where encapsulation is only used when the requirement is to alter the taken path from that which would have been set by the underlay transport network topology. Therefore, this method would require even less excessive overhead for network traffic steering.

These encapsulation application methods address not the issue of what to encapsulate, but when to encapsulate. As what to encapsulate is the core job of the network function classifier element. The deployment model of SFC can be classified in consideration of both the roles and the capabilities of the involved SFs. An SFC enables the creation of composite network services by constructing an ordered set of service functions. In addition, an SFC can be fine-grained or coarse-grained, depending on the capabilities of the classification function in the ingress classifier. The classifier has the responsibility to identify and then classify packets in order to steer them through the proper SFC. The classification function can be deployed in either a single classifier or multiple service functions that include a classifier. Therefore, deployment models can be centralized or distributed within each SFC [37].

Initial-partial encapsulation holds on to the encapsulation application at the beginning of the service function chain, but it opts out of unnecessary encapsulation whenever possible. This method might seem wasteful in comparison to the As Required encapsulation method, but it allows attaching metadata to any incoming traffic, therefore widening the opportunity window for next-in-line service functions to understand what to do with the incoming data flow.

The As Required encapsulation, on the other hand, is meant to be used only for case scenarios where, due to the complexity of network topology or the specifics of incoming flow data, targeted processing would be in need. Respectively, either an alternate path for encrypted flows would need to skip deep packet inspection, or a statically routed network topology would need an overlay to alter the packet path at a certain point.

Not all SFC encapsulations carry additional fields for metadata. For example, VLAN or MPLS encapsulations do not have any re-purposable fields, but their identifier fields. It is the Network Service Header (NSH) that essentially has all the fields required to carry additional information about data flow and how it should be processed. This, of course, can be overcome by sharing this information not via the data channel but over the control channel from the network controller to forwarding devices.

2.3. Emulation of network topology-aware SF chaining

An emulation was done to test the difference between encapsulation methods. Figure 13. (created by the PhD author) shows encapsulation usage in the emulated topology. It also shows the underlying Mininet network setup. The topology consists of three Open-Virtual switches (OVS) and seven hosts (H1-H7), and a controller. Topology elements were as follows:

- H1 – Data flow source (Src);
- H2, H6 – SF classifiers (CL);
- H3, H4, H5 – Service functions (SF n);
- H7 – Data flow destination (Dst);
- OVS switches – SF proxies (SFP) and SF forwarder (SFF).

Each emulation was run multiple times with the use of four different SF encapsulation protocols – MPLS, VLAN, NSH type-1, and NSH type-2, with and without a context header. Also, different packet data units (PDU) were used – in one configuration, a maximum PDU (MTU minus Headers and Encapsulation) was used, while the other had it fixed at a value of 20 bytes.

The Scapy packet crafting tool was used to add or remove SF encapsulation. The Ryu controller was used to apply a forwarding policy to Open-Virtual switches. Ryu was connected via localhost, therefore, an outbound connection, as it was run in the same environment where Mininet was. Link speed was kept at 10Mbit/s with a 10ms delay.

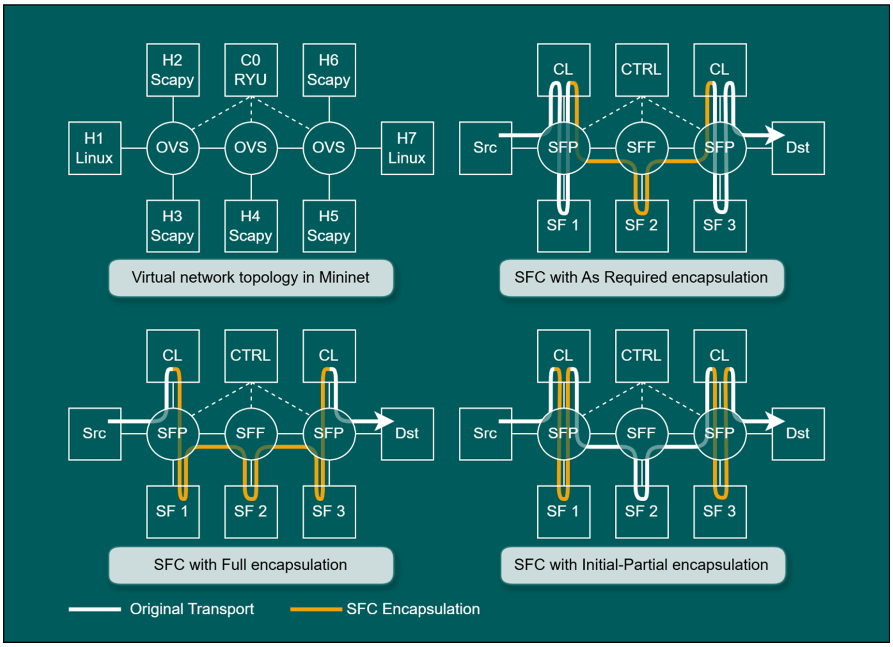


Fig. 13. Virtual network topology and SFC encapsulation application methods

2.4.Results and conclusions

The scientific method used in this research is the conduct of an experiment that is an emulation of network topology for the evaluation of different encapsulation application methods. Shown in Figure 14. (created by the PhD author) are SF encapsulation results, where different encapsulation methods are compared by the overhead in bytes they create.

SFC encapsulation application approach (previously referred to as SF encapsulation method) comparison indicates that in a given topology, it is possible to decrease overhead by half if comparing Full encapsulation against As-Required encapsulation.

In fig. 14. shown graph shows direct dependence on the emulated network topology (two-tier star topology with three service functions). In other cases, service function count and topology type can differ, and depend on network size or other factors. This fact does not remove the validity of the results that clearly indicate on ability to decrease the overhead of data packet transmission.

Discussed overhead refers to the additional header applied to data packets when they cross the service function domain. As shown in Figure 14. overhead for in this work developed and evaluated partial encapsulation approaches (initial-partial and as required) is less than that of a full encapsulation (in all SF path segments).

Overhead decrease (the difference between full and partial encapsulation in emulated topology) is up to 50% (38 bytes of overhead for a single packet versus 16 bytes as shown in fig. 14. for NSH MD-Type 2 with context encapsulation). It conforms to all emulated encapsulation protocols – virtual local area network protocol (VLAN), multi-protocol label switching protocol (MPLS), and network service header protocol (NSH).

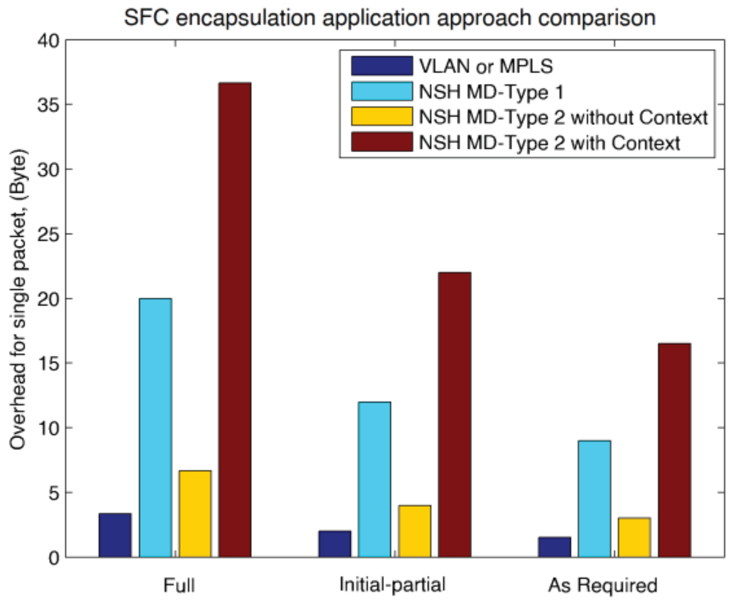


Fig. 14. SFC encapsulation application approach comparison

3. REACTIVE SF PATH DISCOVERY

This section describes the research in appendices 2. and 3. and is a continuation of previously made statements, which are as follows:

- A generalized problem in networking is addressed - Enhancement of network path generation and coordination of transmitted information in them with the use of service function chaining.
- An adaptive solution proposed and covered in this section - A reactive service path discovery with the use of a default path and data flow reclassification.
- An excerpt of the technology introduction from sections 1.1. and 1.3. - A match can be made against any packet header. Sending a packet to the controller signals a need for modification in the forwarding policy. The SF classifier (CL) is placed at the ingress for the application of encapsulation onto incoming network data flows.

3.1. Rule-base update at runtime

The mapping and the rules information at the classifier (CL) component may reveal the traffic rules and the traffic mapped to the service function chain (SFC). The SFC information collected at an SFC component may reveal the service functions (SFs) associated with each chain, and this information, together with classifier rules, may be used to bypass the SFC and trigger any internal attacks [38].

By its architecture, service function chaining (SFC) did not formulate how service function (SF) classification obtains the network traffic classification policy. In recommendations, it was noted that the classification would be set up either by the administrator or the control plane element proactively before the network traffic arrival. It was also noted that there would need to be established communication channels among SF elements that reside within the data plane, and between the data plane and control plane. These communication channels would be used for the enablement of operation, administration, and maintenance (OAM) functionality.

Therefore, setting up a network traffic classification policy for proper service function chaining is just as hard and complex task for network administrator as making a static routing. The whole point of minimizing human intervention was not reached by default in SFC. It still requires the network administrator to preset the SF path map by himself.

To address these issues, the reactive service function path discovery approach was introduced. The core idea here is to make the SF path rule-base at runtime and not before. Once incoming traffic is captured, it is analyzed and set out along a particular SF path.

3.2. Design goals

As it is impossible to guess the context of the incoming traffic, a generalized classification method should be put in place. A default SF path was set up to fill in the post of classification for previously unknown network traffic. Respectively, the default SF path would be used on all network traffic that does not have any existing classification in the rule-base of the SF classifier.

The default SF path is not fit to steer the arriving network traffic to it appropriate service function chains. In fact, it might get it wrong most of the time. This brings up a necessity for a re-classification. Delivering network traffic to the service function is enough to ignite the traffic re-classification. They are configured to process network traffic that they understand and drop all the rest. This working manner is to be used to distinguish proper classification. So, in a case where the arrived network traffic is not fit to be processed in the current SF, then instead of dropping the traffic object, namely packet, SF sends it to the control plane for re-classification.

Through the use of OAM channels, a two-loop working principle for SFC is designed. In it, one loop serves initial network traffic classification while the other is meant for subsequent classification. In this way, it is a task for SFs themselves to fill the service function chaining network traffic classification policy. The three key objectives of reactive path discovery are:

1. A default path setup by the administrator or control plane element.
2. Initial network traffic classification for previously unknown network traffic.
3. Subsequent network traffic classification obtained by SF-initiated policy updates.

The design principle of reactive service function (SF) path discovery, in opposite to proactive discovery, is shown in Figure 15. (created by the PhD author).

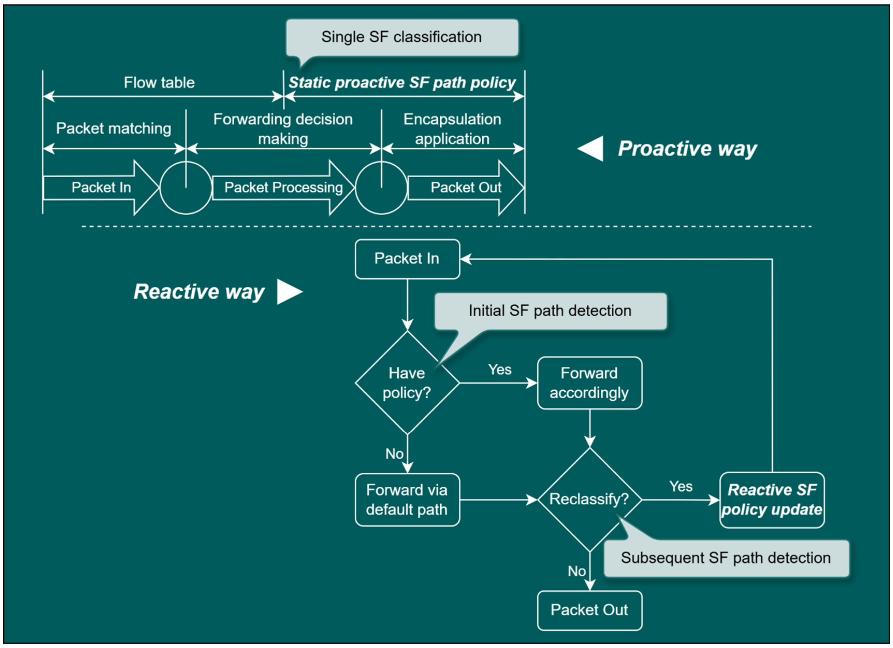


Fig. 15. SF path discovery approaches

In Figure 15. proactive way shows how straightforward the packet classification mechanism is when a predefined static path policy rule-base is set for the incoming network traffic. It starts with packet matching, which results in forwarding decision-making that dictates what encapsulation is to be applied. These three steps are implemented with information found both in the forwarding table and the static proactive SF policy rule-base. However, the reactive way shows an alternative packet processing mechanism that allows for modification of itself at runtime. Respectively, the incoming network traffic does acquire encapsulation, much like in a proactive way at its arrival. But this time around, the acquired encapsulation can be updated via a reclassification process initiated by a service function (SF) that is unable to process the packet.

3.3.Emulations of reactive SF path discovery

An emulation in the Mininet emulator was done. Actually, this solution got two test beds. One where the focus was put on finding how much or if there is any benefit for doing a subsequent SF path detection. The second test focused on differences among asymmetrical and symmetrical SF paths. As it is possible to process network traffic differently for incoming and reverse flows. In Figure 16. (created by the PhD author) Mininet emulation topology and used SF control channel mechanisms are shown.

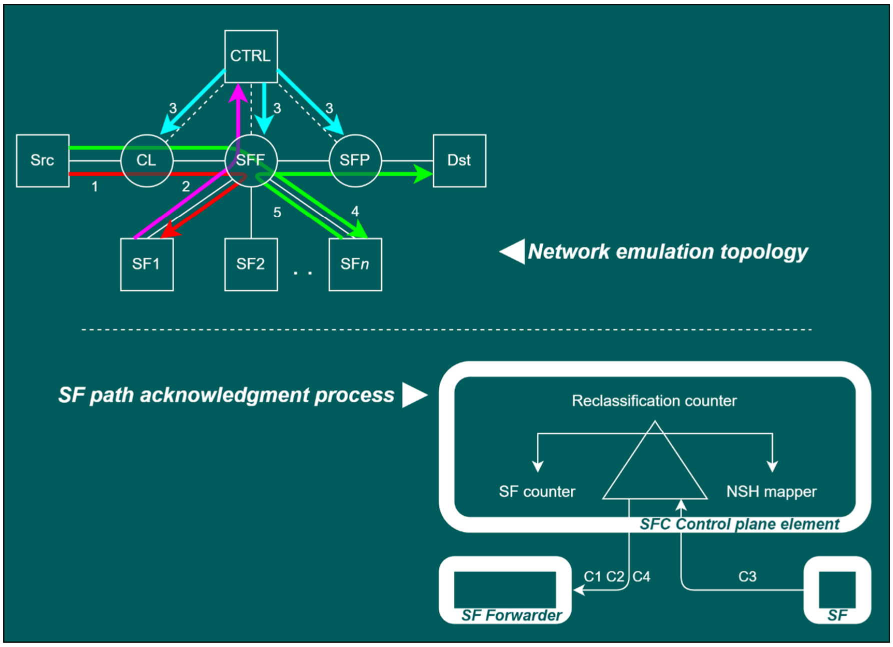


Fig. 16. Network topology and reclassification via control channels

In Figure 16. Mininet network emulation topology consists of the following elements:

- CTRL is the RYU remote controller connected via localhost.
- CL, SFF, and SFP are Open-Virtual switches (OVSs).
- Data flow source (Src) and destination (Dst) are Linux hosts.
- Service function SF1 to SF(n) are Mininet hosts running SCAPY. Topology was fitted with up to 5 SFs during emulations.

An example process of SF path discovery reactively is fulfilled through five steps:

1. A default path leads data flow to the wrong service function (red arrow).
2. Service function initiates SF policy update (purple arrow).
3. Update is issued to all SF forwarding elements upgrading both the SF classification policy rule-base and flow forwarding tables (blue arrow).
4. A now specified path to arriving network data flow leads to the right service function (green arrow).
5. Service function processes data flow, which results in data arrival to its destination (green arrow).

Service function path acknowledgment process is fulfilled via the use of four control channels – C1, C2, and C4 are responsible for communication with the SF classifier, forwarder, and proxy (all SF forwarding elements based on Open-Virtual switches), C3 is used for SFs (Linux hosts running SCAPY) to send update requests to the controller. Controller keeps count of the SFs that the packet has already traveled to and how many reclassifications have occurred to calculate the proper NSH map.

3.4.Results and conclusions

The scientific method used in this research is the conduct of an experiment that is an emulation of network topology for the evaluation of different service function path discovery approaches.

Results acquired from multiple runs of emulation topology given in Figure 16. are shown below. They not only show the advantages of the use of subsequent SF path discovery in both asymmetrical and symmetrical environments, but also explore if any gain in the use of previously researched proposed encapsulation methods is achieved.

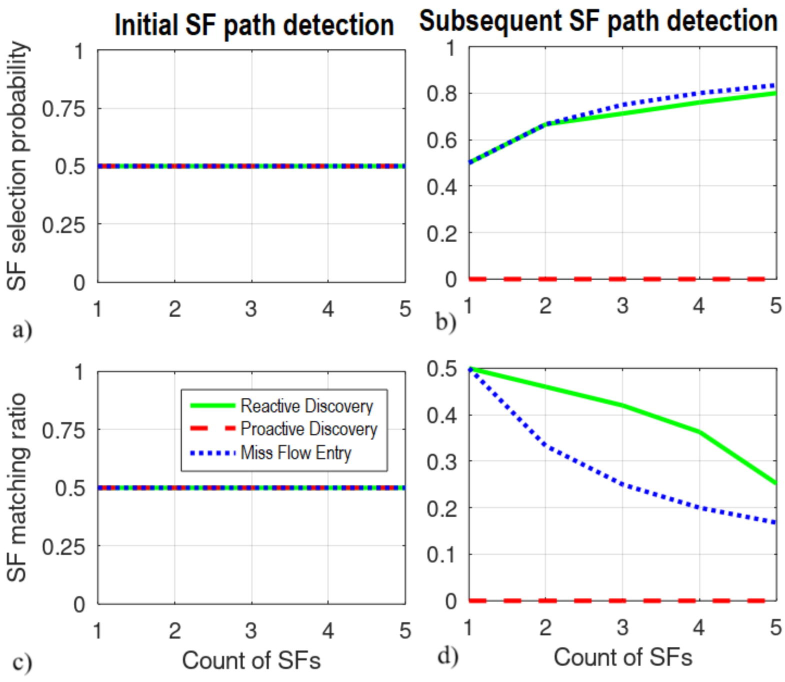


Fig. 17. Reactive and proactive way comparison:
a) probability to select a data flow appropriate SF in initial classification;
b) probability to select data flow appropriate SF in subsequent classification;
c) SF match discovery ratio in initial classification;
d) SF match discovery ratio in subsequent classification

Figure 17. (created by the PhD author) shows the difference in reactive and proactive SF path discovery ways. Graphs are directly dependent on the emulated network topology and characteristics of the service function domain. In the emulation environment SF path consisted of a single SF. A single-parameter classification was used. In other cases, the SF path can contain more SFs, and classification can be enacted again by other parameters than those used in the initial classification. However, from emulation results it is clear that the reactive way is capable on improvement of improving SF path discovery in case of subsequent classification.

Graphs a) and b) of Fig. 17. show the probability with which, as a result of classification, an appropriate SF path will be selected for ingress data flows. Graphs c) and d) show a ratio between all possible matchings and those that are appropriate matches.

In accordance with characteristics, graphs a) and c) show a probability and matching ratio of 50% equally for reactive and proactive ways, as there is only a single opportunity to make the classification (classification at the beginning of SF chain). Whereas in graphs b) and d) a proactive way cannot do subsequent classification, and therefore its value is 0. A reactive SF path discovery way, with the use of subsequent classification, can find an appropriate SF path. The probability of a reactive way in graph b) will never surpass the value 1, and its closing in on the maximal value can be explained by the fact that the possibility of not finding a valid SF path diminishes as subsequent classification count increases. Unfortunately, in graph d) an inverted tendency can be observed as the valid SF path ratio against all possibilities shrinks with the increase of classification count.

Additional outcomes:

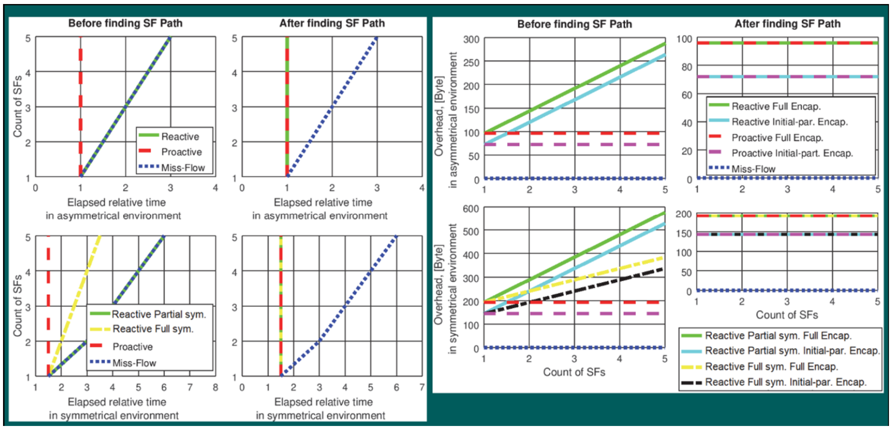


Fig. 18. Elapsed relative time graph and SFC-generated overhead graph

Elapsed relative time (left on fig. 18.) (created by the PhD author) is the interval required for the SF path discovery process to take place. From graphs, it is clear that proactive path setup does not need additional time to be applied, while reactive does on occasion, where the path is still unknown. Once the SF policy is updated, reactive SF discovery does not spend any additional time as well.

Service function chaining (SFC) generated overhead (right on fig. 18.) is a cumulative value gathered from counting how much overhead SF encapsulation makes. Similarly, as with the elapsed time, overhead is bigger for reactive SF path discovery before finding a valid path, but once it has the path, overhead is the same as with proactive SF path policy.

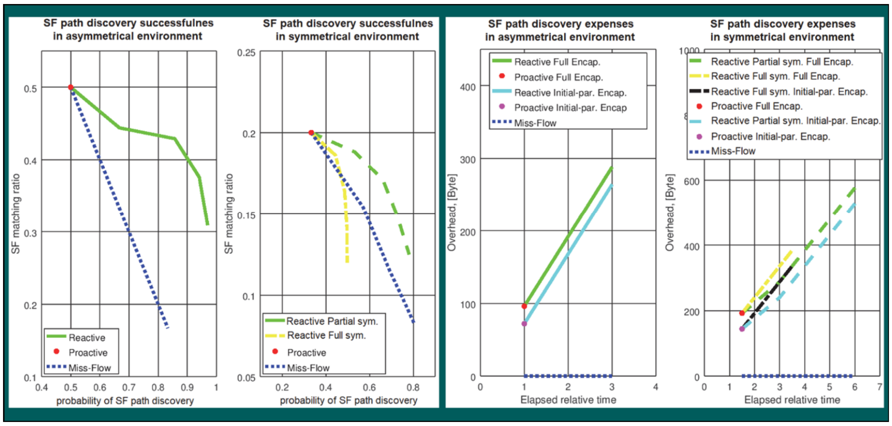


Fig. 19. SF path discovery successfulness graph and SF path discovery expense graph

SF path discovery successfulness (left on fig. 19.) (created by the PhD author) is the proportion between the probability of valid SF path discovery and the SF matching ratio. From this graph, it is clear that as a static value, proactive SF policy does not benefit from more reclassifications as reactive SF discovery does.

The SF path discovery expanses (right on fig. 19.) is a scale between elapsed relative time and SFC generated overhead. It shows how both parameters are directly proportional for reactive SF path discovery, as the gain in one parameter leads to a gain in the other.

Overall results:

This research explored an approach of reactive service function (SF) path discovery, which leveraged network automation via the use of a network controller for policy updates. The reactive approach is a proposed solution that would minimize required human assistance in network traffic classification.

Results indicate that proactive SF policy setup is static, as it cannot use subsequent network traffic classification, at least not in a single conditioned classification. The probability of a valid SF application stays constant regardless of the SF count in the topology.

It is recommended to use asymmetrical service function chaining (SFC) for network traffic that requires a high probability of valid SF path discovery, but symmetrical SFC is an option that is applicable for network traffic where SFs must see both flow directions.

Reactive SF path discovery gains a higher valid path discovery probability in both asymmetrical and symmetrical environments. It allows for SF policy update at runtime, therefore delivering a higher level of automation via the use of control channels.

4. SF SHARE IN INTER-DATA CENTER NETWORKS

This section is an overview of research done in Appendix 6 and a continuation of previously made statements, which are as follows:

- A generalized problem in networking is addressed - Enhancement of network path generation and coordination of transmitted information in them with the use of service function chaining.
- An adaptive solution proposed and covered in this section - Service function share in inter-datacenter networks.
- An excerpts of technology introduction from sections 1.1. to 1.3. - Resources are either simple or can be aggregated to form complex structures that can be viewed as a single entity. The goal of the architecture is to open possibilities, and especially to expose common ground, such that silos can be collapsed whenever and however it makes sense. Regardless of granularity, the classification information can be represented in the network service header (NSH).

4.1.Crossing the Bering Strait

Network resources are more and more often not exclusively used by one operator. Enterprises are already hosted on many multi-domain infrastructures today. The principles of virtual network functions can increase the flexibility to share resources and decrease setup and management costs. The service provider can make available a suite of infrastructure and applications as a platform on which the enterprise can deploy its network applications. With this platform, the enterprise could develop its own network service customized to its business purposes [39].

Figure 20. (created by the PhD author) illustrates Multipath TCP (MPTCP) usage and how unaware of it are the transit network equipment. In theory, only devices that examine network packets up to the transport layer (L4 of the ISO OSI model) should discover their usage, as TCP flags indicate on flows being a part of sub-flow relations.

From a sub-flow viewpoint, the Multipath TCP protocol is completely symmetrical. Both the clients and the server have the capability to create sub-flows. However, in practice, the existing Multipath TCP implementations have opted for a strategy where only the client creates new sub-flows. The main motivation for this strategy is that often the client resides behind a NAT or a firewall, preventing passive sub-flow openings on the client [40].

In Figure 20. SF (Firewall) in the remote site creates a bottleneck for flows that traveled in parallel to the site, now are forced to be concurring again in the queue of the SF buffer. This nullifies the use of MPTCP in case scenarios where SF processing power and bandwidth limitations are not greater than those applied by the transit network on parallel flows. Sub-flows are based on end devices (computer and server) being in a multihomed environment, respectively, multiple communication paths between conversation parties are available. In fig. 20. both the user's network and the remote site are multihomed environments, but MPTCP can also leverage a scenario where only single side is multihomed.

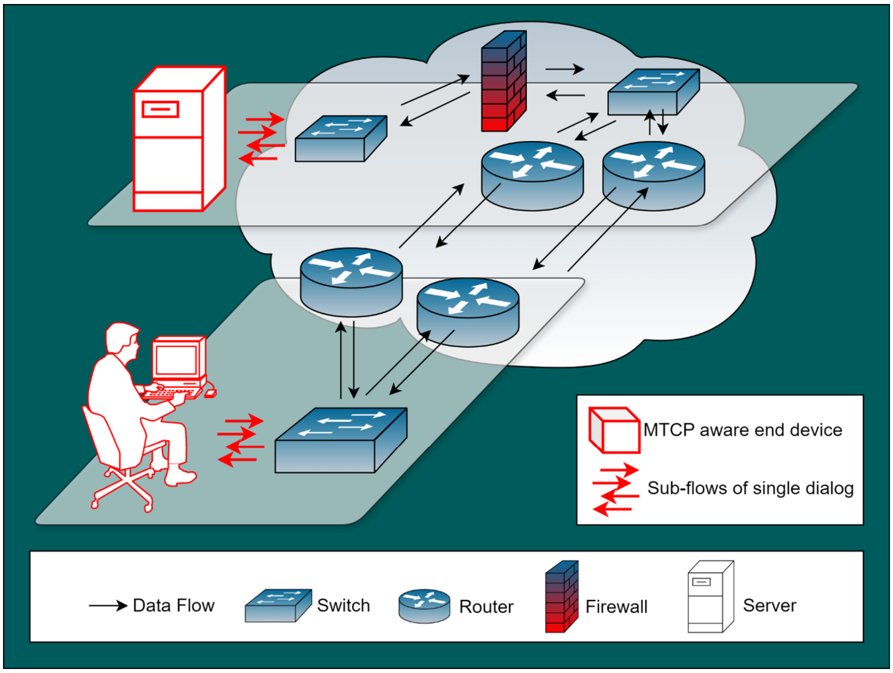


Fig. 20. Multipath TCP bottleneck

Security-wise, MPTCP also uses a key to map sub-flows to the dialog. Since key generation is implementation-specific, there is no requirement that they simply be random numbers. An implementation is free to exchange cryptographic material out of band and generate these keys from this material, in order to provide additional mechanisms by which to verify the identity of the communicating entities. For example, an implementation could choose to link its MPTCP keys to those used in higher-layer transport layer security (TLS) or SSH connections [41].

The global infrastructure of connected devices is still a very much segregated system. Service function chaining (SFC) elements placed in one data center are not in synchronization with their equivalents placed in another data center. Not only does this create unnecessary withholding of processing power, but it may also become a source of possible bottlenecks in the network.

If MPTCP data flow belonging to the same dialog is divided into sub-flows and directed towards multiple IP addresses but is steered through the same SF path at the receiving end, it defeats the purpose of dividing the flow in the first place.

It is often that data centers find themselves neighboring other data centers. Sharing a direct link between them is beneficial to both tenants and operators, as the resources can be accessed. The inter-data center network is what enables service function (SF) sharing. In this way, multipath transmissions can avoid bottlenecks as flows can be steered through equivalent SFs residing in multiple locations.

4.2. Design goals

Service function sharing between data centers via leased line allows for extending the underlay networks' functional capacity and increasing overall capability. This functionality can not only be used to double an existing SF, for example, having two firewalls (FWs) instead of one, but also to get new SFs in the service function chain (SFC), like adding a deep packet inspection (DPI) after FW. In Figure 21. (created by the PhD author) a service function (SF) sharing in inter-data center networks is shown. The communication spans over three remote sites. Where two are multihomed and the third has a direct connection to the second one via a leased line.

There are many reasons for having a spare SF in reach. One might be redundancy for the avoidance of network service faults. It refers to network service performance anomaly, including service interruption, performance degradation, etc. It can happen that the network Service performance runs far from the expected capacity as configured at the beginning of its instantiation. At this time, the SFs may still work, but probably at low performance [42].

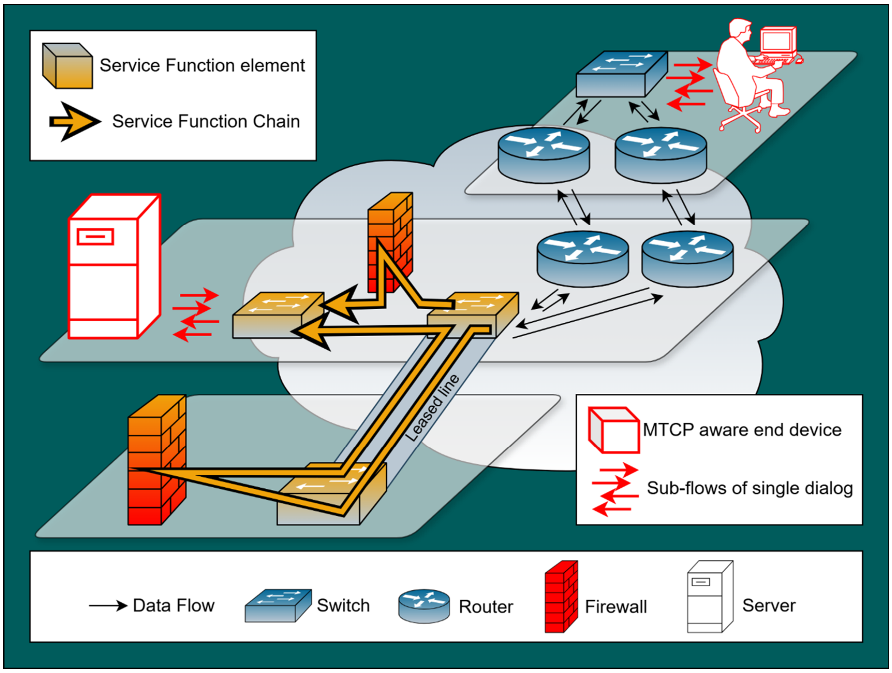


Fig. 21. Service function sharing in an inter-data center network

Even though the global network mostly consists of packets, network traffic is not homogeneous. There are two extremes of flow types that are called elephant flows and mice flows. As the naming hints first type is something big while the other is something small. The units of measurement set aside the thing that is measured as big or small, which can be the size of the packet itself, the size of the volume per period, or the length of a single flow. Of course, there is the third type, which can be referred to as normal flow, sitting in between the other two.

The Internet in 2025 is a best-effort service. When traversed by any type of flows, it is not possible to determine which type would have better transmission parameters. It was taken into consideration as the same uncertainty applied in the SFC, reliant on SF sharing in inter-data center networks, as their resource load and computation power is a constantly changing variables.

Operations, Administration, and Maintenance (OAM) tools are an integral part of the SFC architecture. These serve various purposes, including fault detection and isolation, and performance management. For example, there are many advantages of SF path liveness detection, including status reporting, support for resiliency operations and policies, and an enhanced ability to balance load [43].

The goal of this study was to determine if it is beneficial to steer one of two sub-flows of multipath TCP traffic over a leased line into a neighboring data center for service function chaining and afterwards receive it back to forward it to the destination. In real-life scenarios, the network equipment of both data centers would need to be under a single administrative domain, as the control of service function would require a joint understanding of service function encapsulation, like network service header (NSH) field value meaning.

4.3.Emulation of multipath SFC

A network simulation was created for this research using of GNS3 network simulator. In this research topology was created in a way to represent a miniaturized public network. It is shown in Figure 22. (created by the PhD author).

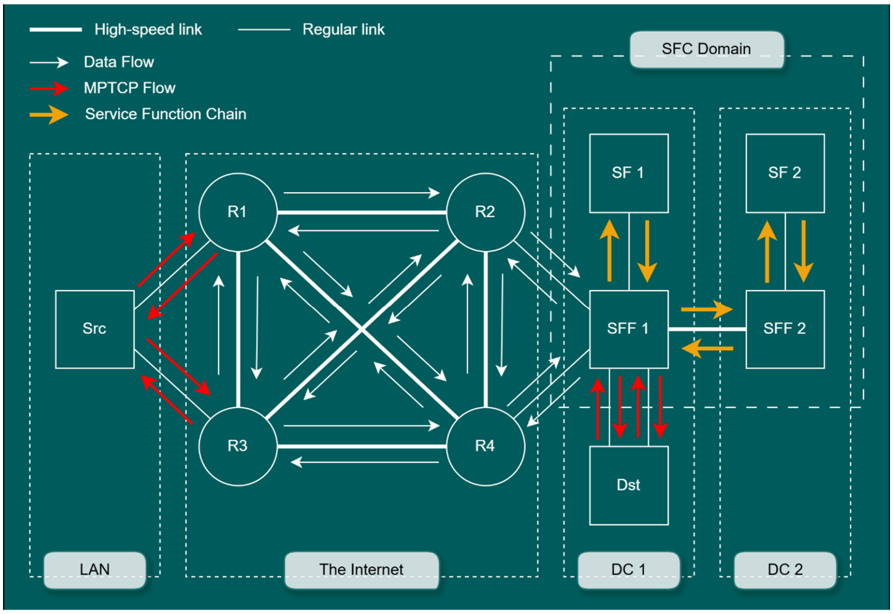


Fig. 22. Virtual network topology of the SF share in the inter-data center network

Shown in Figure 22. network setup consisted of the following elements:

1. Src and Dst are TCP connection source and destination, respectively.

2. R1 – R4 are routers running Enhanced Interior Gateway Routing Protocol (EIGRP).
3. Service function forwarders SFF 1 and SFF 2, and service functions SF 1 and SF 2 are all run on Open-Virtual switches (OVS).
4. Regular link speed is Ethernet, but High-speed link is Fast Ethernet; both are deliberately chosen much smaller than the physical machine capability.

For network traffic generation, an Iperf bandwidth testing tool was used. Three different TCP flow types were generated accordingly:

1. Normal – a TCP connection with a single dialog run for 10 sec.
2. Elephant by length – a TCP connection with a single dialog run for 60 sec.
3. Mice by length – a TCP connection with 5 dialogs run in parallel for 0.2 seconds repetitively for an overall time span equivalent to 60 seconds.

Multiple reruns were made over a path with no service functions (SF), with a single SF, and with two SFs. An average of all three flow types was calculated for the given paths.

4.4.Results and conclusions

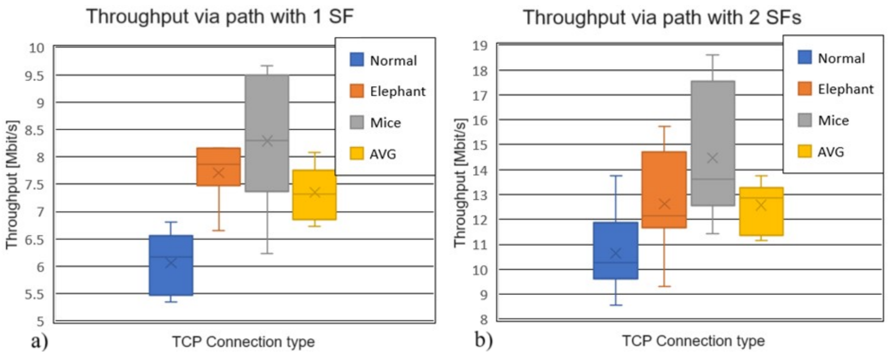


Fig. 23. MPTCP flow throughput comparison:
a) multiple flows of a single dialog are steered via a single SF;
b) multiple flows of a single dialog are steered via two SFs in parallel

The scientific method used in this research is the conduct of an experiment that is an emulation of network topology for the evaluation of the impact on data transfer rate by service function share.

Figure 23. (created by the PhD author) shows network topology emulation results, where graph a) is case when the whole dialog is steered to one SF, within single data center, but graph b) shows the result of developed and evaluated solution when flows of the dialog are steered to two SFs (SF1 and SF2) in parallel with use of connection between data centers (for example leased line).

Results have a direct dependency on the topology, where in one case, 1 SF was used, but in the other case, 2 SF paths were used, and each contained 1 SF. In other cases, SF paths can contain multiple SFs, or more than two SF paths can be used. However, the results of the emulated topology clearly show an increase in throughput for all types of flows in the **SF share solution. Increase (between use of 1 SF and use of 2 SFs in parallel) is up to 60% (8 Mbit/s of throughput versus 14 Mbit/s for AVG (average of all flows) value in fig. 23.).**

5. SEGMENT ROUTING WITH MPLS ON REPROGRAMMABLE DATA PLANE

This section is an overview of research done in Appendix 8 and a continuation of previously made statements, which are as follows:

- A generalized problem in networking is addressed - Cut on use of task-specific or proprietary devices by advancement of the feature set for re-programmable units.
- An adaptive solution proposed and covered in this section - Service function path declaration with designated segment identifier in segment routing.
- Some excerpts of the technology introduction from sections 1.6. and 1.7. - Programmable protocol-independent packet processing (P4) enables re-programmability capability on network equipment devices. Protocol-independent switch architecture (PISA) allows its users to programmatically add functionality to it. Segment routing is a packet routing method that combines source routing with packet encapsulation. If MPLS is used as a data-plane encapsulation, an MPLS label fills the role of the descriptor.

5.1. Giving purpose to purposeless

A reprogrammable data plane is a big step forward in network equipment customization. Programmable protocol-independent packet processing (P4) and protocol-independent switch architecture (PISA) are a combination of technologies that enable data plane programmability. A network administrator can implement custom packet processing without waiting for vendor support. P4 can be used to implement In-band Network telemetry (INT).

For example, tunneled packets such as VXLAN packets raise the question of whether flow identification should be based on outer or inner headers. The answer may vary depending on the goals of tracked flow monitoring, deployment aspects, operational issues, and the capabilities of the distributed telemetry monitoring system. [44].

In theory, it is possible to define and collect any device-internal information using the INT approach. In practice, however, it seems useful to define a small baseline set of metadata that can be made available on a wide variety of devices. The exact meaning of the metadata, like the unit of timestamp values, the precise definition of hop latency, queue occupancy, or buffer occupancy, can vary among devices for any number of reasons, including the heterogeneity of device architecture, feature sets, resource limits, etc. It is assumed that the semantics of metadata for each device model used in a deployment are communicated with the entities interpreting reported data in an out-of-band fashion. General metadata examples are - node ID, ingress interface identifier, ingress timestamp, egress interface identifier, egress timestamp, hop latency, egress interface link utilization, queue occupancy, buffer occupancy [45].

The advancement of network equipment generalization shifts the telecommunication industry, as previously each network device served its own purpose and had built-in functionality to do so. PISA switches have the ability to execute programs that are written in the P4 language, and therefore, they have no single built-in functionality. They are purposeless.

Segment routing with MPLS (SR-MPLS) combines source routing with network packet encapsulation. It works in the data plane as it is based on MPLS. The routing approach does not need any modification to the existing structure of MPLS. Its main advantage over simple source routing is that transit routers maintain the ability to take routing initiative, as the path that is to be taken is segmented into network segments.

A Segment Routing domain is a set of nodes participating in the source-based routing model. These nodes may be connected to the same physical infrastructure. They may as well be remotely connected to each other. If multiple protocol instances are deployed, the SR domain most commonly includes all of the protocol instances in a network. However, some deployments may subdivide the network into multiple SR domains, each of which includes one or more protocol instances. It is expected that all nodes in an SR domain are managed by the same administrative entity [46].

Implementation of SR-MPLS on top of P4 switches was the goal of this study. Creating this functionality as a program that can be repurposed on multiple network instances was another subtask. A service function chaining domain was constructed to accommodate different segment routing paths.

5.2. Design goals

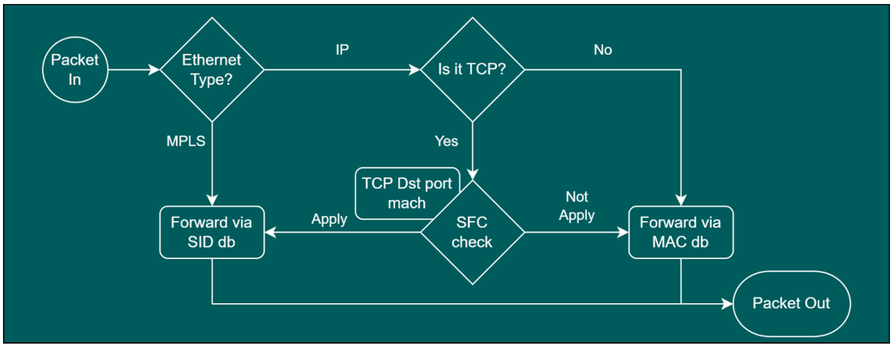


Fig. 24. P4 switch working principle

P4 was used to program PISA switches, with the program describing how to implement packet processing within them. P4 switches were used as service function forwarders and classifiers, and proxy elements. A P4 program code was developed for the P4 switch according to the scheme shown in Figure 24. (created by the PhD author).

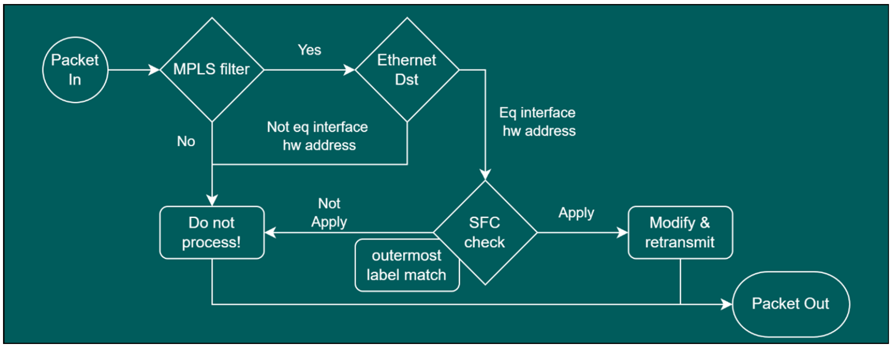


Fig. 25. Service Function working principle

SFC network traffic classification rule-base was written into the network control element, which in this study was implemented as a simple Python program connected to P4 switches via localhost. Another Python program was developed for service functions (SF) to work according to the scheme shown in Figure 25. (created by the PhD author).

SR-MPLS logic was implemented via the use of developed programs and was used to forward traffic along service function paths. SR-MPLS label distribution was static, just as network traffic classification. Both encapsulation attributes were written into the network controller element proactively.

5.3. Emulation of SFC via SR-MPLS on P4

An aspect of PISA is its ability to divide packet processing functionality between message processing and the rest of the network interface card (NIC) is for the division of control plane responsibilities. In other network deployments, the NIC message processing block configuration is tightly coupled with the host operating system, whereas the main control is controlled by network-focused control plane software [47].

Emulation network setup was made with the use of Mininet and P4-Utils. P4-Utils allowed to use the behavioral model version 2 (bmv2) software switches that work according to the V1model of portable switch architecture (PSA). The V1model working principle is shown in Figure 26. (created by the PhD author). The main advantage of the P4 switch is that packet processing can be customized to the user's requirements.

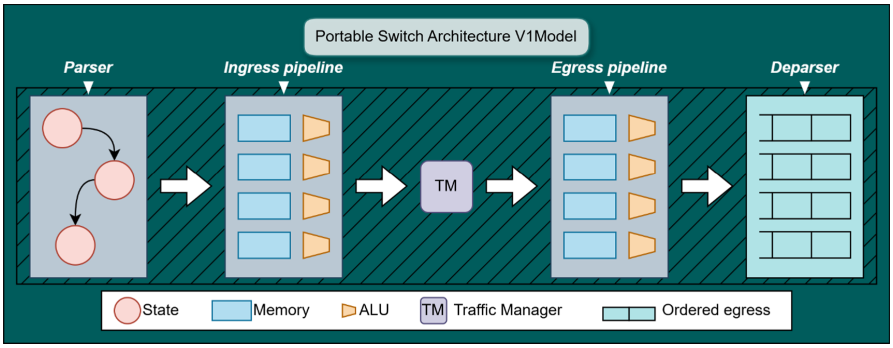


Fig. 26. V1model of portable switch architecture

In Figure 26. a PSA consists of the following elements:

- Parser – Evaluates the contents of the incoming packet according to custom filters. States hold a functionality of either accepting, discarding, or transferring a packet to another state according to the outcome of the applied filter.
- Match action pipeline – Enables packet modification according to the user's values held in memory and with the use of an arithmetical logical unit (ALU). Modified can only be values that were recognized within the filters of the parser. New elements, like additional headers, can also be added.
- Traffic manager – Functions according to vendor-specific principles, as it is outside of the user-customizable space. The traffic manager (TM) does fixed functions such as packet reordering, queuing, and buffering. Its working principle can differ from vendor to vendor or even among models of switches of the same vendor.
- Deparser – Reconstructs packet from modified header and payload for proper transmission via required egress interface. It also recalculates checksums in order for the packet to be accepted at its final destination.

In fact, a headend can steer a packet flow into a valid SR Policy in various ways:

- Incoming packets have an active SID matching a local BSID at the headend.

- Per-Destination Steering - incoming packets match a BGP/Service route, which recurses on an SR Policy.
- Per-Flow Steering - incoming packets match or recurse on a forwarding array of which some of the entries are SR Policies.
- Policy-Based Steering - incoming packets match a routing policy that directs them on an SR Policy [48].

Another important aspect to not overlook is how the default tiebreaking rules are specified. On a need for a forwarding path's select following rules are applied:

1. Determine the lowest administrative distance among the competing forwarding equivalence classes (FECs). Then filter away all the competing FECs with a higher administrative distance.
2. If more than one competing FEC remains after step 1, select the smallest numerical FEC value. The numerical value of the FEC is determined according to the FEC encoding.

These rules deterministically select an FEC to install in the forwarding plane [49].

In some cases, it is needed to utilize a packet recirculation, which is a mechanism to repeat ingress processing on a packet after it has completed egress processing. Unlike a resubmit, where the resubmitted packet contents are identical to the packet that arrived at the ingress parser, a recirculated packet may have different headers than the packet had before recirculation. This could be useful in implementing features such as multiple levels of tunnel encapsulation or decapsulation [50].

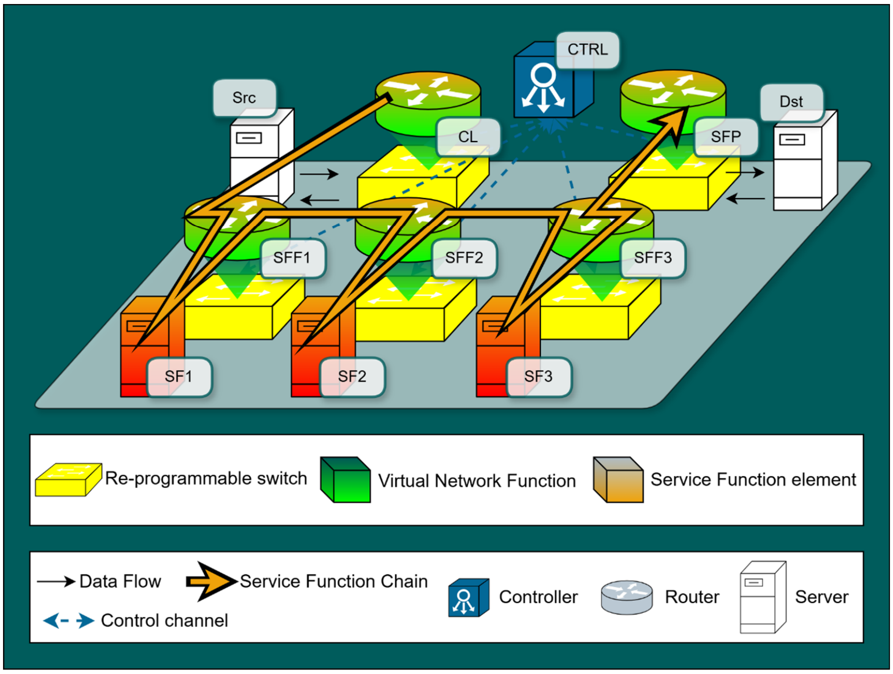


Fig. 27. Service function chaining on P4 via SR-MPLS

In the emulated topology, P4 switches worked as service function forwarders (SFFs) and had a logic to emulate segment routing with MPLS (SR-MPLS) as transport for service function chaining (SFC) fulfillment. Emulation topology is shown in Figure 27. (created by the PhD author). Classifier (CL) adds MPLS label stack as SFC encapsulation to the packet received from the source (Src). Service function proxy (SFP) removes the last label at the egress before transmission to the destination (Dst).

5.4.Results and conclusions

The scientific method used in this research is the conduct of an experiment that is an emulation of network topology for the evaluation of service function chaining with the use of segment routing.

Service function chain shown in Figure 27. had four variations – first went through none of the SFs and required no encapsulation, second went through SF2, third went through SF1 and SF3, and fourth went through all SFs. Results of network emulation are shown in Figure 28. (created by the PhD author).

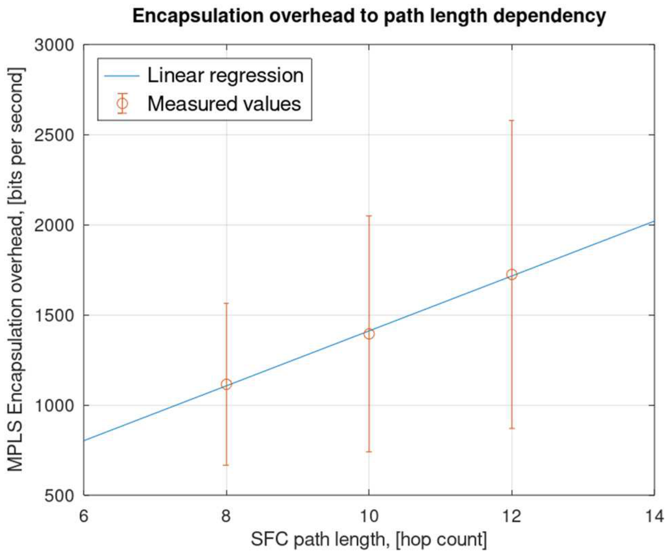


Fig. 28. Encapsulation overhead to path length dependency

MPLS encapsulation created overhead dependency in bits per second from the service function path hop count, as shown in Figure 28. Results have a direct dependency on the emulated network topology (two-tier star topology with three SFs). In other cases, the topology type or service function count can vary, etc. However, emulation results clearly show that overheads increase as the hop count gets bigger for longer SF paths.

The overhead in discussion refers to that which is created by MPLS encapsulation, which is attached to original data flows while they cross the service function domain**. Evaluated increase (between an SF path of 8 hops and an SF path of 12 hops) is approximately 40% (11000 bit/s versus 17000 bit/s of overhead in fig. 28.).** The increase originates from two causes: an MPLS label stack is used for SF path description, and only an initial classification is used at the ingress of the SF domain. In this work, it is proposed to eliminate both mentioned causes by the use of a single segment identifier (MPLS label) to describe an SF path altogether.

P4 enables any packet value modification of SR-MPLS routed network traffic.

6. ANALYSIS OF THE INTERNET OF THINGS IN ACTION

This section is an overview of research done in Appendix 4 and a continuation of previously made statements, which are as follows:

- A generalized problem in networking is addressed - Fragmentation mitigation of the Internet of Things while facilitating the use of unified communication standards for the rise of compatibility among devices in use.
- An adaptive solution proposed and covered in this section - Recommendation on implementation of specific characteristics for IoT things definition.
- Some excerpts of technology introduction from sections 1.4. - Being autonomous and connected to the global infrastructure of interconnected devices, not necessarily requiring human-assisted input for their task accomplishment, they form a unity that can be referred to as the Internet of Things (IoT). For a device to be a constrained device, it has to have limitations in any or all of these: data transmission, processing power, and energy.

6.1. A master key technology

The Internet of Things (IoT) is an ever-growing mesh of network nodes directly or indirectly attached to the Internet. Each of these nodes can be a standalone solution or an integration of already existing objects. An example of the first type can be a ping-tag that allows to locate attached to it objects like keys, a wallet, or even a car. The second type can be any home appliance, like an oven, fridge, or even small-sized electronics like lightbulbs or motion sensors.

Cloudlets can be used to facilitate IoT. Edge computing for IoT networks is one of the application areas for service function chaining. IoT devices are energy-constrained. Keeping their power on consumes their precious energy, while it is necessary if IoT devices need to wait for request arrival. It is preferable for sensor devices to send their readings to the nearby data aware networking (DAN) elements while their power is on and to go to sleep after sending the data to save energy. In turn, the DAN elements accumulate readings from surrounding IoT devices and aggregate those. In this case, a chain of functions is triggered by the delivery of data from IoT devices or named data object (NDO) producers [51].

Unfortunately, existing recommendations do a terrible job of defining what an IoT device is. They proposed an unreachable future-proof conceptual design that covered all good characteristics and functionalities possible, and made no note on how to achieve such a result. A non-exhaustive list includes interconnectivity, heterogeneity, and self-regulation.

As in old houses, a master key was made to open all the doors; designers of the concept forged an IoT thing into a general network entity. The definition goes as follows: A device is a piece of equipment with the mandatory capabilities of communication and optional capabilities of sensing, actuation, data capture, data storage, and data processing. The devices collect various kinds of information and provide it to the information and communication networks for further processing. Some devices also execute operations based on information received from the information and communication networks [52]. So, in essence, by its description, a thing is a network device that can sense or act upon the surrounding environment, or is an interpreter between these kinds of devices and the global network, or is a memory read-write system to and from the previous two categories (standalone solution or integration of already existing object). The only mandatory prerequisite is for things to be discoverable via the Internet.

6.2.Paradox of interconnectivity

Required device interconnectivity is achievable via SDN deployments. Services such as software-defined networking in a wide area network (SD-WAN) are bringing many advantages to customers, who can deploy and customize their logical network in a flexible and easy manner. The main motivation to automate logical network design and deployment is to mitigate operators' tasks when customers want flexible customization in a short delivery time. This design and deployment should be performed in a declarative manner based on design intent or network specification produced during the service design phase, to help operators reduce frequent template updates. It has to be noted that this automation could also be used for ML pipeline design and deployment [53].

The more it was thought out to keep things intact, the more it kept them apart. In this study identified definitions of attributes that are pertinent to IoT networks are as follows:

- Interconnectivity – any of the things must be able to reach any other thing on the network. This requirement not only initiates a need for unified mapping of the things but also a global database holding information on what is available in the network.
- Heterogeneity – things can run on any hardware and software and be connected to the global network through any means of communication.
- Erratic by design – IoT network must withstand any unplanned event like an outage or blackout, and any planned event like handover, reboot, or shutdown of things.
- Colossal scale – half of Internet activity is done by machine-to-machine (M2M) communication, and the trend will not diminish; therefore, the global network must be prepared to accommodate an increasing number of things.
- Identification-based connectivity – unified identification of each thing is a must to identify things in the network; however, a unified identification may not be easy to achieve, as physical things may hold a serial number, digital things may lack one, and have different values to fill in for it.
- Self-regulative – like in ad hoc networks, IoT networks must be able to self-heal, self-configure, and self-manage to maintain a minimal level of autonomy.
- Location-based capability – the whereabouts of things are to be traceable to ensure lawful usage and map relocation.
- Security – things must keep a level of integrity among themselves where the trustworthiness of information shared is guaranteed – an attribute that requires an authentication and authorization mechanism implementation.
- Value of privacy – either of the source being the owner or user of the thing, personal information processed by the thing must be treated with regard to rights on privacy, which does not override lawful intercept. Some examples of personal information leak that needs to be prevented are identity theft, medical record disclosure, or even access to timetables.
- Plug and play – user-friendly gadgets are those with a simplistic startup; therefore, things should follow best practices on putting ease of use as the top priority. Such devices are directly applicable straight out of the box.
- Manageability – the workflow of IoT needs to be overseen and administered at a level that allows it to fulfill users' required goals and desires, with which the network was set up in the first place.

These attributes may be imposed partially, creating sets of things that work with a set of other things but not all, ultimately leading to network segmentation and uninterconnectivity.

6.3. Classification proposal

First, segmentation of things according to their status in relation to the surrounding/attached ecosystem is shown in Figure 29. (created by the PhD author) and is as follows:

1. Standalone – a thing that primarily carries out a function that is the same function that it provides to the Internet of Things.
2. Add-on – a thing that primarily carries out a function that is not the same function that it provides to the Internet of Things, leaving the IoT function to be optional.
3. Module – a thing that has an IoT functionality only if it is paired with other things in a way that creates a new structure/object.

Things must not belong to two or more segments. This segmentation allows for the granulation of both physical and logical characteristics of things.

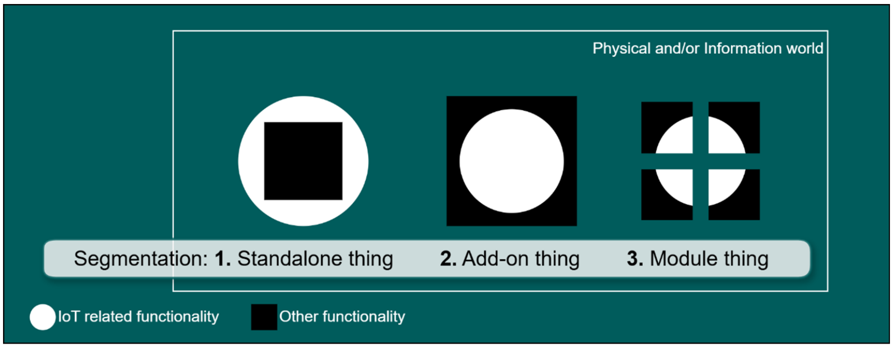

Physical and/or Information world
Segmentation: 1. Standalone thing
2. Add-on thing
3. Module thing
IoT related functionality
Other functionality

Fig. 29. IoT things segmentation

Classification of physical world objects by physical size would allow for the standardization of attributes that a thing should comply with. Such attributes would be those that were listed above, like interconnectivity and heterogeneity goals.

Physical scaling could be as follows:

1. Wearable – all things that leverage the human body, both for having IoT functionality as their primary or secondary duty. Small objects like a smartwatch (Add-on thing), smartphone (Standalone thing), wireless earbuds (Module thing).
2. Portable – all things that are intended to be a carry-on item and can still carry out their IoT functionality while they are not in a stationary state. Larger than wearable objects like a laptop (Standalone thing), an air quality meter (Add-on thing), a Wireless camcorder (Module thing).
3. Fixed – all things that have no gain in their IoT functionality if classified either as wearable or portable. Large electronic devices or household appliances like smart scales (Add-on thing), smart TV (Standalone thing), smart lighting (Module device).

Shown in Figure 30. (created by the PhD author) is classification by physical size. It depicts things proposed dimensions against the adult human body. The dimensions could greatly vary against children for fixed-size things, but portable and wearable things should still be comfortable to carry for all.

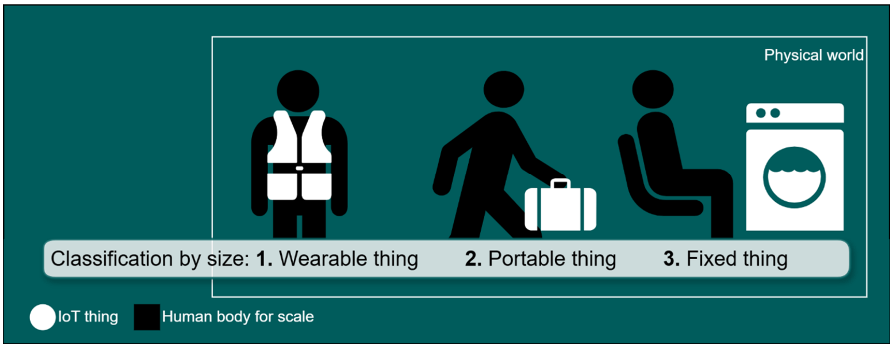


Fig. 30. IoT things classification by size

Last but not least is the classification by communication technology. This is perhaps the most overlooked classification, as it could have been implemented since the dawn of the Internet of Things. IoT things can function with the use of already existing technologies and protocols like IP/TCP and Ethernet, or use specifically for them defined ones like Thread and Matter. Either way, if accounted this classification could allow narrowing down the previously mentioned attributes list to those that are possible to be implemented according to the technology. In fact, these technologies already have their structural placement in the physical world.

6.4.Structural placement

Communication technologies, especially those that are in use for IoT, can be categorized into two big groups – wired and wireless. Both are widely used and can be applied separately or together for each of the previously mentioned groups of IoT things.

Wired technologies are commonly reserved for backbone networks like optical fiber lines connecting two base stations, or PoE (Power over Ethernet) current delivery for end devices at the industrial or enterprise facility.

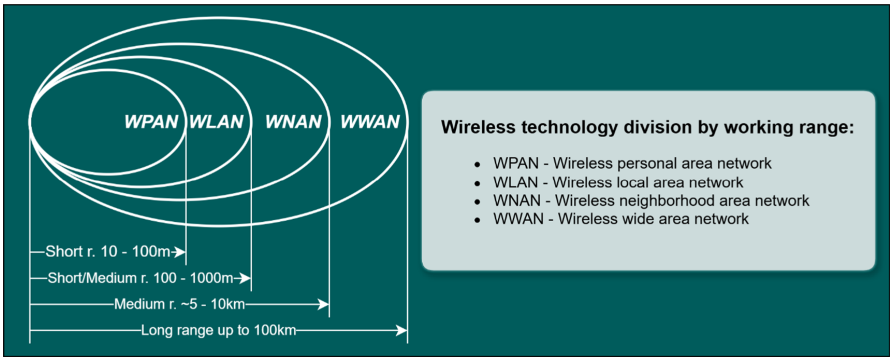

WPAN
WLAN
WNAN
WWAN
Short r. 10 - 100m
Short/Medium r. 100 - 1000m
Medium r. ~5 - 10km
Long range up to 100km
Wireless technology division by working range:
WPAN - Wireless personal area network
WLAN - Wireless local area network
WNAN - Wireless neighborhood area network
WWAN - Wireless wide area network

Fig. 31. Wireless communication ranges

Wireless technologies are widely used in plug-and-play (PnP) scenarios like household appliances, wearables, and wireless payment systems. Wireless technologies are mostly categorized by two parameters – range of reach in physical distance (shown in Figure 31.) (created by the PhD author) and radio frequency bands they occupy (shown in Figure 32) (created by the PhD author).

Distinguishing the thing's structural placement is important, as data that is collected must be processed in large amounts. Data management capability of the IoT can be enhanced by big data technologies for transferring, storing, processing, validating, and querying IoT data more efficiently, as well as for extracting information and actionable knowledge from IoT data. Besides, additional capabilities for the integration of big data technologies are also required. The required additional capabilities for the integration of big data technologies are the capabilities of adopting big data collection, adopting big data aggregation, adopting big data storage, adopting big data integration, and adopting big data analysis [54].

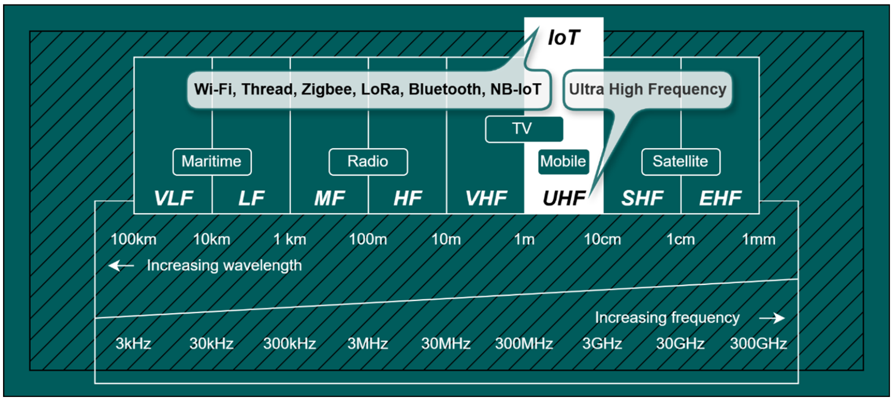


Fig. 32. Wireless communication on frequency bands

6.5.Conclusion

The scientific method used in this research is the conduct of a literature analysis for the deduction of a causal relation among the Internet of Things framework definitions and their implementation shortcomings.

This research has shown that the IoT concepts model is too generalized and therefore lacks a clear vision of what an IoT thing is. Therefore, the industry has become overwhelmed with things that lack the ability to interoperate with one another.

This research has also journaled the vast number of attributes a thing should be in compliance with, but only the reachability over the Internet and means of its identification are obligatory. **It is proposed that things should be classified for the elimination of raised issues by their dimensions, range of work, and primary function. The author's chosen classification is a proposal at a time when a problem of no classification is self-evident but not yet addressed.**

7. THREAD MESH-NETWORK DENSITY CONSIDERATIONS

This section is an overview of research done in Appendix 5 and a continuation of previously made statements, which are as follows:

- A generalized problem in networking is addressed - Fragmentation mitigation of the Internet of Things while facilitating the use of unified communication standards for the rise of compatibility among devices in use.
- An adaptive solution proposed and covered in this section - Recommendation on IoT Thread mesh network density compliance.
- Some excerpts of technology introduction from section 1.5. - Thread technology is a data-link layer that allows connecting local area network devices in a mesh network. It is an IoT communication mechanism that operates with the use of IP routing. There are three types of nodes – those that do not sleep and operate as routers, those that do not sleep but do not operate as routers, and those that sleep and wake up only to transmit or receive data.

7.1. Vertigo of Thread

Thread technology bases its communication on the creation of personal area networks (PANs). Each PAN consists of routing-capable end devices that create one tier of network and their child devices that only speak with their parents and create the second tier. Thread network is built to sort of withstand link disjoints, but the process of regeneration is not well thought out.

On a link breakage stray node initially calls for a parent to adopt it, but if it does not find any, it generates a new PAN for itself. Once the initial set of all nodes can potentially be rejoined, there is a problem. Now there are two sets of PANs. To return to a single PAN, a regeneration process must be enacted in all nodes. This means that even those nodes that hadn't suffered from any link breakages must now terminate their existing connection to join the new PAN.

This research explored the Thread mesh-network by changing its density and evaluated the damage that link breakage does to each setup. The purpose is to mitigate the effect of multiple regenerations of the thread.

7.2. Design goals

As Thread is designed for small local area networks like residential homes or small enterprise buildings, their IoT equipment doesn't really change its location. For example, large household applications like ovens, refrigerators, and washing machines are fixed nodes. The same goes for smart blinds, smart screens, lighting, or conditioners are all built-in equipment that stays fixed in its place. A single router will assume the leader role for certain functions in the thread network. This leader is required to make decisions within the network. For example, the leader assigns router addresses and allows new router requests. The leader role is dynamically elected, and if the leader fails, another router assumes the role. It is this autonomous operation that ensures there is no single point of failure [55].

In a thread network, the leader collects, collates, and distributes information about border routers and other servers available on the network back to the thread network. Individual servers use the thread management framework (TMF) messages to inform the leader of their capabilities [56].

The research focus was put on static nodes. If the nodes are to be static, then the density of the mesh network cannot be manipulated with a change of node count in the network. Therefore, to gain a difference in mesh network topology, the number of routing-capable nodes became the variable element. The second variable was the events that took place during the simulation runtime. Four different scenarios were played out. In three of them, the end user's traffic was present, but in one, it was absent. And in one, only a single node failed, but in another, two nodes failed. In this manner, it was deducible what the minimum requirement for a stable thread mesh network was.

7.3.Simulation of the Thread network with various mesh densities

The simulations were run in the OpenThread network simulator (OTNS). OpenThread is a version of Thread that is free to use and maintained by Google. Simulated topology consisted of 17 network nodes and is shown in all three variants in Figure 33. (created by the PhD author).

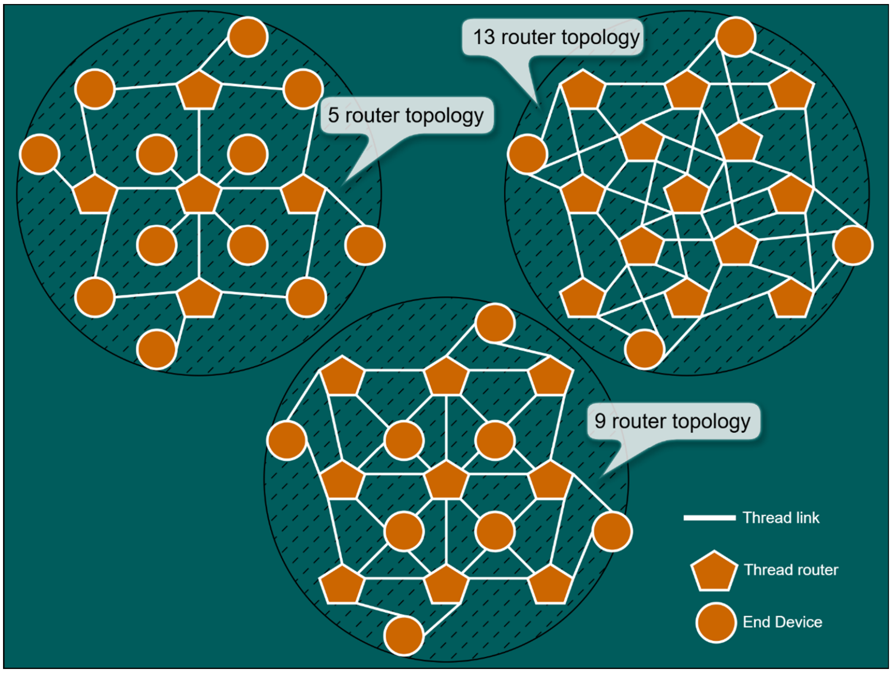


Fig. 33. Simulated thread network topologies

Each topology had been run under these four scenarios:

1. Silent network – only traffic that is in place is generated by the Thread to maintain link connectivity;
2. Active network – there is thread traffic, and additionally to that, an end-user traffic is generated;
3. Single node failure – one of the central routers is turned off to imitate a node failure that causes link disjoints;
4. Two router failures – two central routers are turned off to imitate an even bigger node failure and, in some cases, cause topology destruction.

Multiple reruns of each of the scenarios are done to obtain a plausible result of the thread response pattern. This pattern varied among the topologies shown in Figure 33.

In Fig. 33. all shown topologies differ in link count. The link count is the attribute that makes one topology mesh denser than the other. The 13-router topology has the most links as it only has four end nodes. On link breakage, this topology is the most sturdy of all, considering that each router has at least two more connections with the rest of the topology.

7.4. Results and conclusions

The scientific method used in this research is the conduct of an experiment that is an emulation of network topology for the evaluation of the impact of link count on personal area regeneration frequency. From the gathered data, mean points and their linear regressions of learned values were calculated. The function of linear regression was chosen as fitting for observation of non-simulated values, as measured thread network activity is directly related to node roles, data transfer, and node liveliness. Results in the graph are shown in Figure 34. a). (created by the PhD author).

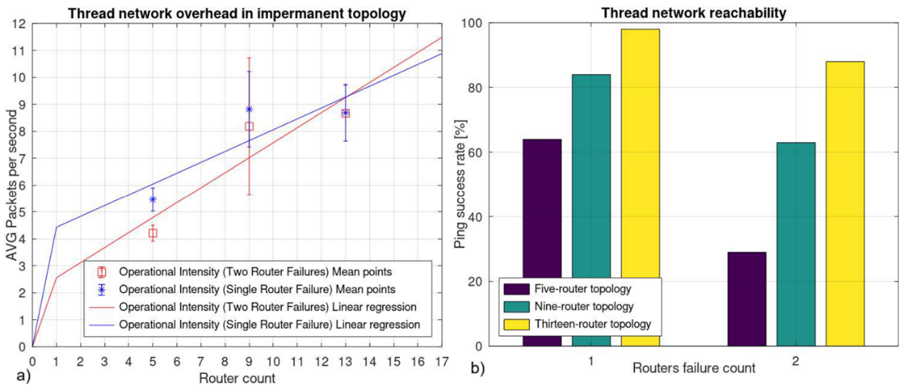


Fig. 34. Thread network emulation results:
a) Thread network overhead in an impermanent topology;
b) Thread network reachability

Measured emulated Thread network topologies' reachability is shown in Figure 34. b) where value 1 is a single router disconnect, but 2 is two router disconnects. Even though only three topology types were emulated, due to the Thread mesh working principles, it is enough to obtain appropriate emulation results for other cases where the device count is bigger. Reachability was measured by observation of the effect caused by the router's disconnection from the mesh. Even with 2 central routers disconnected 13 router topology is capable of maintaining connections between transmitting and receiving network nodes with the use of backup paths. **The result of the experiment, in the case of the 13 router topology, is approximately 84% successful Ping data packet delivery, which equals at least 500 pings received in a time frame of 10 min.**

The result of this work is a proposed recommendation – locate Thread network devices at a distance that allows for at least two backup connections to be formed for each end device. This density allows for dodging personal area network regenerations on link breakage (cases when routing-capable devices are disconnected from the network) as end devices have a sufficient count of backup paths.

8. ANALYSIS OF INTENT-BASED NETWORKING BUILDING BLOCKS

This section is an overview of research done in Appendix 7 and a continuation of previously made statements, which are as follows:

- A generalized problem in networking is charted - Automation of human-managed network configuration, maintenance, and telemetry analysis by provision of intent-based networking.
- An adaptive solution proposed and covered in this section - Recommendation for facilitating intent-based networking adaptation.
- Some excerpts of technology introduction from section 1.8. - Intent-based networking (IBN) is automated network management. The operator only needs to state the necessary outcome without the direction on how to achieve it. Software-defined network is one of many underlay technologies used to make artificial intelligence and machine learning capable of overseeing a communication network.

This research explored industries and academic achievements in the implementation and standardization of IBN. Its building blocks and their relation to one another are determined. From these discoveries, it was concluded that the overly complex task of management itself limits IBNs' adaptation. The following are key areas for network maintenance that can be used to transform insights into value:

- Automated - provide tools for automatically aggregating the network data and setting up automatic tasks for network maintenance.
- Reactive - dimension issues and find root causes when network faults are detected.
- Proactive - actively probe the network status and predict possible network performance degradation at an early stage [57].

8.1.Profiling of input

It has been stated that a customer's service request is (or should be) technology agnostic. That is, a customer is unaware of the technology that the network operator has available to deliver the service, so the customer does not make requests specific to the underlying technology but is limited to making requests specific to the service that is to be delivered [58].

Meaning, an intent is to be abstract, as its initiator should not make any assumptions of having the ability to directly manipulate the network's performance and direct resources towards a specific task with no regard for other users.

Profiling is the building block that deals with user input. Via different mechanisms of gathering data in conjunction with artificial intelligence (AI) profiling building block generates an abstract that is a logical value of the user's intent that holds enough information for network policy generation. Profiling is shown in Figure 35. (created by the PhD author).

IBN users don't have an equal technical knowledge base. The policy continuum defines the set of actors that will create, read, use, and manage policy. Each set of actors has its own terminology and concepts that they are familiar with. This captures the fact that business people do not want to use CLI, and network operations center personnel do not want to use non-technical languages [59].

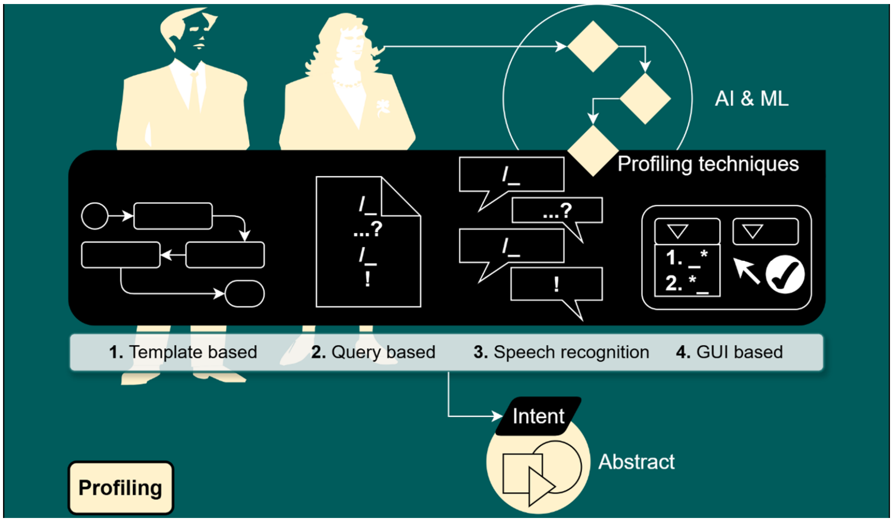


Fig. 35. Profiling of the user's request

8.2. Translation of abstract

From abstract translation building block (shown in Fig. 36.) (created by the PhD author) generates an applicable network policy (compliant with network management). It consists of rules with defined action, event (on which to do the action), and conditions (to recognize the event).

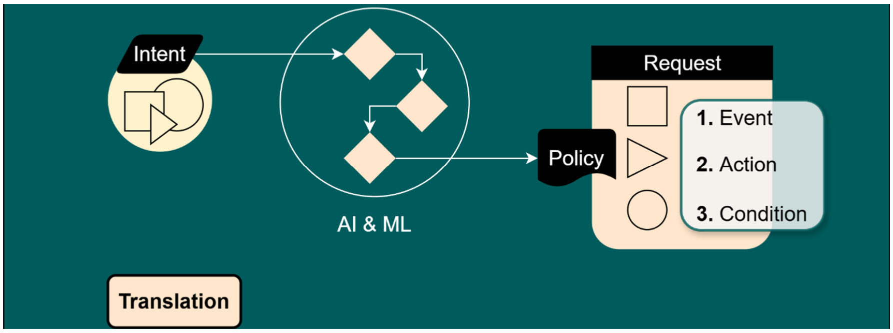


Fig. 36. Translating intent into policy

Each policy rule is a declarative statement consisting of a Boolean expression that describes a situation to which the policy applies. When the expression is true, one set of actions is initiated, and when false, a different set of actions is initiated [60].

8.3. Resolution of policy

Resolution building block (shown in fig. 37.) (created by the PhD author) makes sure that the policy which is to be applied does not negatively affect already existing network state. On a positive result, it passes the intent in the form of an applicable configuration on to the next building block. Resolution block uses AI analyses for intent examination. When it comes to failures, sometimes it is difficult to quickly find alternatives for affected intents. In this case, a prioritization scheme can be used, by first refining intents that have expressed high reliability and fast recovery in case of a failure. Recovery-based prioritization can work well on network failure, and the intent explicitly provides this information. But, in case of a performance intent drift, a QoS-based prioritization scheme may be followed [61].

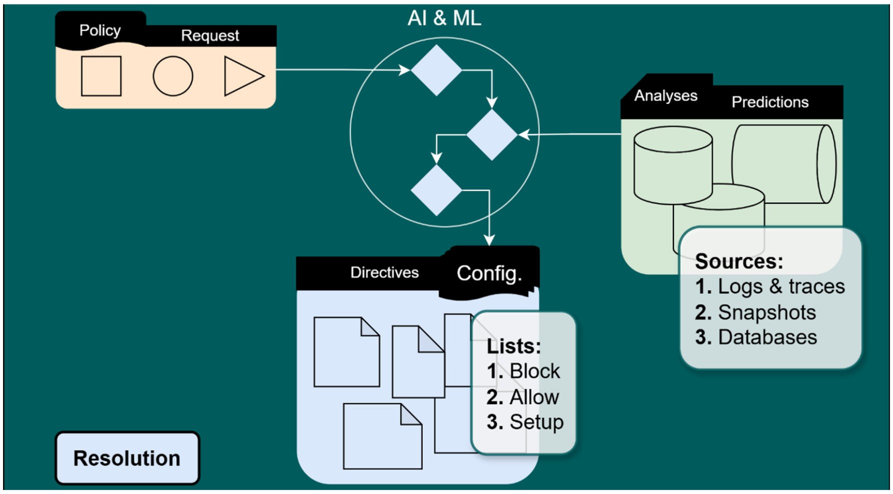


Fig. 37. Resolution of intent's footprint

8.4. Activation of configuration

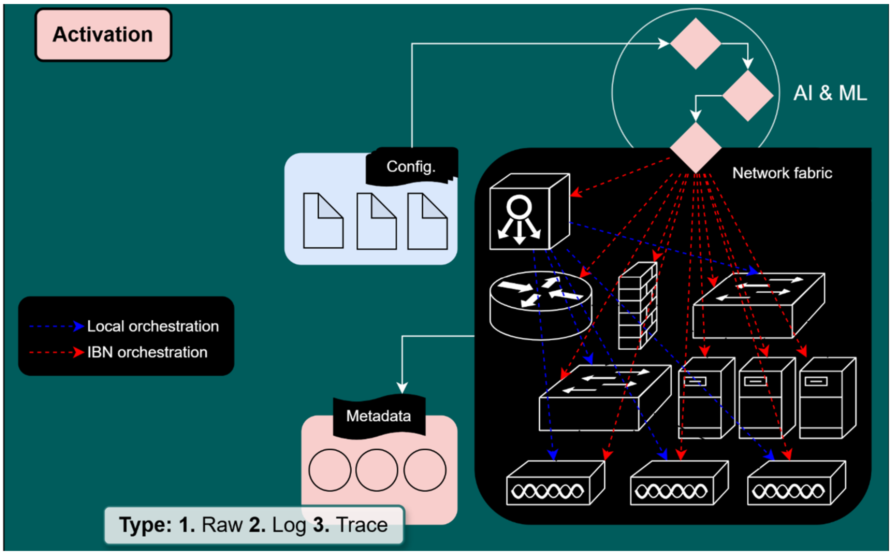


Fig. 38. Actuating the necessary modifications

Activation building block (shown in Figure 38.) (created by the PhD author) is where configuration takes effect on the network fabric. Once it is applied, the change in network state is snapshotted and compared with an earlier result. This allows to catch whether an intent drift has occurred. For monitoring, telemetry is gathered on a regular basis as a management task, which includes actions as handover, disaster recovery, data backup, and so on. From the network fabric's point of view, IBN does a centralized orchestration. Although centralization can lead us to think of the single-point-of-failure concept, it is not the case for IBN. Conceptual centralization highly differs from centralized deployment. It brings all the benefits of having a single point of decision, but retaining the benefits of distributed systems. For instance, control decisions in underlay SDN can be centralized, while the mechanisms that enforce such decisions into the network (SDN controllers) can be implemented as highly distributed systems. The same approach can be applied to NFV. NFs can be implemented in a central computing facility, but they can also take advantage of several replication and distribution techniques to achieve the properties of distributed systems [62].

8.5. Assurance of everything

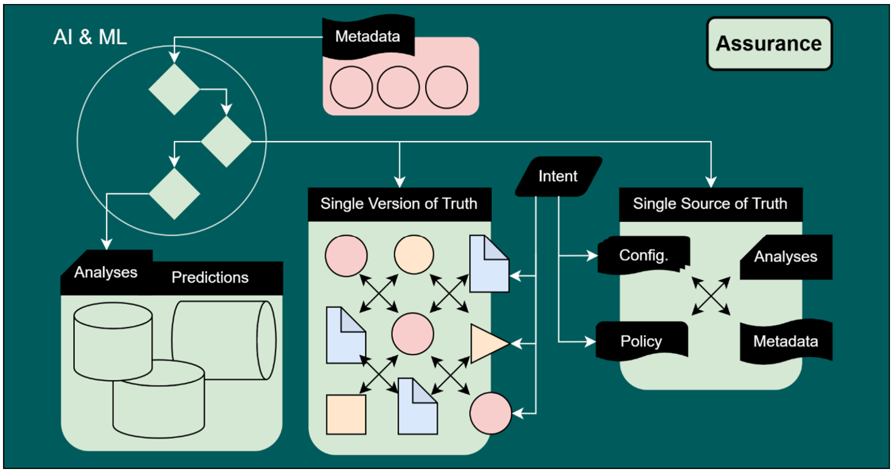


Fig. 39. Assurance of operational successfulness

Assurance is the building block that makes intent fulfillment possible. Intent fulfillment is concerned with the functions that take intent from its origination by a user (generally, an administrator of the responsible organization) to its realization in the network [63].

Assurance is the heart of intent-based networking. It is the block that delivers an analysis of the existing network similar to human made one. It not only collects telemetry but also makes predictions on what's to come based on previous experiences and, therefore, can act proactively. AI that plans ahead and outreaches for operators' assistance if necessary. Its main goal is ensuring network security and scalability, and continuity. Unfortunately, such capable AI has not yet been born. The assurance building block is shown in Figure 39. (created by the PhD author). Due to the fact that the complexity of networks is increasing significantly. This brings unprecedented challenges to the operation, administration, and maintenance (OAM) for IBN, such as:

- The network architecture is more flexible, the network deployment scenarios are diversified, and the user requirements are personalized, thus making networks more dynamic and more complex to manage.

- The introduction of diversified terminals, such as in the Internet of Things (IoT), has led to growth in the number of user equipment (UE) by orders of magnitude, thus the cost of network management tends to increase.
- The use of manual decision-making mechanisms makes it difficult to quickly understand any change in network conditions and to quickly adapt the network to such changes.
- Mechanisms for utilizing and analyzing the large amount of network data generated daily need to be deployed and operated in the networks, in order to produce value.
- The extensive use of network connectivity among UEs and things leads to high requirements in terms of security and the protection of data being exchanged through networks.
- The decoupling of software from the hardware in networks requires efficient management solutions to realize high-frequency updates of software being used in these networks.
- Insufficient openness of network capabilities limits the efficient utilization of network resources and the enhancement of user value [64].

8.6. Lifting the burden

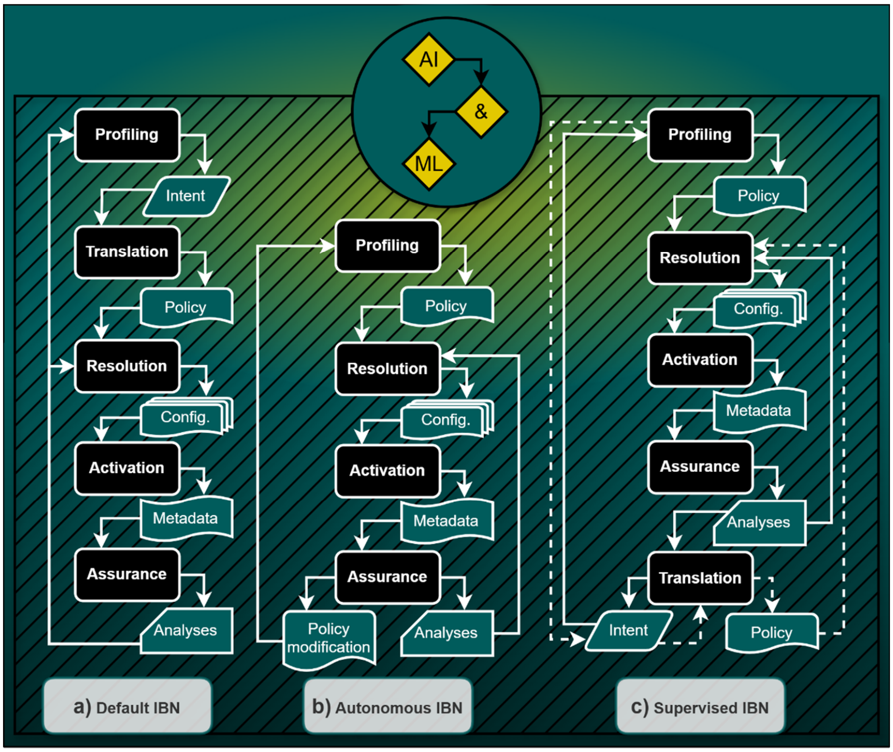


Fig. 40. Intent-based networking models:
a) default IBN model;
b) IBN model for autonomous networks;
c) IBN model for supervised networks

One way to ease data collection is to not collect everything, but only what is needed. An important aspect is the way and frequency the monitoring data are collected. Usually, a publish-subscribe model is envisioned, where only relevant data from network elements in key points of presence (POPs) is gathered. These data could follow a periodic, an event-driven, or a poll-based approach in order to collect a sufficient amount of knowledge of the network state [65].

To further fast-track the adaptation of IBN into a control overlay of enterprise networks, it is determined that a change in the model's concept needs to be made. This research has drawn out the existing IBN schema and two possible derivatives. Schemas are shown in Figure 40. (created by the PhD author).

The existing IBN model's structural schema is a) in Figure 40. It shows that the control is realized with the use of a two-loop system. The biggest loop facilitates the whole lifespan of an intent. The smaller one brings the assurance capability to life.

Proposed in this research is the idea of adjusting the given model to two types of networks – those that require supervision of a human operator, and autonomous networks that do not. Autonomous networks IBN model b) in Figure 40. has an input of initial policy, which it works upon when it detects that an upgrade could benefit the network, and, in a manner of suggestion, it returns a policy modification request to the profiling block for human revision and application into the policy. This level of autonomy allows for to escape of ever having an intent drift due incompatibility of existing policy and incoming intents.

Supervised, on the other hand, c) in Figure 40. holds an ability to work with incoming intents, but it also allows for the operator to input network policy changes. This allows to escape of the need for the AI to be able to perform network maintenance at an unprecedented event by giving full operational control to the operator.

8.7.Conclusion

The scientific method used in this research is the conduct of a literature analysis for the deduction of a causal relation among the Intent-based networking model building blocks and their implementation obstacles.

Intent-based networking has a strong reliance on artificial intelligence and machine learning capabilities. IBN is a network management technology that is made to mitigate human intervention in network creation and oversight by the use of artificial intelligence and machine learning (ML). From intent profiling with natural language processing (NLP), to network availability foresight via big data and generative adversarial networks (GAN) [66,67].

Unfortunately for the industry of networking, these systems haven't made a big leap in autonomy just yet, as the pool of data sets is not deep enough to train them. Therefore, an effective management system that could take the policy-based network management (PBNM) into its own hands has not been developed. **To ease the assurance task, the IBN could be introduced separately to fit the needs of different networking areas rather than focusing on having a universal solution.**

CONCLUSIONER

In this dissertation, 8 joint research works are developed. All of which are done in packet-switched networks for an enhancement of the following telecommunication technologies:

- Software-defined networking (SDN);
- Network function virtualization (NFV);
- Service function chaining (SFC);
- Programmable protocol-independent packet processing (P4);
- Intent-based networking (IBN);
- The Internet of Things (IoT).

From the conducted research results, multiple important aspects are concluded about the working principle of studied technologies, that are pointing to shortcomings in them, possible optimization, and ways for the use of new solutions to abolish the shortcomings.

It has been concluded that by default, the service function chaining (SFC) domain holds multiple shortcomings – it does not define conditions for encapsulation application, it functions only thanks to a predefined manual service function (SF) path policy, and it has no definition of functionality for multihomed tenancy. For the mitigation of stated shortcomings, solutions are proposed – network topology-aware SFC encapsulation, reactive SF path policy creation, and SF share in inter-data center networks.

It has been found that, by default, the Internet of Things (IoT) defines 11 various attributes (reachability, security, identification, continuity, etc.), without any connection to physical device parameters (dimensions, positioning, data transmission type, etc.), thus fragmenting interconnectivity among devices. Partially, this problem is mitigated by Thread and Matter technologies that enable centralization of device management in a small local area network, but Thread itself suffers from link breakages that are started by its network regeneration process. Solutions proposed are – a recommendation for IoT device categorization, and a recommendation for Thread network density.

It has been concluded that programmable switch architecture (PSA) enables virtual realization of network functions, but program code needs to be developed for the definition of desirable functionality. By creation of SFC functionality on top of PSA, it has also been concluded that SF paths made by use of segment routing with MPLS (SR-MPLS) have dependency on encapsulation label count. A program code is developed for SFC creation on PSA via SR-MPLS encapsulation [10], and by evaluation of experiment results, a recommendation is proposed for SF path declaration by use of a single segment identifier.

It has been concluded that the intent-based networking (IBN) model consists of multiple building blocks – profiling, translation, resolution, activation, and assurance block - each of which is realized with the use of artificial intelligence. An assurance block that is committed to the provision of a network state forecast can not be realized, as existing artificial intelligence functionality is incapable of accomplishing the given task. To ease AI given task prediction generation, 2 alternative IBN models are proposed.

The work objective (make an analysis of existing solutions and propose new adaptive solutions for telecommunications technologies) and its related tasks result in the development and evaluation of 7 adaptive solutions for 4 telecommunication problems.

Achieved results are ordered according to hypotheses:

1. **Reactive SF path discovery and coordination of transmitted information along multiple administrative domains is achievable via the use of service function chaining.** Hypothesis derives from the problem of enhancing network path generation and coordination of transmitted information in them with the use of service function chaining. The proposed solutions are:

1.1. A service function path encapsulation according to the necessity for service function chaining.

Two data packet encapsulation methods are developed in this work. They allow to impose an encapsulation only in service chain segments where it is necessary, in opposite to encapsulation in all segments of the chain. It allows to decrease overhead (a 50% overhead decrease is achieved in emulated network topology that is 38 bytes of overhead for single packet versus 16 bytes as shown in fig. 14. for NSH MD-Type 2 with context encapsulation), enables multihomed tenancy, and removes service path limitations (segment count dependency from header, SF path steering only within a single administrative domain).

[Solution results from research in Appendix 1. Its explanation is in section 2. Background technologies are introduced in Sections 1.2. and 1.3.]

1.2. A reactive service path discovery with the use of a default path and data flow reclassification.

Solutions developed in this work are a combination form the usage of the default path and network data packet re-classification, in opposition to predefined policy. This mitigates human intervention in service function path creation. **The author's proposed solution is capable of subsequent SF path discovery for single-conditioned policy-steered network traffic, while the existing solution is not.**

[Solution results from research in Appendices 2. and 3. Its explanation is in section 3. Background technologies are introduced in Sections 1.1. and 1.3.]

1.3. Service function share in inter-datacenter networks.

Solution developed in this work allows steering a single multipath TCP (MPTCP) belonging to sub-flows via different service functions (SFs), in opposition to single SF usage. The SF share can be put in use in inter-data center networks to increase overall connection throughput (in an emulated network topology, a 60% throughput increase was measured, that is 8 Mbit/s of throughput versus 14 Mbit/s for AVG (average of all flows) value in fig. 23.).

[Solution results from research in Appendix 6. Its explanation is in section 4. Background technologies are introduced in Sections 1.1. and 1.3.]

2. **Fragmentation of the Internet of Things can be mitigated with the use of a unified transmission medium, and device compatibility can rise with the use of a unified application layer, if more specific characteristics of things are defined.** Hypothesis derives from the problem of fragmentation mitigation of the Internet of Things by the use of unified communication standards for the increase of compatibility among devices in use. The proposed solutions are:

2.1. Recommendation on the implementation of specific characteristics for IoT things definition.

IoT mesh is a vertically fragmented system due complexity of the protocol stack, variation in data format, and multiple communication procedures [68]. Therefore, developed in this work are recommendations for categorization of IoT things according to their physical size **(wearable, portable, fixed)**, and according to IoT functionality role in relation to object's main purpose fulfillment **(standalone, add-on, module)**, and according to communication parameters **(what physical signal it can transmit)**. The author's proposed classification allows overcoming the fragmentation of the network relating to a lack of interconnectivity among devices.

[Solution results from research in Appendix 4. Its explanation is in section 6. Background technology is introduced in Section 1.4.]

2.2. Recommendation on IoT Thread mesh network density compliance.

Developed in this work is a recommendation stating that it is necessary to locate Thread network devices at a distance that allows for at least two backup paths to be created. This density decreases personal area network regeneration frequency on connection loss (in the emulated network topology selected density provided connection continuity of 84% which equals over 500 pings received in a period of 10 min. at 2 central routers disconnect by measuring packet loss). **There is no set recommendation for link count besides that which is proposed in this research.**

[Solution results from research in Appendix 5. Its explanation is in section 7. Background technology is introduced in section 1.5.]

3. **Service function chaining via segment routing can be built on top of programmable devices, thus serving as an alternative to the use of purpose-specific or proprietary devices with similar capabilities.** Hypothesis derives from the problem of the cut on the use of task-specific or proprietary devices by the advancement of the feature set for re-programmable units. The proposed solution is - **Service function path declaration with designated segment identifier in segment routing. Also author's built software can be reused as an alternative to proprietary solutions with similar capabilities.**

   In this work, a software for SFC via SR-MPLS on P4 switches was developed and evaluated, and is available in a repository [10]. It can be concluded that the SF path declaration needs to be made by use of a single MPLS label for all of the SF paths, if transport is segment routing. This allows for dodging the increasing overhead for longer SF paths (in emulation, a 40% increase is observed when the SF path with 8 hops and the SF path with 12 hops are compared, that is 11000 bit/s versus 17000 bit/s of overhead in Fig. 28.).

   [Solution results from research in Appendix 8. Its explanation is in section 5. background technologies are introduced in Sections 1.6. and 1.7.]

4. **Computer network creation and management can be automated by the use of intent-based networking if artificial intelligence functionality is delegated separately to different network management types.** Hypothesis derives from the problem of automation of human-managed network configuration, maintenance, and telemetry analysis by provision of intent-based networking. The proposed solution is - **Recommendation for facilitating intent-based networking adaptation.**

In this work, a split of the IBN model into two sub-models is proposed. One for autonomous networks, but the other for supervised networks. The division would ease the tasks of assurance block (an action set that, with the use of artificial intelligence, provides analysis and forecast of network performance). **Studied references of Appendix 7 (works of other authors) strongly suggest the need for ease of tasks of the assurance block. The proposed solution is yet to be tested.**

[Solution results from research in Appendix 7. Its explanation is in section 8. Background technology is introduced in section 1.8.]

Developed and evaluated adaptive solutions are the result of multiple finished research works. Hypotheses defined in this dissertation have been found to be true via the conduct of analysis and experimental evaluation by emulation of proposed solutions application for given problems in various telecommunication technologies. This dissertation and its results together can be used as a forging ground for future research.

REFERENCES

[1] United Nations General Assembly, Transforming our world: the 2030 Agenda for Sustainable Development, A/RES/70/1, October 21 2015, [Online], Available: https://sustainabledevelopment.un.org/post2015/transformingourworld, last viewed May 15, 2025

[2] National Physical Laboratory of the United Kingdom, "Packet Switching: The First Steps on the Road to the Information Society." [Online], Available: https://www.npl.co.uk/getattachment/about-us/History/Famous-faces/Donald-Davies/UK-role-in-Packet-Switching-(1).pdf.aspx, last viewed May 15, 2025

[3] Mihaeljans, M., Skrastiņš, A. Network Topology-aware Service Function Chaining in Software Defined Network. In: 2020 28th Telecommunications Forum (TELFOR 2020): Proceedings, Serbia, Belgrade, 24-25 November, 2020. Piscataway: IEEE, 2020, pp.1-4. ISBN 978-1-6654-0500-3. e-ISBN 978-1-6654-0499-0. Available from: doi:10.1109/TELFOR51502.2020.9306554

[4] Mihaeljans, M., Skrastiņš, A. Reactive Service Function Path Discovery Approach in Software Defined Network. In: 2021 29th Telecommunications Forum (TELFOR 2021): Proceedings, Serbia, Belgrade, 24-24 November, 2021. Piscataway: IEEE, 2021, pp.29-32. ISBN 978-1-6654-2586-5. e-ISBN 978-1-6654-2585-8. Available from: doi:10.1109/TELFOR52709.2021.9653356

[5] Mihaeljans, M., Skrastiņš, A. Evaluation of Reactive Service Function Path Discovery in Symmetrical Environment. Telfor Journal, 2022, Vol. 14, No. 1, pp.2-7. ISSN 1821-3251. e-ISSN 2334-9905. Available from: doi:10.5937/telfor2201002M

[6] Mihaeljans, M., Skrastiņš, A. IoT Concept and SDN Fusion in Consumer Products: Overview. In: 2023 3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME 2023), Spain, Tenerife, 19-21 July, 2023. Piscataway: IEEE, 2023, pp.1-6. ISBN 979-8-3503-2298-9. e-ISBN 979-8-3503-2297-2. Available from: doi:10.1109/ICECCME57830.2023.10252518

[7] Mihaeljans, M., Skrastiņš, A. Efficient Multipath Service Function Chaining in Inter-Data Center Networks. In: 2023 31st Telecommunications Forum (TELFOR 2023): Proceedings, Serbia, Belgrade, 21-22 November, 2023. Piscataway: IEEE, 2023, pp.1-4. ISBN 979-8-3503-0312-4. e-ISBN 979-8-3503-0313-1. Available from: doi:10.1109/TELFOR59449.2023.10372615

[8] Mihaeljans, M., Skrastiņš, A. Openthread Network Density Evaluation: Quantitative Analysis. In: 2023 Symposium on Internet of Things (SIoT 2023): Proceedings, Brazil, São Paulo, 25-27 October, 2023. Piscataway: IEEE, 2023, pp.1-5. ISBN 979-8-3503-2944-5. e-ISBN 979-8-3503-2943-8. Available from: doi:10.1109/SIoT60039.2023.10390236

[9] Mihaeljans, M., Skrastiņš, A., Poriņš, J. Service Function Chaining via SR-MPLS over P4: A Rudimentary Analysis. In: 2024 International Conference on Intelligent Computing, Communication, Networking and Services (ICCNS 2024): Proceedings, Croatia, Dubrovnik, 24-27 September, 2024. Piscataway: IEEE, 2024, pp.1-8. ISBN 979-8-3503-5470-6. e-ISBN 979-8-3503-5469-0. Available from: doi:10.1109/ICCNS62192.2024.10776149

[10] M. Mihaeljans and A. Skrastins, P4 program code used for SFC via SR-MPLS on PSA, [Online], Available: https://github.com/MartinsMihaeljans/SFC-on-P4/tree/main, last viewed June 2024

[11] Mihaeljans, M., Skrastiņš, A., Poriņš, J. Paramounts of Intent-Based Networking: Overview. Elektronika ir elektrotechnika = Electronics and Electrical Engineering, 2024, Vol. 30, No. 6, pp.53-59. ISSN 1392-1215. e-ISSN 2029-5731. Available from: doi:10.5755/j02.eie.38680
[12] E. Haleplidis, K. Pentikousis, S. Denazis, J. H. Salim, D. Meyer, and O. Koufopavlou, Software-Defined Networking (SDN): Layers and Architecture Terminology, RFC 7426, p. 12, Jan. 2015.
[13] ITU-T, Framework of Software-Defined Networking, Recommendation Y.3300, p. 14, June 2014.
[14] R. Boutaba and I. Aib, "Policy-Based Management: A Historical Perspective," Journal of Network and Systems Management, vol. 15, no. 4, pp. 447–480, Dec. 2007.
[15] ONF, OpenFlow Switch Specification, Version 1.4.0 (Wire Protocol 0x05), ONF TS-012, p. 15, Oct. 14, 2013.
[16] ONF, Software-Defined Networking: The New Norm for Networks, ONF White Paper, p. 5, Apr. 13, 2012.
[17] ETSI, Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV, ETSI GR NFV 003 V1.5.1, p. 11, Jan. 2020.
[18] ONF, SDN Architecture, Issue 1.1, ONF TR-521, p. 10, 2016.
[19] P. Quinn, U. Elzur, and C. Pignataro, Network Service Header (NSH), RFC 8300, p. 22, Jan. 2018.
[20] ITU-T, Service Function Chaining in Mobile Networks, Recommendation Y.2242, p. 12, Dec. 2018.
[21] Connectivity Standards Alliance, Matter Device Library Specification R1.3, Document 23-27351, p. 27, Apr. 17, 2024.
[22] ITU-T, Requirements of the Network for the Internet of Things, Recommendation Y.4113, p. 12, Sep. 2016.
[23] ITU-T, Common Requirements for Internet of Things (IoT) Applications, Recommendation F.748.0, p. 12, Oct. 2014.
[24] Connectivity Standards Alliance, Matter Overview, p. 2, 2024.
[25] Thread Group, Thread Technical White Paper: Thread Usage of 6LoWPAN, p. 6, Jul. 2015.
[26] Thread Group, Elegantly Connecting Your Smart Home Network, Thread Technical White Paper, p. 7, Sep. 2023.
[27] Connectivity Standards Alliance, Matter Specification R1.3, Document 23-27349, p. 49, Apr. 17, 2024.
[28] Connectivity Standards Alliance, Matter Application Cluster Specification R1.3, Document 23-27350, p. 106, Apr. 17, 2024.
[29] The P4.org Applications Working Group, P416 Language Specification, Version 1.2.4, p. 12, May 15, 2023.
[30] The P4.org API Working Group, P4Runtime Specification, Version 1.3.0, p. 9, Dec. 1, 2020.
[31] E. Rosen, A. Viswanathan, and R. Callon, Multiprotocol Label Switching Architecture, RFC 3031, p. 4, Jan. 2001.
[32] Z. Ali, C. Filsfils, P. Camarillo, D. Voyer, S. Matsushima, R. Rokui, A. Dhamija, and P. Maheshwari, Building Blocks for Network Slice Realization in Segment Routing Network, Internet-Draft, TEAS Working Group, p. 4, Sep. 7, 2022.
[33] M. Stevens, W. Weiss, H. Mahon, B. Moore, J. Strassner, G. Waters, A. Westerinen, and J. Wheeler, Policy Framework, Internet-Draft, Expires Mar. 2000, p. 18, Sep. 13, 1999.

[34] J. François, A. Clemm, D. Papadimitriou, S. Fernandes, and S. Schneider, Research Challenges in Coupling Artificial Intelligence and Network Management, Internet-Draft, draft-irtf-nmrg-ai-challenges-03, p. 20, Mar. 4, 2024.
[35] ITU-T, Architectural Framework for Machine Learning in Future Networks Including IMT-2020, Y.3172, p. 10, Telecommunication Standardization Sector of ITU, Jun. 2019.
[36] ETSI, Network Functions Virtualisation – Introductory White Paper Issue 1, p. 11, 2013.
[37] ITU-T, Deployment Models of Service Function Chaining, Supplement 41, Series Y, p. 12, Jul. 2016.
[38] S. Aldrin, C. Pignataro, N. Kumar, R. Krishnan, and A. Ghanwani, Service Function Chaining (SFC) Operations, Administration, and Maintenance (OAM) Framework, RFC 8924, p. 16, 2020.
[39] ETSI, Network Functions Virtualisation (NFV); Use Cases, ETSI GS NFV 001 V1.1.1, p. 21, Oct. 2013.
[40] O. Bonaventure, C. Paasch, and G. Detal, Use Cases and Operational Experience with Multipath TCP, RFC 8041, p. 12, Jan. 2017.
[41] A. Ford, C. Raiciu, M. Handley, O. Bonaventure, C. Paasch, TCP Extensions for Multipath Operation with Multiple Addresses, RFC 8684, p. 17, 2019.
[42] ETSI, Network Functions Virtualisation (NFV); Management and Orchestration, ETSI GS NFV-MAN 001 V1.1.1, p. 100, Dec. 2014.
[43] J. Halpern and C. Pignataro, Service Function Chaining (SFC) Architecture, RFC 7665, p. 25, Oct. 2015.
[44] The P4.org Applications Working Group, Telemetry Report Format Specification, Version 2.0, p. 8, Oct. 8, 2020.
[45] The P4.org Applications Working Group, In-band Network Telemetry (INT) Dataplane Specification, Version 2.1, p. 7, Nov. 11, 2020.
[46] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir, Segment Routing Architecture, RFC 8402, p. 4, Jul. 2018.
[47] The P4.org Architecture Working Group, P4 Portable NIC Architecture (PNA), Version 0.7, p. 5, Dec. 22, 2022.
[48] C. Filsfils, K. Talaulikar, D. Voyer, A. Bogdanov, and P. Mattes, Segment Routing Policy Architecture, RFC 9256, p. 19, Jul. 2022.
[49] A. Bashandy, C. Filsfils, S. Previdi, B. Decraene, S. Litkowski, and R. Shakir, Segment Routing with the MPLS Data Plane, RFC 8660, p. 8, Dec. 2019.
[50] The P4.org Architecture Working Group, P416 Portable Switch Architecture (PSA), Version 1.2, p. 29, Dec. 22, 2022.
[51] ITU-T, Framework for Service Function Chaining in Information-Centric Networking, Recommendation Y.3073, p. 11, Aug. 2019.
[52] ITU-T, Overview of the Internet of Things, Recommendation Y.2060, p. 10, Jun. 2012.
[53] ITU-T, ITU-T Y.3170-series – Machine Learning in Future Networks Including IMT-2020: Use Cases, Supplement 55, p. 19, Telecommunication Standardization Sector of ITU, Oct. 2019.
[54] ITU-T, Functional Framework and Capabilities of the Internet of Things, Recommendation Y.2068, p. 24, Mar. 2015.
[55] Thread Group, Thread Network Fundamentals White Paper, Thread Technical White Paper, p. 6, Sep. 2022.

[56] Thread Group, Thread v1.3.0 Public Specification, p. 210, Feb. 27, 2023.
[57] ITU-T, Guidelines for Intelligent Network Analytics and Diagnostics, E.475, p. 11, Telecommunication Standardization Sector of ITU, Jan. 2020.
[58] Q. Wu, W. Liu, and A. Farrel, Service Models Explained, RFC 8309, p. 9, Internet Engineering Task Force (IETF), Jan. 2018.
[59] C. Li, Y. Cheng, J. Strassner, O. Havel, W. Xu, and W. Liu, Intent Classification, Internet-Draft, draft-li-nmrg-intent-classification-00, p. 7, Apr. 22, 2019.
[60] Wang Changkun, "Policy-based network management," WCC 2000 - ICCT 2000. 2000 International Conference on Communication Technology Proceedings, pp. 101-105 vol.1, Beijing, China, 2000.
[61] A. Leivadeas and M. Falkner, "A Survey on Intent-Based Networking," IEEE Communications Surveys & Tutorials, vol. 25, no. 1, pp. 649, First Quarter 2023.
[62] P. Martinez-Julia, S. Homma, and D. R. Lopez, Artificial Intelligence Framework for Network Management, Internet-Draft, draft-pedro-nmrg-ai-framework-04, p. 5, Oct. 22, 2023.
[63] A. Clemm, L. Ciavaglia, L. Z. Granville, and J. Tantsura, Intent-Based Networking - Concepts and Definitions, RFC 9315, p. 13, Internet Research Task Force (IRTF), Oct. 2022.
[64] ITU-T, Framework for Evaluating Intelligence Levels of Future Networks Including IMT-2020, Y.3173, p. 9, Telecommunication Standardization Sector of ITU, Feb. 2020.
[65] A. Leivadeas and M. Falkner, "Autonomous Network Assurance in Intent Based Networking: Vision and Challenges," 2023 32nd International Conference on Computer Communications and Networks (ICCCN), pp. 1-10, Honolulu, HI, USA, 2023.
[66] C. Li et al., " Intent Classification draft-li-nmrg-intent-classification-00," IETF Network Working Group, 2019
[67] S. Hares, "Intent-Based Nemo Overview draft-hares-ibnemo-overview-01," IETF Internet-Draft, 2016
[68] W. Rafique, L. Qi, I. Yaqoob and M. Imran, R. Rasool, W. Dou “Complementing IoT Services Through Software Defined Networking and Edge Computing: A Comprehensive Survey” in Communications Surveys & Tutorials Volume: 22, Issue: 3, 2020

APPENDICES

Appendix 1

M. Mihaeljans and A. Skrastins
Network Topology-aware Service Function Chaining in Software Defined Network
2020 28th Telecommunications Forum (TELFOR)
Belgrade, Serbia, 2020

Network Topology-aware Service Function Chaining in Software Defined Network

M. Mihaeljans, *Student Member, IEEE,* A. Skrastins, *Member, IEEE*

***Abstract* — In this paper, we study a way to reduce data overhead made by packet encapsulation in Service Function Chaining (SFC). SFC uses packet encapsulation for creation of network transport topology overlay to alter packet forwarding process through Service Function (SF) path. We propose two SFC encapsulation application approaches as alternatives to approach defined in SFC architecture. To examine our proposal, we used a reference SDN network in Mininet. The result of this study is a comparison between approaches of SFC encapsulation application which shows that our proposed approaches allow to reduce overall overhead. Overhead for a single packet can be reduced by one third or by half of the original amount depending on which approach has been used.**

***Keywords* — Network Service Header, Network function, Packet encapsulation, Software Defined Network, Service Function Chaining.**

I. INTRODUCTION

NETWORK Function (NF) is computer network element in which a packet processing is performed while packet is being forwarded from source to destination [1]. An example of a network function is Firewall (FW), Network Address Translation (NAT) or Deep Packet Inspection (DPI). Packet forwarding by itself is not counted as network function.

Legacy networks are static and packet steering from one NF to another in them is heavily coupled with network topology. Packet forwarding is defined on step-by-step bases because routing and switching decisions of packet steering are made locally in network devices. Therefore, new NF deployment or making changes in packet steering through existing network functions requires involvement of a network administrator to make an adjustment of static routes. Making changes in computer network is a complex process which in case of error can lead to misconfiguration or even a network down time.

In Software Defined Network (SDN) control of the network is separated from packet forwarding. A controller is used to apply changes in packet forwarding policy dynamically. Compared to the legacy network SDN is more resilient to misconfiguration because it requires less involvement in configuration of data plane network forwarding infrastructure from the administrator like flow priority or timeout setup in switches.

To ease administrator's task of packet forwarding paths creation and for allowance of dynamic path creation an architecture of Service Function Chaining (SFC) was introduced by Cisco [2]. NF's which are a part of SFC are called Service Functions (SF). SFC architecture defines principles of packet steering through SF path. A requirement of packet encapsulation is set in SFC architecture for packet forwarding from one SF to another [3]. In this paper, we study the data overhead added to packet traffic by SFC encapsulation. We provide a two-way redesigned SFC encapsulation application approach. The aim of the approach is to reduce the data overhead added by SFC encapsulation.

Martins Mihaeljans is with the Riga Technical University, Azenes str. 12 - 325, LV1010 Riga, Latvia (e-mail: martins.mihaeljans@edu.rtu.lv)
Andris Skrastins is with the Riga Technical University, Azenes str. 12 - 325, LV1010 Riga, Latvia (e-mail: andris.skrastins@rtu.lv).

II. RELATED WORKS

Study of SFC security issues in SFC inter-domain has argued that SF paths are not secured among SF's [4]. Authors of this study also came to conclusion that it would not be possible to apply security protocols like IPSec in SFC enabled domain where SFC encapsulation would be applied inside IP header. SFC for Network Function Virtualization (NFV) has been studied with intent to overcome traffic steering problems in microservices based network architecture [5]. Study of VLAN-based traffic steering for hierarchical SFC has pointed out benefits and limits of dividing SFC domain in subdomains [6]. Authors of paper by reference [7] propose a compact SFC header which is only 6 bytes short and makes no additional overhead to the packet. Authors claim that it could be encoded in packet's Source MAC address field. Authors of paper by reference [8] proposed Service Function Label protocol as an ISO OSI Layer 5 protocol which could ease traversing middleboxes.

III. PROBLEM STATEMENT

From previous studies we drove a conclusion that one of most important elements of SFC is SFC encapsulation which brings restrictions for SF path like disability to cross boundaries of single administrative domain and a definitive value of SF path's length and an additional overhead for each packet.

IV. SFC ARCHITECTURE

SFC is used when it is necessary to forward different service data flows through same physical network [9]. For example, e-mail delivery could require a path with Anti-Spam detection, but Instant Messaging (IM) data delivery could require IM Security threat prevention in SF path. In SDN to provide different paths the network administrator can implement traffic forwarding policy in two ways. He can create static routes of required NFs or he can create dynamic SF paths by defining a list of required services and leaving the task of their match with appropriate NFs to network controller. Figure 1. provides a schematic view of SDN architecture and SFC example policy application, where it is required for traffic to cross FW then NAT and Load Balancer (LB).

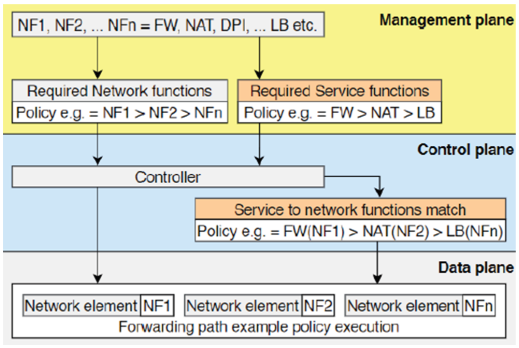

NF1, NF2, ... NFn = FW, NAT, DPI, ... LB etc.
Management plane
Required Network functions
Policy e.g. = NF1 > NF2 > NFn
Required Service functions
Policy e.g. = FW > NAT > LB
Control plane
Controller
Service to network functions match
Policy e.g. = FW(NF1) > NAT(NF2) > LB(NFn)
Data plane
Network element NF1
Network element NF2
Network element NFn
Forwarding path example policy execution

Fig. 1. SFC policy application in SDN

Dynamic path policy is useful in big computer networks for example data centers or in case where more than one similar service is placed on many SFs. Traffic forwarding without SFC is made with use of policy provided by controller for each switch independently, but with SFC traffic forwarding is made on bases of initial classification distributed to all SFC elements (see figure 3.).

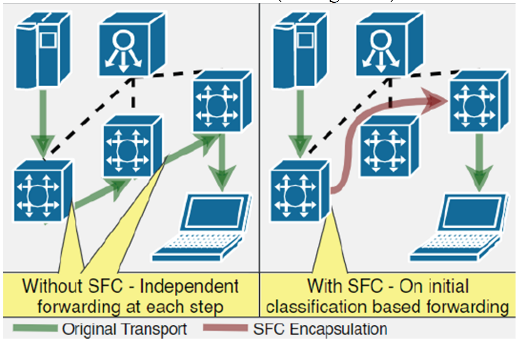

Without SFC - Independent forwarding at each step
With SFC - On initial classification based forwarding
Original Transport
SFC Encapsulation

Fig. 3. SFC Domain [1]

SFC architecture defines elements to be used to perform actions for traffic forwarding. SFC domain and its elements (see figure 3.) are described in list below:

- SF Forwarder (SFF) is a network device for traffic forwarding between SFs. It can perform SFC encapsulated packet forwarding and SFC encapsulation removal.
- SF Classifier (CL) used for traffic classification for directing it through right SF path. It can perform application and replacement of SF encapsulation.
- SF Proxy (SFP) is a SFF placed in boundaries of SFC domain to steer traffic to CL at SFC domain ingress or to remove SF encapsulation at SFC domain egress.
- SF is an NF which is a part of SFC domain.
- SFC Encapsulation is used within SFC domain for traffic forwarding and metadata exchange if NSH is used.
- SFC domain is an overlay of physical network transport topology. SFC domain cannot be bigger than single administrative domain.

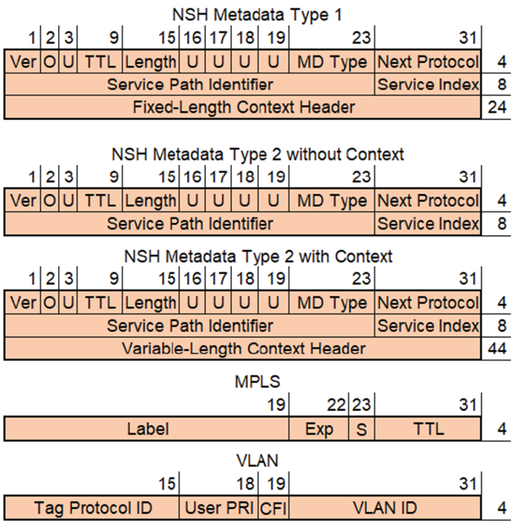

NSH Metadata Type 1
1 2 3 9 15 16 17 18 19 23 31
Ver O U TTL Length U U U U MD Type Next Protocol 4
Service Path Identifier Service Index 8
Fixed-Length Context Header 24
NSH Metadata Type 2 without Context
1 2 3 9 15 16 17 18 19 23 31
Ver O U TTL Length U U U U MD Type Next Protocol 4
Service Path Identifier Service Index 8
NSH Metadata Type 2 with Context
1 2 3 9 15 16 17 18 19 23 31
Ver O U TTL Length U U U U MD Type Next Protocol 4
Service Path Identifier Service Index 8
Variable-Length Context Header 44
MPLS
19 22 23 31
Label Exp S TTL 4
VLAN
15 18 19 31
Tag Protocol ID User PRI CFI VLAN ID 4

Fig. 4. SFC encapsulations

To compliment with SFC architecture requirements the NSH protocol is developed. It allows to follow traffic along SF path. NSH alongside other encapsulation protocols used in this paper is shown in figure 4. and contains flowing headers:

- Base Header – a 4-byte header with important fields as Next Protocol for determining of incapsulated packet most outer header protocol and Metadata type (MD-Type) for determining the metadata added to packet.
- Service Function Path header – a 4-byte header which consists Service Path Identifier (SPI) for determining which SF path packet must follow and a Service Index (SI) for determining where at SF path packet is currently located.
- Context header – A 0 to 36-byte field which consist of non, fixed, or variable length metadata depending on Metadata Type used.

V. PROPOSED ENCAPSULATION APPLICATION APPROACHES

SFC architecture impose an SFC encapsulation to be used along the SF path with an exception when SF is not aware of SFC. This approach of encapsulation application imposes restrictions of SF path length and adds an additional overhead. We propose a Network Topology-aware SFC to overcome both issues. It is a two-part solution which provides two alternative approaches:

- **Initial-partial encapsulation** is applied because of initial classification and continued partial in the SF path.
- **As Required encapsulation** is applied only if it is required to alter the original traffic path.

As Required SFC encapsulation is useful within boundaries of a single datacenter. For example, it could be applied if regular path would need to be altered for inclusion of more SFs and removed for traffic to follow its regular path.

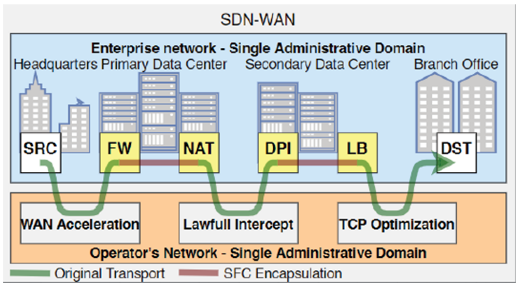


Fig. 5. Extended SFC path through SDN-WAN

Initial-partial encapsulation approach allows to extend SF path over multiple administrative domains (see Fig. 5). The SFC paths starts at Source (SRC) in Headquarters of Enterprise network, but is continued in Operator's network for WAN Acceleration, after which it could be forwarded trough more NFs out of which only those in datacenters require SFC encapsulation.

VI. Emulation Network Setup

For comparison between SFC encapsulation application approaches we created a virtual network (see figure 6.) and used Multiprotocol Label Switching (MPLS), Virtual Local Area Network (VLAN) and Network Service Header (NSH) protocols as SFC encapsulation. We used a Mininet network emulator for SFC domain creation in virtual machine (VM). We also used SCAPY packet crafting tool and its contributory library, which contains NSH protocol. Ryu controller was used for Open-source virtual switch (OVS) configuration and forwarding policy impose. Description of OVS is available by reference [11]. For SCAPY it is available by reference [12].

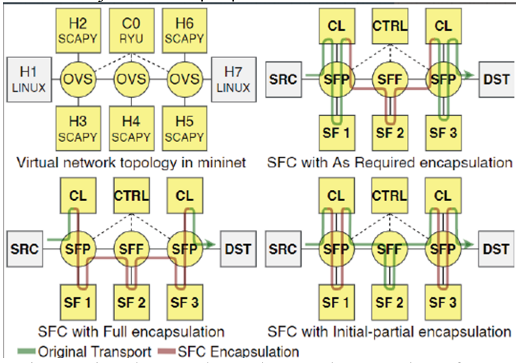


Fig. 6. Virtual network topology and approaches of SFC encapsulation application

We created 26 different scenarios of forwarding the same TCP data flows with different packet count and payload sizes through SFC domain. Half of the scenarios were made with maximal size of Protocol Data Unit (PDU) of SFC Encapsulation (see Fig. 7.), while for the other half PDU was set constant with payload of 20 bytes.

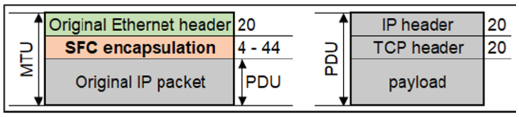


Fig. 7. PDU of SFC encapsulation

Next difference in scenarios was approach of SFC encapsulation application and used protocol itself. NSH was examined with MD-Type 1, containing fixed length (16 bytes) Context, and with MD-Type 2 containing ether none or variable length (44 bytes) Context. As SF paths were asynchronous ACK responses were not taken into a count of results. To ensure trustworthy of results some requirements had been made. Maximum Transmit Unit (MTU) was set down to 1400 bytes to ensure that fragmentation would not occur. Encapsulation which would be added in SFC domain was counted in packet creation to determine maximal payload length before packets were sent from SRC. Each link had 10Mb/s bandwidth and 10ms of delay which are much lower values then physical or even virtual machine could handle. Attributes were set to maintain independence from VM performance in the results. We added NSH SI, SPI and Next Protocol fields to Ryu REST application for flow policy setup. Description of Ryu controller is available by reference [13].

VII. Results

Our results show that with use of Initial-partial or As-Required encapsulation approaches the overhead made by SFC encapsulation is significantly minimized. In figure 8. it is depicted that overhead added to a single packet can be reduced at least by half of the original amount.

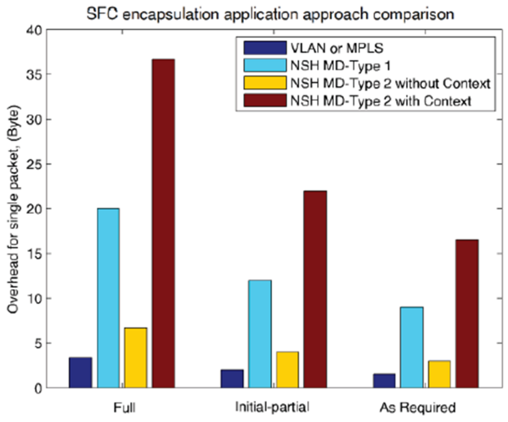


Fig. 8. Encapsulation application approach comparison

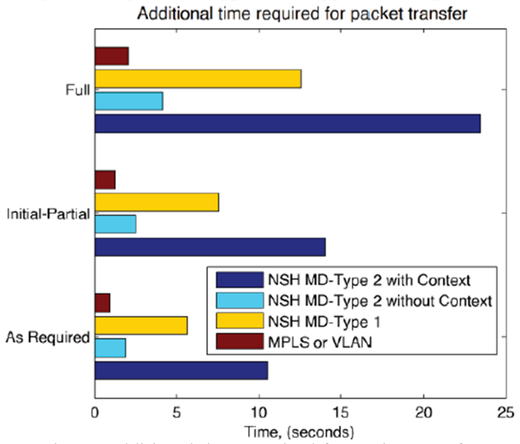


Fig. 9. Additional time required for packet transfer comparison

From our evaluations it was also evident that some of packet transmissions between network nodes were significantly longer than others. That is explainable by necessity for use of more packets within each link between network nodes due to size of additional overhead.

Figure 9. dose not show time difference between TCP flow transmissions from SRC to DST. It shows difference between additional time necessary to carry the same amount of data over a link from one network node to another summed for all SFC domain nodes for each scenario separately. As we can see that with use of As Required encapsulation overhead for NSH MD-Type 2 with Context packet transfer requires only 11 seconds while with Initial-Partial encapsulation it is 14s but with Full encapsulation it is 24s. The same pattern is continued with the rest of encapsulation protocols.

In figure 10. and figure 11. it is depicted that packet count required for delivering the same amount of traffic does not change the arrangement of encapsulation application approaches between the graphs. Best results of gained overall overhead are achieved with As Required encapsulation and Initial-Partial encapsulation application approaches. These results also show already well-known truth that significantly larger amount of overhead will be added to traffic with smaller PDU and more packets used to deliver data.

VIII. CONCLUSION

In this paper, we pointed out issues in SFC architecture brought by SFC encapsulation application approach. To resolve these issues, we introduced a Network Topology-aware SFC which allows to reduce overall overhead, use SF path between multiple administrative domains, make length of SF path an indefinite value.

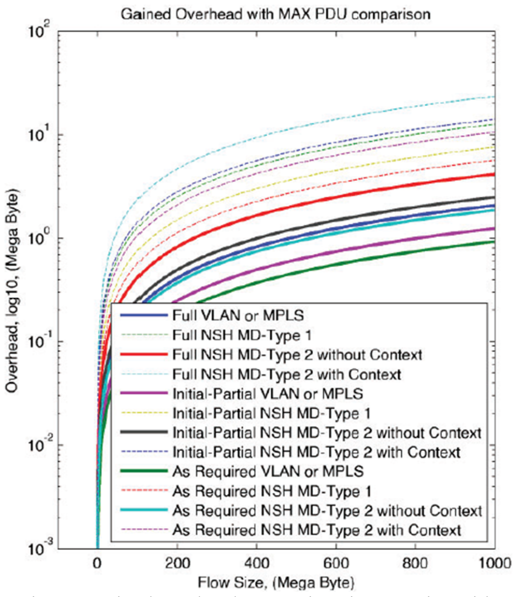


Fig. 10. Gained overhead comparison in scenarios with maximal PDU used

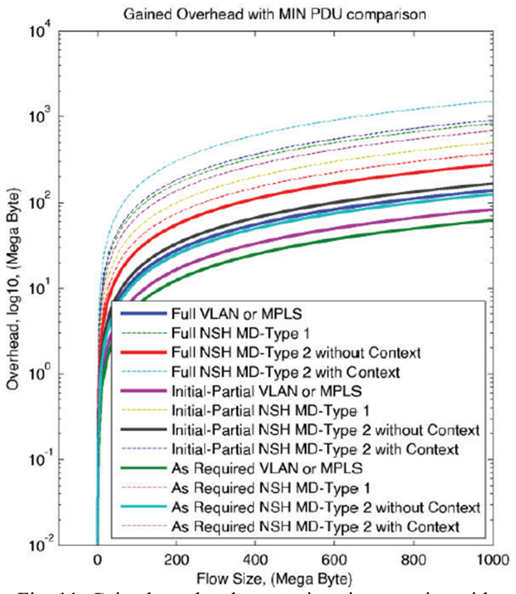


Fig. 11. Gained overhead comparison in scenarios with minimal PDU used

REFERENCES

[1] J. Halpern and C. Pignataro, "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, Available: https://www.rfc-editor.org/info/rfc7665
[2] P. Quinn and J. Guichard, "Service function chaining: Creating a service plane via network service headers," in Computer, vol. 47, no. 11, pp. 38–44, 2014
[3] P. Quinn and T. Nadeau, "Problem Statement for Service Function Chaining", RFC 7498, DOI: 10.17487/RFC7498, April 2015, Available: https://www.rfc-editor.org/info/rfc7498
[4] H. Gunleifsen and T. Kemmerich, "Security Requirements for Service Function Chaining Isolation and Encryption" in 17th IEEE International Conference on Communication Technology, pp. 1360-1365, 2017
[5] B. Dab, I. Fajjari, M. Rohon, C. Auboin, A. Diqu´elou, "An Efficient Traffic Steering for Cloud-Native Service Function Chaining", in 23rd Conference on Innovations in Clouds, Internet and Networks (ICIN), 2020
[6] H. Hantouti, N. Benamar, T. Taleb, "VLAN-based Traffic Steering for Hierarchical Service Function Chaining", in IEEE Wireless Communications and Networking Conference (WCNC), 2019
[7] H. Hantouti, N. Benamar, T. Taleb, "A Novel Compact Header for Traffic Steering in Service Function Chaining", in IEEE International Conference on Communications (ICC), 2018
[8] C. Huang and J. Zhu, "Service Forwarding Label for Network Function Virtualization and Application-centric Traffic Steering", in 12th International Joint Conference on e-Business and Telecommunications (ICETE), 2015
[9] A. Bujari, C. E. Palazzi, D. Polonio, M. Zanella, "Service Function Chaining: a lightweight container-based management and orchestration plane", in 16th IEEE Annual Consumer Communications & Networking Conference (CCNC), 2019
[10] P. Quinn, J. Halpern, C. Pignataro, "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, Available: https: https://www.rfc-editor.org/info/rfc8300
[11] OpenvSwitch, [online] Available: http://openvswitch.org/, last viewed September 2020
[12] P. Biondi, Scapy, [online] Available: https://scapy.readthedocs.io/, last viewed September 2020
[13] Nippon Telegraph and Telephone Corporation, Ryu, [Online] Available: https://ryu.readthedocs.io/en/latest/writing_ryu_app.html last viewed September 2020

Appendix 2

M. Mihaeljans and A. Skrastins
Reactive Service Function Path Discovery Approach in Software Defined Network
2021 29th Telecommunications Forum (TELFOR)
Belgrade, Serbia, 2021

29th Telecommunications forum TELFOR 2021 Serbia, Belgrade, November 23-24, 2021

Reactive Service Function Path Discovery Approach in Software Defined Network

Martins Mihaeljans
Institute of Telecommunications
Riga Technical University
Riga, Latvia
martins.mihaeljans@edu.rtu.lv

Andris Skrastins
Institute of Telecommunications
Riga Technical University
Riga, Latvia
andris.skrastins@rtu.lv

***Abstract*— In this paper we study a way to exclude the necessity for proactive Service Function (SF) path policy from Service Function Chaining (SFC) classification process. SFC architecture uses network traffic classification for steering network traffic through SF path. Architecture does establish overall requirements for classification but is unable to dynamically discover SF paths. Therefore, we propose a reactive SF path discovery method. Reactive SF path discovery gains an advantage over proactive discovery by use of reclassification for unknown network traffic. We examined our method in a Software Defined Network (SDN) emulation in Mininet emulator with RYU SDN controller. The result of this study is a comparison between proactive and reactive SF path discovery, which shows that our SF path discovery method has higher probability of successful SF path discovery.**

Keywords—Forwarding Path, Network Function, Network Service Header, Service Function Chaining, Service Function Classifier, Software Defined Network

I. Introduction

Often, to accomplish the complicated task of end-to-end service delivery successfully, network traffic must cross many Network Functions (NF). Firewall (FW) and Deep Packet Inspection (DPI) are some of well-known examples of NFs. In Software Defined Network (SDN) packet forwarding is managed via Control plane but executed in Data plane. With expansion of virtual NF capabilities network operators had leaned in favor to SDN over legacy networks. This switch led to introduction of Service Function Chaining (SFC). SFC is a mechanism of steering network traffic through ordered set of Service Functions (SF). SF are SFC-aware NFs. SF path is a way which network traffic takes through SFC domain.

In this paper we study a way to exclude requirement to know a valid SF path before the first network traffic arrives. As SFC uses SF classification for creation of SF paths, the classification result – SF paths policy is made proactively. We argue that this requirement can be overthrown with introduction of a default SF path and reactive SF path discovery which allows for SF paths to be established at the time of previously unknown network traffic arrival. Outcome of this study shows that reactive SF path discovery does guaranty higher probability of finding successful SF path and enables reclassification for unknown network traffic.

II. Related works

Authors of study [6] discuss link failure and how it can break SF path and causes a complete service outage. As a solution they introduced a flexible detouring mechanism with minimal delay for failures. Our research does acknowledge this drawback of link failure as well. SF paths must be able to adapt to dynamically to topology changes.

In study [7] authors argue that the SF paths performance may vary according to which SFs are selected at SF classification. As an exploration of this statement, we looked at SF matching ratio in figure 8. Which shows how many unsuccessful SF path discoveries are possible before finding a valid path.

Study [8] shows that in a case where different service is ordered other than the service provided by an SF, a method of detecting it and notifying related information is needed.

Study [9] demonstrates an enhanced SF allocation within SF domain with centralized database. Similarly, a centralized SF mapping does allow reclassification capability for our classification method. Thanks to it, we tracked faulty paths.

In study [10], the authors have explored SF paths to shorten the delay of the SF path's updates at failure of an SF and have propose a self-recovery scheme.

The optimization of SFC classification is studied in [11]. Authors proposed that SFC domain elements should also host classification rules. This study does acknowledge the importance of reclassification capability throughout SF path.

III. Proactive SF path discovery and problem statement

For network administrator to create an SFC domain, it is necessary to predict what kind of incoming network traffic will arrive. This manual labor requires network maintenance every time path policy update is required. This approach is shown in figure 1. defined as proactive SF path discovery.

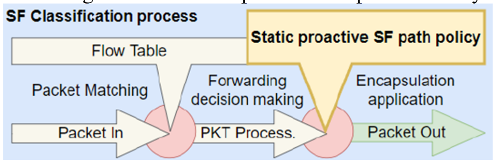

SF Classification process
Static proactive SF path policy
Flow Table
Packet Matching
Forwarding decision making
Encapsulation application
Packet In
PKT Process.
Packet Out

Fig. 1. Proactive SF path discovery

Previous studies have shown that initial classification as well as reclassification plays a huge role in SF path determination. Drawback of proactive SF path discovery is reclassification, as it occurs only for revised failure scenarios. Consequently, method is error prone due to necessity for predictions of service outcomes. Thus, updates in SF path policy due to misconfiguration can lead to a system downtime.

IV. Service Function domain boundaries and control

A. Service Function Chaining architecture

Managed via SFC Control plane element, SFC domain contains all SFC data plane components listed below:

- Service Function (SF) – SF encapsulation aware NF.
- SF Path – Path that is taken to travers SFC domain.

- SF Classifier (CL) – Encapsulates network traffic packets upon entering SFC domain.
- SF Forwarder (SFF) – Responsible for network traffic forwarding between SFC elements.
- SF Proxy (SFP) – Removes SF encapsulation from network packets upon exiting SFC domain.
- SF Encapsulation – Additional packet encapsulation for traffic steering through SF path. We used Network Service Header (NSH) protocol.

In figure 2. SF utilization is shown. Unlike other SFC elements SF classifier does not resides in SF path. SF path begins only when SF encapsulation is applied.

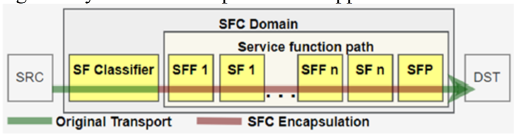


Fig. 2. Service Function Chaining Domain [1]

In our previous study [5] we explored the possibilities of altering SF encapsulation application method. We proposed two alternatives to Full SF encapsulation application method shown in figure 2. Our proposed alternatives were:

- **Initial-Partial encapsulation** – SF encapsulation that is applied at the time of initial classification and continued partially in SF path.
- **As Required encapsulation** – SF encapsulation that is applied only if SF path differs from underlying transport network topology.

Previously we provided more in-depth review of what SFC is. So, in advance, for gain of better understandings, we do advise usage of that study as an extra explanatory if a need for such material rises.

B. Service Function Chaining Control plane

Although Control plane is outside of the scope of SFC Architecture [1], IETF has published draft on SFC Control Plane components and requirements [2] which describes conveying information between SFC control elements and SFC Data plane elements. A set of interfaces for interaction between SFC-aware elements has been defined as well as additional SFC elements which are listed below:

- SFC Control Element (only in context of this paper abbreviated as CTRL) – A logical entity that instructs SFC Data plane elements on how to process packets.
- SFC Classification Rule – A rule in SF classifier's policy for binding incoming data flow packets to appropriate SF path.
- SF Path's Forwarding Policy Table – reflects traffic forwarding policy which has been enforced by SFF.

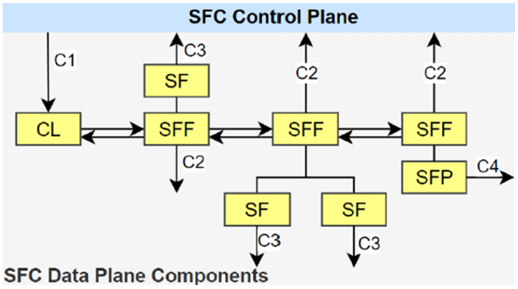


Fig. 3. SFC Control Plane Interfaces [2]

Figure 3. shows SFC Control plane interfaces that are required for conveying control information. Each interface has its own purpose:

- C1 – Interface to manage classification rules in CL.
- C2 – Interface to manage traffic forwarding in SFF.
- C3 – Interface to interact with SFs.
- C4 – Interface to give instructions for SF Proxy.

These interfaces allow SFC Control Plane element to share instructions on network traffic handling to all SFC domain elements and retreat policy rule or SF state as well as any other valuable information from them.

V. OpenFlow and Open-source virtual switch

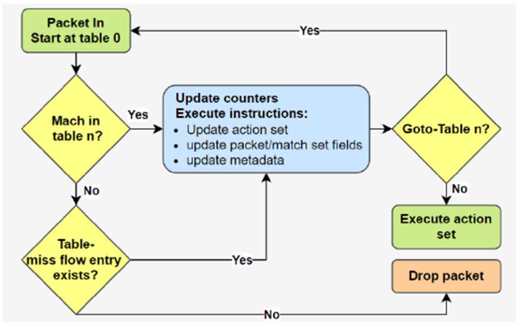


Fig. 4. Packet processing in Data plane

Packet processing in Open-source Virtual Switch (OVS) which uses OpenFlow protocol [3] is shown in figure 4. On packet arrival it is matched against entries of Table 0. If no match is found the packet is forwarded via Miss Flow Entry or dropped. Miss Flow Entry is a multipurpose traffic policy rule which can send network traffic out via any OVS interface.

Without SFC the only way to ensure that a known and unknown data flow would reach appropriate SF, is to steer network traffic through all SFs.

We used Miss Flow Entry as an alternative to both SF path discovery methods. As steering was accomplished with use of static Miss Flow Entry at flow tables of all OVS it did not require an additional encapsulation and network traffic traversed all SFs.

VI. Proposed Reactive SF path discovery method

Reactive SF path discovery method, in which controller does setup forwarding policy automatically from information gathered of SFs, leverages reclassification for self-correction and does not require manual correction of misconfiguration.

Reactive SF path discovery method working principle is show in figure 5. and in network topology in figure 6. It resembles a conjunction of L2 switch and OVS learning phases as method performs similar processing for an unknown data flow.

On Packet-In event L2 switch would broadcast unknown packet, but in our method, it is forwarded to a single default SF path which provides initial classification only for unknown network traffic.

Next, if a required SF does not reside within the default SF path an event of reclassification occurs. At this point, similarly to OVS, reactive SF path discovery method asks for forwarding policy to be updated by SFC Control element. Update request contains the unknown network traffic packet.

Now controller can make decision on creation of a new policy rule according to this packet.

After classification policy rules have been updated, network data flow takes a different from default SF path. If this new SF path turns out to be faulty as well, a repeated reclassification event is in order. This cycle of reclassification does occur up until a successful subsequent SF path discovery is obtained or until time to leave (TTL) value of NSH encapsulation is reached, or the cycle is disrupted by specifics of network data flow itself, for example, session timeout.

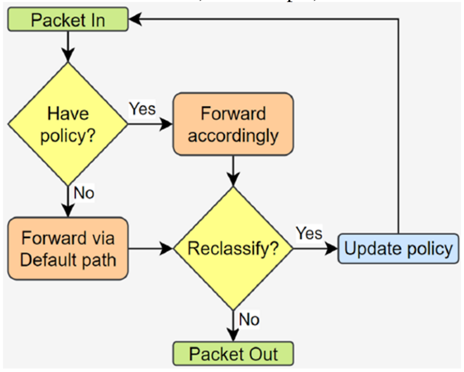


Fig. 5. Reactive packet forwarding process

VII. Software Defined Network emulation

We emulated virtual SDN environment in Mininet. RYU SDN controller [12] served for control of Data plane and as a framework for SFC Control plane element (CTRL). For packet processing in SFs, we used SCAPY packet crafting tool [13] and its contributory library.

We did 5 SDN network emulations. Each emulation differed from another with count of SFs. In all emulations 3 OVS switches were used. Each switch reassembled a single SF element. All SFs were connected to SF forwarder. SFC-aware elements in network topology are show in figure 6. Underlaying network topology behind SFC elements in Mininet is described below:

- CTRL is remote controller connected via localhost.
- CL, SFF, SFP are OVSs.
- SRC and DST are Mininet hosts.
- SF1 to SF(n) are Mininet hosts running SCAPY. We used up to 5 SFs during emulations.

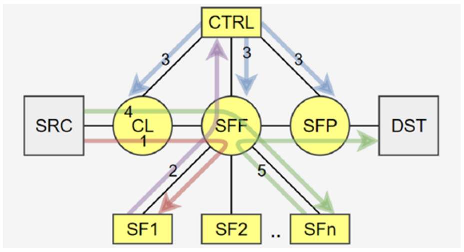


Fig. 6. Successful subsequent SF path detection

SFC was enforced asymmetrically – network data flow from SRC to DST went through SF path but returning data flow traversed network without SFC. SFs were TCP proxies. Their purpose was to check 5-tuple and let data flow through or ask for policy update to the CTRL if needed. We declared that proactive SF path discovery could not do reclassification because it is impossible to define a static policy for unknown data flows. Throughout all emulations desired SF path was a single SF.

Communication process between SFs and SFC Control plane element is shown in figure 7. The communication from SF to CTRL can be considered as utilization of interface C3, while from CTRL to OVS (CL, SFF and SFP) can be considered as utilization of interfaces C1, C2 and C4. CTRL knows default SFP. CTRL memorizes reclassification, so that it would not decide to take similar action again. After policy update, controller sends the packet received from SF back to the CL for it to gain a new NSH and enter SFC domain again.

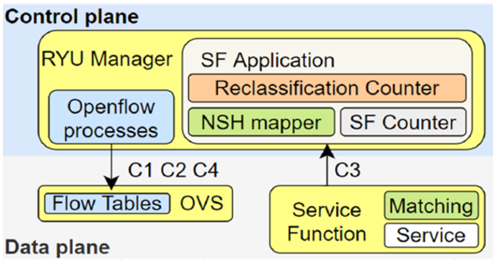


Fig. 7. Service Function Path acknowledgment process

VIII. Results

Results are not dependent on performance of physical computing capabilities of server on which emulation process took place. In fact, we do not seek to clarify performance aspects of this approach as much as we want to outline the possible benefit of subsequential SF detection. Subsequential SFP detection allows unknown data flow to be reclassified even if classification policy is single conditional. As we nullified a double condition in classification and SF matching, proactive SF path discovery could not do a reclassification.

In figure 8. Miss Flow Entry and reactive SF path discovery tend to gradually increase their score while proactive SF path discovery remains constant regardless of how many SFs are in SF domain. All three methods do start at probability value of 0.5 in figure 8. That is because with only one SF in SF domain there are only two possible outcomes of finding SF path – either it will be successful, or it will not. Matching ratio is equivalent of how many possibilities of reclassification can occur before right service is selected.

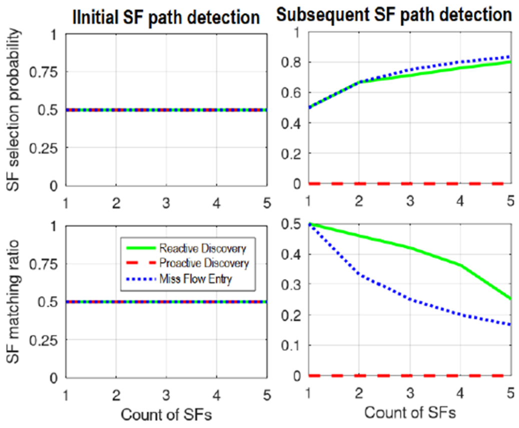


Fig. 8. Probability and matching ratio comparison

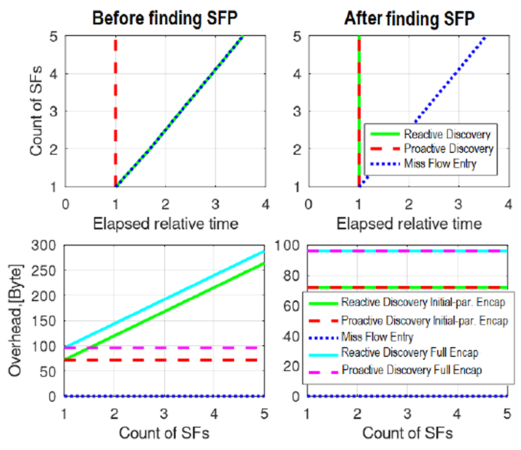


Fig. 9. Elapsed time and overhead comparison

Subsequent SF path discovery elapsed time in relative values is shown in figure 9. Elapsed time is dependent on network performance, so in our emulation network setup value of 1 is approximately 11ms. Time required for service to be selected depends on how many reclassifications are possible before a successful SF utilization occurs. Once it does occur, path to reach SF does not differ between reactive and proactive SF path discovery methods. So, time does not differ either.

Overall single packets overhead gained by additional packet encapsulation with NSH protocol accumulated from each network link is shown in figure 9. Difference of Initial-Partial and Full encapsulation is dependent on emulation topology. As in our topology all SFs were connected to a single SSF, only 1 link could escape encapsulation with use of Initial-partial Encapsulation. It was connection from SFF to SFP.

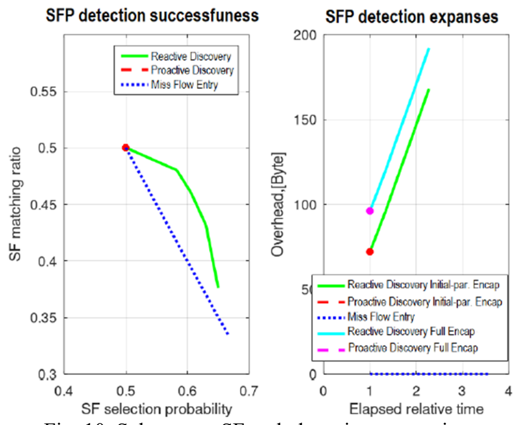


Fig. 10. Subsequent SF path detection comparison

Proactive SF path discovery is static as it does not require additional time for SFP detection and does not accumulate additional overhead due to reclassification as shown in figure 10. Reactive SF path discovery on the other hand does require additional time for successfully finding SFP as well as it generates additional overhead.

IX. Conclussion

In this paper, we discussed drawbacks of establishing SF paths policy proactively and proposed reactive SF path discovery method to overcome them. Evaluation results have proven that reactive SF path discovery significantly increases probability of finding successful SF utilization.

Figure 8. shows that it is important to distinguish initial SF path detection form subsequent SF path detection in order to acknowledge that proactive SF path discovery is going to drop data flows while reactive SF path discovery is going to do a subsequent SF paths detection.

Figure 9. and 10. shows that subsequent SF paths detection does require additional resources only for SF paths discovery process. This is a huge improvement compering to use of Miss Flow Entry which by default will continue to steer all data flows through all SFs, while reactive SF path discovery will require only for the first packet to cross multiple SFs. As result, SFP discovery process causes minimal performance impact as most of the data flows is intact.

We focused this study on task of finding a balance between use of Miss Flow Entry and proactive SF path discovery methods. The outcome is reactive SF path discovery method in which previously manual labor of SF path discovery is now delegated to the controller. Controller does decision making on SF paths discovery reactively therefore it allows for previously unknown data flows to be classified.

References

[1] J. Halpern and C. Pignataro, “Service Function Chaining (SFC) Architecture”, RFC 7665, DOI 10.17487/RFC7665, October 2015
[2] Boucadair, M., "Service Function Chaining (SFC) Control Plane Components & Requirements", draft-ietf-sfc-control-plane-08 (Informational), October 2016
[3] Open Networking Foundation, “OpenFlow Switch Specification Version 1.5.1,” 2015
[4] P. Quinn, U. Elzur and C. Pignataro, “Network Service Header (NSH),” IETF RFC 8300, 2018
[5] M. Mihaeljans and A. Skrastins, “Network Topology-aware Service Function Chaining in Software Defined Network”, in 28th Telecommunications forum TELFOR, 2020
[6] H. Jeong, S. M. Raza, D. T. Nguyen, S. Kim, M. Kim and H. Choo, “Control Plane Design for Failure Protection in Software Defined Service Function Chains”, in 14th International Conference on Ubiquitous Information Management and Communication, 2020
[7] S. Lee, S. Pack, M. Shin, “Service Function Path Adaptation in SFC”, 19th International Conference on Advanced Communication Technology, 2017
[8] S. Choi, M. Choi, M. Shin, S. Lee, “Yang Data Model for SFC Control Plane”, in 18th Asia-Pacific Network Operations and Management Symposium, 2016
[9] R. Kang, F. He, T. Sato, E. Oki, “Demonstration of Network Service Header Based Service Function Chain Application with Function Allocation Model”, in IEEE/IFIP Network Operations and Management Symposium, 2020
[10] S. Lee and M. Shin, “A Self-recovery Scheme for Service Function Chaining”, in International Conference on Information and Communication Technology Convergence, 2015
[11] M. Polverini, J. Gal´an-Jim´enezy, F. G. Lavaccaz, A. Cianfrani, V. Eramo, “Dynamic In-Network Classification for Service Function Chaining ready SDN networks”, 10th International Conference on the Network of the Future, 2019
[12] Nippon Telegraph and Telephone Corporation, Ryu, [Online] Available: https://ryu.readthedocs.io/en/latest/writing_ryu_app.html, last viewed May 2021
[13] P. Biondi, Scapy, [online] Available: https://scapy.readthedocs.io/, last viewed May 2021

Appendix 3

M. Mihaeljans and A. Skrastins
Evaluation of reactive service function path discovery in symmetrical environment
Telfor Journal, vol. 14, no. 1, pp. 2-7, 2022

Telfor Journal, Vol. 14, No. 1, 2022.

Original scientific paper

Evaluation of Reactive Service Function Path Discovery in Symmetrical Environment

Martins Mihaeljans, *Student Member, IEEE*, and Andris Skrastins, *Member*, *IEEE*

***Abstract* — In this paper we continue our study of path discovery process for Service Function Chaining (SFC) in Software Defined Network (SDN). By default, service function (SF) paths are established proactively - before data transmission takes place. We have argued that this constraint can be eliminated with the use of our proposed Reactive SF path discovery approach. Such SFs as network address translation (NAT) or stateful firewall (FW) are SF path's symmetry dependent requiring a visit of both ingress and egress flows. Thus, we evaluated SF path discovery processes in Mininet emulation network. Outcome of this study is a comparison of proactive and reactive SF path discovery processes for both asymmetrical and symmetrical SF paths. It shows that even in symmetrical environment reactive SF path discovery has a higher probability of successful SF path detection.**

***Keywords* — Network Service Header, Network function, Packet encapsulation, Symmetrical connection, Software Defined Network, Service Function Chaining.**

I. Introduction

END-TO-END service delivery is a complicated task which can stress the forwarding plane and lead to unexpected errors, due requirement for data flow to cross multiple network functions (NF). Service Function Chaining (SFC) architecture [1] defines NFs as service functions (SF)s. Common examples of SFs are firewall (FW) and load balancer (LB). SFC is a method of data flow steering through ordered SF path derived from possibilities of Software Defined Networking (SDN) and Network

Function Virtualization (NFV). SDN allows network operators to implement complicated network topologies by use of virtual NFs. This ability is a key benefit for operators to prefer SDN over legacy networks.

This paper is a continuation of our study of Reactive Service Function Path Discovery Approach in Software Defined Network [2]. We have argued that preset of a complete SF path policy before initial data flow arrival is an unnecessary requirement. This study focuses on previously untangled question of SF path symmetry's effect on path discovery process.

Outcome of this study is a comparison of proactive and reactive SF path discovery process for asymmetrical and symmetrical SF paths, indicating that even in symmetrical environment reactive SF path discovery has a higher probability of valid SF path discovery.

Paper received May 22, 2022; revised June 27, 2022; accepted June 28, 2022. Date of publication August 05, 2022. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Grozdan Petrović.

This paper is revised and expanded version of the paper presented at the 29th Telecommunications Forum TELFOR 2021 [2].

Martins Mihaeljans is with the Riga Technical University, Azenes str. 12 - 325, LV1010 Riga, Latvia,
(e-mail: martins.mihaeljans@edu.rtu.lv).

Andris Skrastins is with the Riga Technical University, Azenes str. 12- 325, LV1010 Riga, Latvia (e-mail: andris.skrastins@rtu.lv).

II. Related Works

H. Hantouti and N. Benamar [3] have argued that SF path symmetry has a great effect on network service delivery. In

their study a reverse path calculation algorithm is introduced, yet SF path discovery is not considered. The study also suggests that having fully symmetrical SF paths is redundant and negatively affects the performance of Service Function Forwarders (SFF). Authors propose the use of partially symmetrical SF paths as a solution.

Authors of study paper [4] state that SF placement in virtual environment allows flexible SFC deployments. In such deployments resource cost between different SFs is unequal.

For proper resource cost computation, authors have proposed an algorithm that is based on the use of predefined candidate path sets. Although, service applicability and probability of valid service discovery has not been considered, research does outline the importance of distinguishing valuable SF paths.

Study [5] finds partial SF path symmetry to be an adequate choice for enabling network slicing technique in 5G networks. Unlike our emulation environment, SF symmetry requirement is detected by a controller and not the SFs themselves. Such configuration might require an extensive integration of each element into the proposed framework. Which is an obstruct that might lead to inability to use plug-and-play systems.

It's pointed out that symmetrical SF paths are mandatory for some SFs. Given examples are stateful FW and Intrusion Detection System (IDS). Such SFs require data flow to traverse them from both ingress and egress directions. An opposite are SFs concerned only with the incoming data flow such as spam checks and URL filters. Use of partially symmetrical SF path would result in latency reduction, greater policy flexibility and lowered deployment cost.

Study [6] describes the necessity of dynamic topology changes in SFC domain as a link failure can cause a complete service outage. Similarly, study [7] introduces a self-recovery scheme for SF failure scenarios. Authors of study discuss an objective of SFs hosting classification rules and participating in management of path policy.

SF participation in path policy update process has been covered in our previous research study [2] as well. Unlike study [7] we do not encourage hosting classification rules at SFs as it might not only prolong policy update process but also lead to policy synchronization issues and requirement for SF hierarchy to eliminate concurring extensive path policy update requests.

III. Service Functions and Problem Statement

A. Open-source Virtual Switch and OpenFlow

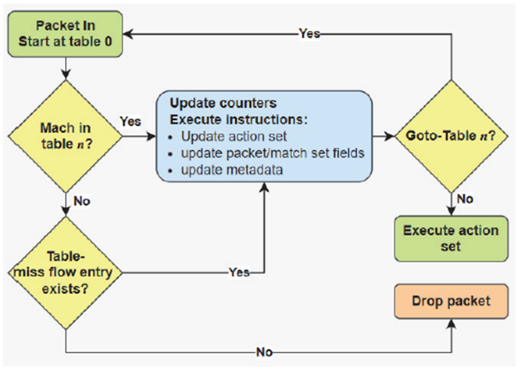


Fig. 1. Open-source virtual switch [8].

An open-source virtual switch (OVS) [8] enables packet processing in SFC domain. OVS working principle is shown in Fig. 1. On data flow arrival, OVS primarily determines if it has a match in existing path policy.

Miss-Flow entry is a multipurpose rule in path policy that is used only if no match against other rules is found. In flow table without Miss-Flow entry unmatched flows are dropped. In our emulations we used Miss-Flow entry to forward data flow through all SFs. Imitating an alternative method to discovery processes. Use of Miss-Flow entry does not require an additional encapsulation to be applied on data flows.

B. Service Function Chaining

SFC domain architecture shown in Fig. 2 implies no restrictions regarding path symmetry or forwarding direction. For example, SF1 can be crossed many times before data flow is forwarded to the destination.

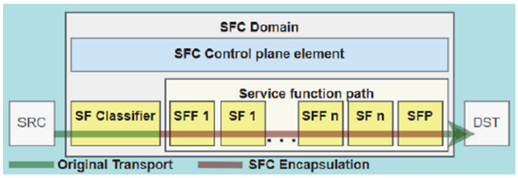


Fig. 2. Service Function Chaining domain.

SFC domain consists of classifier (CL) for adding SFC encapsulation on an incoming data flow and selecting a proper SF path. SF forwarder (SFF) for forwarding a data flow from one SFC element to another. Service function (SF). SF proxy (SFP) for SFC encapsulation removal when a data flow is destined to leave SFC domain. We have greatly explained SFC domain elements in our previous work [9] where we introduced two different SFC encapsulation application methods:

- **Initial-Partial encapsulation** – SF encapsulation that is applied at the time of initial classification and continued partially in SF path.
- **As Required encapsulation** – SF encapsulation that is applied only if SF path differs from underlying transport network topology.

C. Complicate Service Functions

Complicate SFs require for data flow to traverse them for both directions. A stateful FW is an example when an ACK won't be accepted if ACK SYN packet from a reverse direction wouldn't have been received prior. IDS is a SF that relies on the capability of comparison between received data flow and signature database for the detection of malicious activities.

An opposite are SFs requiring only for the incoming data flow to traverse them. Such SF is an Anti-spam check. It verifies the incoming e-mail and is not interested in two-way dialog between servers.

Three different SF path types shown in Fig. 3 are:

1. **Asymmetrical SF path** – Data flow crosses SFC domain only in one direction.
2. **Partially symmetrical SF path** – Data flow crosses SFC domain in both directions, but path taken in each direction can differ.
3. **Symmetrical SF path** – Data flow crosses SFC domain in both direction via the same path and returning data flow visits SFs in reverse order.

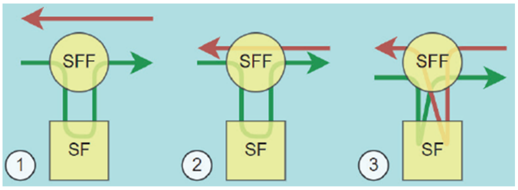


Fig. 3. SF path types.

Asymmetrical environment – emulation network setup where SFC domain ingress data flow is steered trough SF path but returning data flow traverses topology without entering SFC domain. A compliant type is **Asymmetrical SF path**.

Symmetrical environment – emulation network setup where SFC domain is traversed for both ingress and egress directions. Path types are **Partially symmetrical SF path** and (Fully) **Symmetrical SF path**.

D. Control plane for Service Function Chaining

Control interfaces shown in Fig. 4 and Fig. 7 enable complete SF path synchronization among all SFC elements. With the use of control interfaces SFC data plane components share information related to SF activity and availability with the controller and between each other. While Fig. 4 depicts the general purpose of each interface, Fig. 7 shows usage in our emulation network topology.

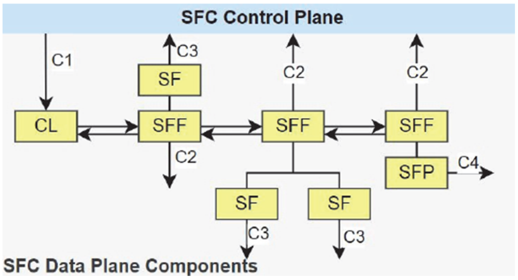


Fig. 4. SFC Control interfaces [10].

SFC Control plane components and requirements draft [10] define SFC classification rule as an entry in SF classifier's policy for binding an incoming data flow to an appropriate path. These entries fill up SF path's policy table that reflects data flow forwarding policy enforced by SFF. Interfaces from Fig. 4 can be described as follows:

- C1 – CL classification rule management
- C2 – SFF data flow forwarding management
- C3 – SF interaction interface
- C4 –SF Proxy instruction interface

E. Problem statement

A proactive SF path discovery method prohibits the use of dynamic SF paths. This constraint derives from the objective where in a default case (described by SFC architecture) SF encapsulation is added to an arriving data flow only if a previously defined classification policy exists.

In Fig. 5 a proactive SF path discovery method is shown. At data flow arrival incoming packets are matched against OVS flow table. When a match is found a forwarding decision is made and processing continues at SF path's policy table where an encapsulation is added.

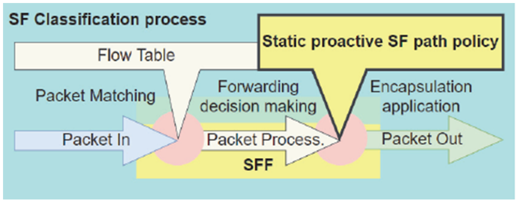

SF Classification process
Static proactive SF path policy
Flow Table
Packet Matching
Forwarding decision making
Encapsulation application
Packet In
Packet Process.
Packet Out
SFF

Fig. 5. Proactive SF path discovery.

A manually created proactive SF path policy is unable to make a data flow reclassification. Thus, dynamic path discovery cannot be achieved. Such a constraint does negatively affect a possibility to detect a valid SF path for an incoming data flow. Returning data flow can only find a valid SF path if required policy has been previously setup.

IV. Reactive Service Function Path Discovery

In our previous study Reactive Service Function Path Discovery Approach in Software Defined Network [2] we proposed **Reactive SF path discovery method** (shown in Fig. 6) as an alternative to proactive discovery. Our method gains an additional subtask for reverse path mapping for the use of (Fully) symmetrical SF paths. Partially symmetrical SF paths are discovered via a reactive SF path discovery process for returning data flow.

As shown in Fig. 7, reactive SF path discovery relies on SF's ability to inform the controller about success or failure in service application by use of control interface C3. The information also consists of received packet which SF had not been able to process. A controller does the packet process. Afterwards, it updates SFF path policy tables through interface C2 and SF classifier through interface C1. A controller also updates SF proxy for proper SF encapsulation removal at the egress through a control interface C4. SF path generation automation is achieved via the use of counting available SFs and reclassifications made for a received data flow, and mapping NSH headers.

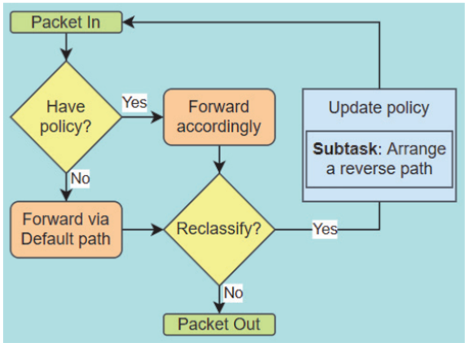


Fig. 6. Reactive SF Path discovery.

Work principle consists of four steps (see Fig. 8):

1. Initial SF path detection – All previously unknown data flows are steered through a default SF path.
2. Updating SF path policy – If SF indicates that service has not been applied and SF path is faulty then SF paths policy update is enforced among all SFC domain elements.
3. Reclassifying data flow – SF path policy update results in data flow reclassification that enables subsequential SF detection.
4. Subsequential SF path detection (an equal process to initial SF path detection only exploited on previous detection failure) – steering traffic via a newly created SF path.

Steps 2 to 4 are optional if an arriving data flow has had a previously created SF path policy at the time of arrival, otherwise they are essential in the detection of SF path with a valid SF application.

This discovery process can be repeated as many times as required for a valid SF path to be found. Or it can be disrupted in cases where time to leave (TTL) value of Network Service Header (NSH) [11] encapsulation is reached, or cycle disruption occurs due to unpredictable matters connected to communication specifics like session timeout, low delay tolerance and other.

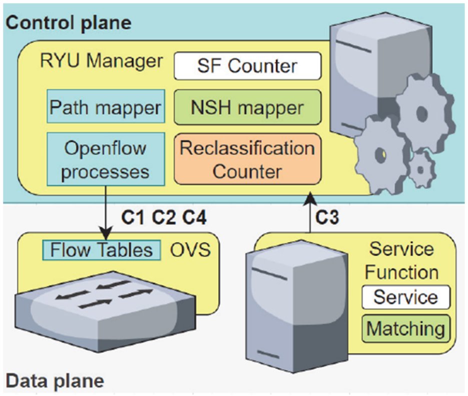


Fig. 7. Control interface usage.

V. Network Emulation Setup

SFC domain topology, emulated in Mininet network emulator is shown in Fig. 7. It consists of:

- 3 OVS switches serving as SFFs.
- 2 host serving as a data flow source and destination.
- 1 to 5 hosts (depending on emulation requirements) serving as SFs.
- Ryu controller [12] serving as a SFC control element.

SFFs connected to a data flow source and destination for ingress traffic and vice versa for egress traffic served also as SF classifiers and SF proxies. All SFs were connected to SFF at the middle of the topology (OVS with only SFF function in Fig. 8). We did emulations with up to 5 SFs in topology. Each emulation differed from one another with the count of SFs. SF path's length stayed 1 SF long throughout all emulations. Packet processing in SFs was made by the use of SCAPY packet crafting tool [13] and its contributory library that contains a NSH protocol.

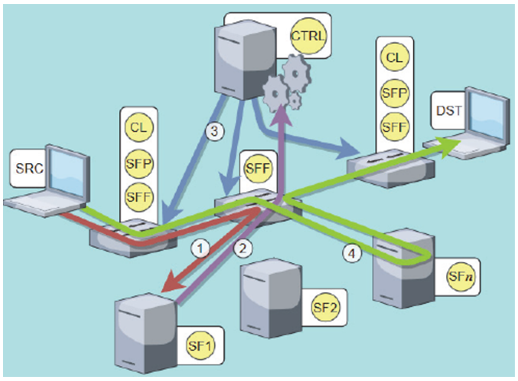


Fig. 8. Network emulation topology.

For a data flow to be steered through a desired path other than the one taken by the direction of underlying transport topology an additional packet encapsulation is required. We used Network Service Header (NSH) protocol as SF encapsulation in our emulation setup as it is complying with requirements implied by both SFC architecture and SFC control plane components and requirements draft.

Ryu SDN controller (CTRL in Fig. 8) enabled dynamic SF path creation with the use of gathering network related information from all SFC elements. This capability is the key benefit of network automation that relaxes the necessity of manual configuration from a network administrator.

However, in reactive SF path discovery a single proactive SF path rule entry is required. It is the default SF path that all previously unknown data flows will take at the initial SF path detection. It is configured manually.

VI. Results

SF path symmetry is a fundamental SF path attribute. It affects all SF path parameters. The direct effect of SF path symmetry on a valid SF path discovery is revealed by evaluation of such parameters as the probability of valid SF path discovery and SF matching ratio.

As SF path's symmetry is not a performance dependent subject (in the context of this study) we solely focused on distinguishing the impact of symmetry constraint on SF path detection. We categorized this section according to measured values which are:

- **Probability of valid SF path discovery** – a likeliness of discovering a valid SF path out of all possible SF paths available.
- **SF matching ratio** – indication of the unequalness between all possible valid SF path detections and required reclassification count to distinguish them.
- **Elapsed relative time** – a time interval required for SF path discovery process.
- **SFC generated overhead** – a cumulative value of overhead gained by the use of SFC encapsulation.
- **SF path discovery successfulness** – a scale between the probability of valid SF path discovery and SF matching ratio.
- **SF path discovery expenses** – a scale between elapsed relative time and SFC generated overhead.

A. Probability of valid SF path discovery

The probability of valid SF path discovery is shown in Fig. 9. Initial SF path detection is equal for all discovery methods as reclassification cannot occur on a first occasion (step 1 in Fig. 8) in a single conditioned path policy. In asymmetrical environment all SF path discovery methods at initial SF detection have a probability of 0.5 as a single decision can only lead to either a valid or a faulty SF path.

In symmetrical environment all discovery methods at initial SF path detection gain a lower value of 0.33 as a path for returning data flow must be considered as well.

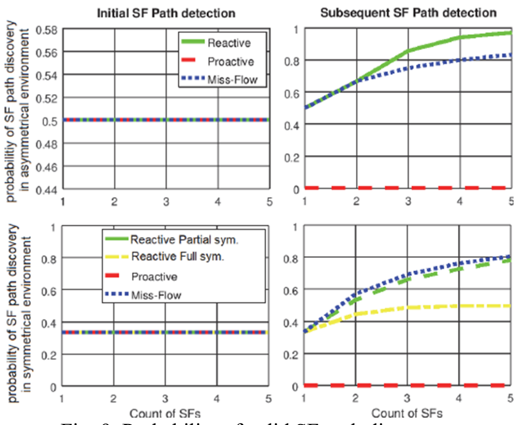


Fig. 9. Probability of valid SF path discovery.

In asymmetrical environment at subsequent SF path detection both Miss-Flow and reactive discovery shows a gradual increase of their capability of finding a successful SF path, but proactive discovery falls to a value of 0 as it is incapable of data flow reclassification.

In symmetrical environment at subsequential SF path detection a similar pattern to asymmetrical environment is encountered for all discovery processes. Reactive discovery with full symmetry climbs half the growth in value of partial symmetry, because returning path cannot do reclassification due constraint that it must be equal with a forwarding path taken by an incoming data flow.

B. SF matching ratio

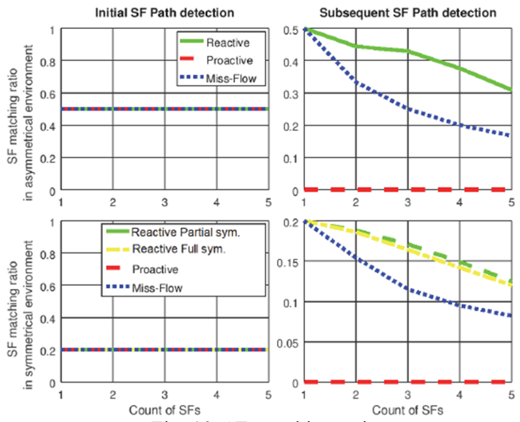


Fig. 10. SF matching ratio.

For matching ratio (Fig. 10) in both asymmetrical and symmetrical environments at initial detection all discovery processes held equal values (0.5 in asymmetrical env. and 0.2 in symmetrical env.). At subsequent SF path detection reactive discovery processes gain a higher ratio as they require for additional reclassification in comparison to Miss-Flow. Proactive SF path discovery falls to a value equal to 0 as it cannot do reclassification.

C. Elapsed relative time

Fig. 11 shows the time required for service application both spent while searching for an SF path and time required when the SF path is found.

While static methods as proactive discovery and Miss-Flow do not change at either of time objectives, reactive discovery processes do gain an advantage over Miss-Flow entry after a valid SF path is detected both in asymmetrical and symmetrical environments.

Relative time values are in the range from 10ms to 100ms. The use of relative time values enables the calculation of methods effectiveness in other setups.

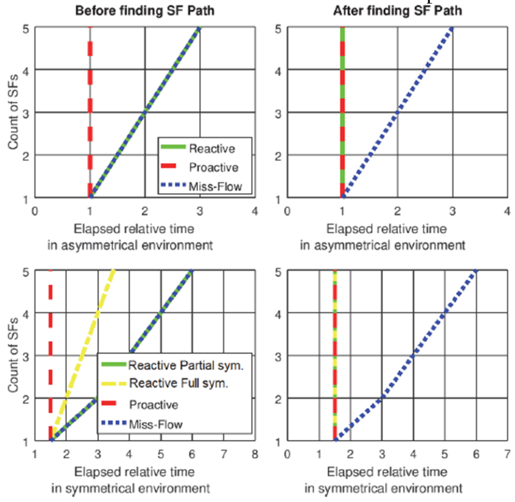


Fig. 11. Elapsed relative time.

D. SFC generated overhead

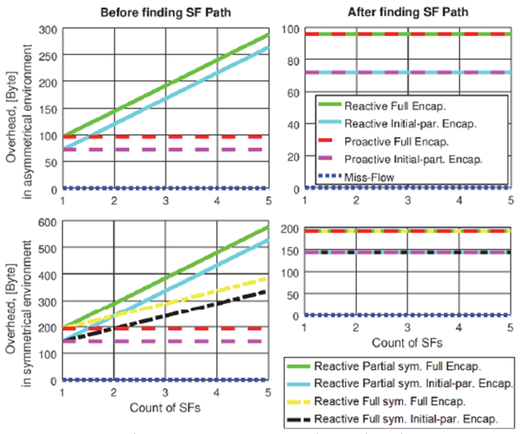


Fig. 12. SFC generated overhead.

SFC generated overhead in bytes is shown in Fig. 12. The division between objectives of searching and having a valid SF path is complimentary to the one describing time domain in Fig. 11.

While static methods as proactive SF path discovery and Miss-Flow entry does not change between objectives, the reactive SF path discovery in all cases does limit its overhead once a valid SF path is found.

Difference between Full SFC encapsulation application method and our (in earlier study [10] proposed) Initial-partial SFC encapsulation method is overhead truncation only by 20 bytes constantly throughout emulations. An explanation to such consistency hides in the simplicity of our network emulation topology. In all emulations we were able to remove encapsulation from only a single link.

As the use of Miss-Flow entry requires no additional packet encapsulation it stays at the value of 0 bytes. It might seem advantageous, but the drawback hides in Fig. 11. Where others are either still or improve their score, Miss-Flow entry has the longest time spent on discovery.

E. SF path discovery successfulness

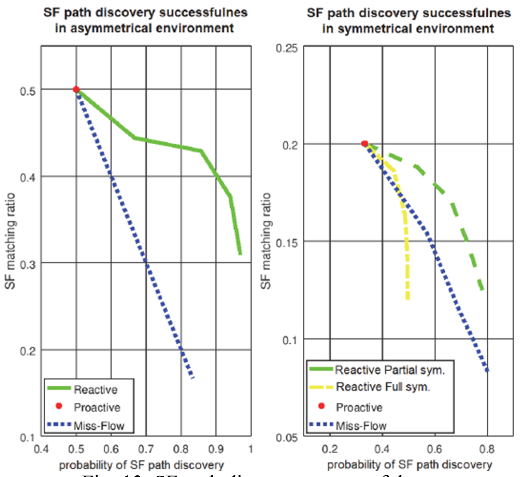


Fig. 13. SF path discovery successfulness.

Comparison in Fig. 13 places SF matching ratio and the probability of valid SF discovery at scales. As probability is closer to the value of 1 as more likely for a successful SF path detection to occur. It is the other way around for SF matching ratio as it is closer to the value of 0 as less reclassifications are likely to occur before a successful SF path detection is made.

Comparison indicates that in asymmetrical as well as symmetrical environments it is more likely to find a valid SF path either by doing a data flow reclassification (if SF path turns out to be faulty and trying a subsequential SF path detection) or by steering a data flow through all SFs in the topology no matter if the service that these SFs provide gets applied or not. The least desirable option is proactive discovery as it stays static for both comparison scales (probability and matching ratio) regardless of the environment.

F. SF path discovery expenses

At comparison in Fig. 14 overhead and elapsed relative time are at scales. Expenses mentioned do relate only to the discovery process itself (while a valid path has not been found). Spending less time in path detection as well as adding the least overhead are considered as the best outcome for the discovery processes.

In asymmetrical environment staying at a value below 100 bytes for overhead and a value of 1 for elapsed relative time, also in symmetrical environment staying at a value below 200 bytes for overhead and a value of 1.6 for elapsed relative time proactive SF path discovery gains an upper hand in both scales thanks to its static nature.

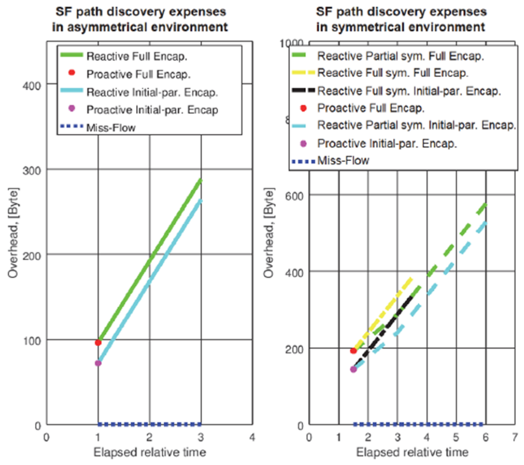


Fig. 14. SF path discovery expenses.

Miss-Flow entry is the second desirable method as during discovery process it gains no overhead. Thus, its only expenses are in time domain where in asymmetrical environment its elapsed relative time grows up to a value of 3 while in symmetrical environment it doubles this value.

Reactive SF path discovery does gain the most overhead (up to 600 bytes) while a valid SF path has not been detected. Afterwards, however, once the valid SF path is found the reactive SF path discovery has equal expenses to those of proactive SF path discovery.

VII. Conclusion

This paper is a continuation of our research of Reactive SF path discovery Approach in Software Defined Network [2] proposing reactive SF path discovery as an alternative to proactive SF path discovery. While proactive SF path discovery relies on the use of a manually configured predefined SF path policy, reactive SF path discovery leverages data flow reclassification for SF path generation reactively at the arrival of a previously unknown data flow.

This study focuses on SF path symmetry's effect on path discovery as it is a fundamental attribute of each path. In addition to proactive and reactive SF path discovery methods we also emulated the use of Miss-Flow entry. Reactive SF path discovery is the only method of those that we emulated which can leverage network automation via a SDN controller.

Results indicate that proactive SF path discovery has a static nature in asymmetrical as well as symmetrical environments, thus it does not support a subsequential SF path detection in single conditioned path policy.

Usage of partial symmetry is advisable in cases where a high probability of valid SF path discovery is required and additional expenses for discovery process can be tolerated.

Usage of full symmetry is advisable in cases where SFs require visibility of both ingress and egress traffic, while a lower probability of valid SF path detection is available.

In overall, reactive SF path discovery gains a higher probability of valid SF path discovery than proactive discovery in asymmetrical and symmetrical environments.

References

[1] J. Halpern and C. Pignataro, "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015.

[2] M. Mihaeljans and A. Skrastins, "Reactive Service Function Path Discovery Approach in Software Defined Network," *2021 29th Telecommunications Forum (TELFOR)*, 2021, pp. 1-4.

[3] H. Hantouti and N. Benamar, "Partially Symmetric Service Function Chains," *2019 2nd IEEE Middle East and North Africa COMMunications Conference (MENACOMM)*, 2019, pp. 1-6.

[4] Y. Zhang and X. Cao, H. Yu and G. Sun, "Service Function Chain Deployment Based on Candidate Paths", *6th International Conference on UK-China Emerging Technologies*, 2021.

[5] H. Hantouti, N. Benamar, M. Bagaa and T. Taleb, "Symmetry-Aware SFC Framework for 5G Networks," *IEEE Network*, vol. 35, no. 5, pp. 234-241, September/October 2021.

[6] M. Polverini, J. Gal´an-Jim´enezy, F. G. Lavaccaz, A. Cianfrani, V. Eramo, "Dynamic In-Network Classification for Service Function Chaining ready SDN networks", *10th International Conference on the Network of the Future*, 2019.

[7] S. Lee and M. Shin, "A self-recovery scheme for service function chaining," *2015 International Conference on Information and Communication Technology Convergence (ICTC)*, 2015, pp. 108-112.

[8] Open Networking Foundation, "OpenFlow Switch Specification Version 1.5.1," 2015.

[9] M. Mihaeljans and A. Skrastins, "Network Topology-aware Service Function Chaining in Software Defined Network," *2020 28th Telecommunications Forum (TELFOR)*, 2020, pp. 1-4.

[10] M. Boucadair, "Service Function Chaining (SFC) Control Plane Components & Requirements," draft-ietf-sfc-control-plane-08 (Informational), October 2016.

[11] P. Quinn, U. Elzur and C. Pignataro, "Network Service Header (NSH)," IETF RFC 8300, 2018.

[12] Nippon Telegraph and Telephone Corporation, Ryu, [Online] Available:https://ryu.readthedocs.io/en/latest/writing_ryu_app.html, last viewed May 2022.

[13] P. Biondi, Scapy, [online] Available: https://scapy.readthedocs.io/, last viewed May 2022.

Appendix 4

M. Mihaeljans and A. Skrastins
IoT concept and SDN fusion in consumer products: Overview
2023 3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)
Tenerife, Canary Islands, Spain, 2023

Proc. of the International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME 2023)
19-20 July 2023, Tenerife, Canary Islands, Spain

IoT concept and SDN fusion in consumer products: Overview

Martins Mihaeljans
Institute of Telecommunications
Riga Technical University
Riga, Latvia
martins.mihaeljans@edu.rtu.lv

Andris Skrastins
Institute of Telecommunications
Riga Technical University
Riga, Latvia
andris.skrastins@rtu.lv

***Abstract*— In this paper we study the design of Internet of Things (IoT) concept and its newly formed communication standards - Matter protocol and Thread technology. The goal of this paper is to distinguish conceptional traits of what IoT framework is and how Software defined network (SDN) functionality assists in modern IoT deployments. This paper is mainly an overview of previous studies and standards thus it concludes with considerations for necessary future research. We discovered that IoT concept bases on a vague design that lacks barriers and allows interpretations of design principles. Therefore, a common IoT framework has not yet been found.**

Keywords—IoT, SDN, Matter, Thread, Constrained network

I. Introduction

Internet of Things (IoT) devices are electronic goods with features of information extraction and enablement of process automation by use of inter-device communication. Although not in a pace forecasted IoT product consumer market grows annually making people live healthier, leisure full and serene lives.

Telecommunication industry has recognized a necessity in standardization and interoperability requirements among IoT products as consumers are reluctant to purchase more gadgets due to their inapplicability with home-hubs and smart voice-activated assistants obtained already. [3]

Industry leaders from both hardware and software manufacturers joined forces and developed a new application layer protocol Matter targeting the soaring interoperability issue. Thus, a promise – an ability to pick up any IoT item from the shelve and connect it directly to any other IoT item if only both support application layer protocol Matter.

Thread data link layer networking technology developed by Thread Group is supposed to facilitate in-premise direct communication among IoT devices without a need of central communication point or cloud-based software support.

Software defined network (SDN) has been around for quite a while and is becoming more and more trending in data center networking due to white switch ability of traffic forwarding. Numerous attempts have been made of SDN functionality adaptation to IoT networks for gain of similar centralized control on IoT things.[5] Such adaptations variate from in-premise controller to in-cloud facilitated one leaving a grey area in operations, administration, and management (OAM) requirements implied on IoT devices themselves. [2]

Further, the level of automation achievable for IoT devices derives from SDN topology in use, therefore, the northbound communication between controller and user application carries as much importance as the southbound one between controller and IoT devices.

An important aspect of IoT network is its characteristic of being a constrained network.[10] Constrained networks by default lacks one or more capabilities that regular networks does have such as sufficient processing power, limitless electrical energy, and constant reachability.

Having a constraint on even one from mentioned capabilities begs the question of the importance of a single network node and how much automation and OAM tasks it should and can be trusted with. [4]

Wireless sensor networks (WSN) have become a key enabling technology for IoT. Issues that on WSN based IoT solutions face are heterogeneity, interoperability, mobility, security and quality of service. To address these issues SDN is used forming software defined wireless sensor networks SD-WSNs. [7]

An impressive leap in usability of IoT networks meant for consumer products is what motivates this study. In addition to introduction of potential future research, this paper serves as an overview of IoT technology and its working principles in fusion with SDN functionality.

Study solely relays on previously conducted research and standardization in place at the time of investigation. The outcome of this paper is a set of discovered issues or drawbacks requiring resolution trough standardization for building more robust and user friendlier IoT networks.

II. Internet of Things concept dissect

A. Elements of IoT

By design that is defined in recommendations [14] IoT is a global infrastructure that interconnects physical and virtual things to enable advanced services used by information society as shown in fig. 1. a thing in terms of IoT is either physical (sensor, actuator etc.) or virtual (trigger, script etc.) object that is identifiable all throughout and can be integrated within communication network.

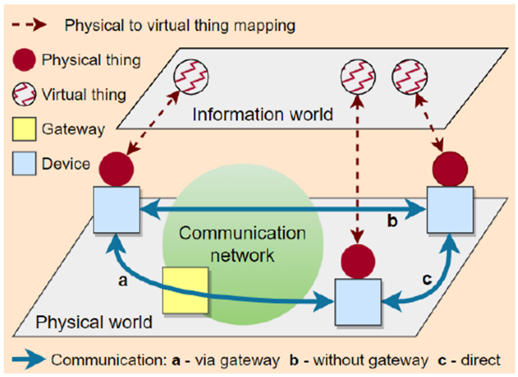


Fig. 1. Internet of Things design [14]

As shown in fig. 1. things can communicate with one another either via gateway or directly or even in some cases without requiring to traverse a communication network. IoT concept spans over both information world and physical world therefore allowing for things to be designed as real objects and virtual entities. In physical world devices represent things. For communication among devices each of them holds a virtual image in the information world. This allows them to be discovered via communication network.

Devices are categorized according to their purpose:

- **Data-carrying devices** – holds data gathered from data-capturing device and are only indirectly connected to communication network.
- **Data-capturing devices** – works as a medium between communication network and data-carrying device as its function is to read or write information.
- **Sensing and actuating devices** – convert information about the surrounding world in digital signal or act upon digital signals expressing their action within the surrounding world.
- **General devices** – are an equipment of various kinds of IoT domain devices like industrial machines, home appliances or smart phones. They have an embedded capabilities for information processing and communication.

So, by a large, an IoT device is any device mandatory facilitating ability of communication and optionally being able to sense, act upon, capture, store, and process data. The only single mandatory requirement of intercommunication allows for such interpretation as to whether a router could potential be counted as an IoT device.

For consumer products intercommunication is achieved by such communication technologies as Bluetooth Low Energy (LE), Wireless Fidelity (Wi-Fi), Thread, Near Field Communication (NFC) etc.

An IoT network under single administrative domain can span over more than one of mentioned communication technologies generating a requirement of distinguishable network topology.

In case of Wi-Fi IoT nodes are connected via central communication point a gateway, hence in a star topology. In Thread IoT devices create a mesh topology and network nodes can communicate directly with each other. A border router is required for Thread network to reach Internet.

B. Characteristics of IoT

Ambitious objectives of IoT detain its standardization:

- **Interconnectivity** – communication network should provide connection to any end device.
- **Heterogeneity** – interconnectivity provokes inclusion of any communication, any software, and any hardware.
- **Erratic by design** – forged to facilitate resource constrained networks, IoT should be resistant to outages, blackouts, latency issues and not only. Planned events like handover, reboot or shutdown must be supported. [11]
- **Colossal scale** – IoT device market grows in numbers rapidly and is predicted to keep the trend going as more and more advanced devices and handy use cases keep on gathering societies attention.

Internet faces efficiency challenge as it is becoming harder to maintain due to ever-increasing number of connected devices, hence increase in amount of data.

To aid efficiency issues research paper [6] looks at the constrained network model as a dynamic environment where its size is a part of equation to calculate connectivity constraint value for each node with use of two-dimensional search algorithm for best node selection.

What makes IoT a vertically fragmented network system is the complexity of available protocol stack, data format variety as well as plethora of communication procedures.[5]

C. Requirements for IoT

Perplexing requirements overshadow functionality:

- **Identification-based connectivity** – each thing must be identifiable to participate in communication, yet different things may hold different identifiers. To reach the objective of interoperability unification must handle issues of heterogeneity.
- **Self-regulative** – networking should facilitate capabilities like those found in ad hoc networks – self-management, self-healing, self-configuration. Data fusion might enable automation of services provided by IoT things.
- **Location-based capabilities** – to preserve lawful utilization as well as handling device relocation maintaining connectivity mapping of things whereabouts should be supported.
- **Security** – communication of things must be kept secure to ensure data integrity and trustworthy and authenticity of all communication parties.
- **Value of privacy** – most of the things have owners and users and therefore the information gathered or processed by thing might contain personal data. Things should not become an easy target for identity theft, medical record disclosure, timetable access. At the same time source of the protected information must be traceable.
- **Plug and play** – on-the-fly integration of things allow for seamless cooperation of interconnected devices and enriches user experience.
- **Manageability** – control of IoT operational workflow must reach at least as far as hold on means to ensure a normal state of network operation.[15][17]

It's been found that due to lack of interoperability between multiple application domains information gathered by IoT devices remains inaccessible. A getaway with cognitive functionality is introduced for interoperability assurance between multiple IoT domains. With use of Sensor Markup Language cognitive gateway addresses syntactic interoperability.

Tremendous growth of network traffic generated by IoT is a concern demanding for ability to share a common network and cloud infrastructure among things.[12]

Research [1] states that although a lot of research has been conducted in consideration to IoT interoperability a holistic solution for the IoT ecosystem is yet to be achieved.

Crucial barrier retaining interoperability could not be communication network reachability issue or functional error of things themselves. It could lay at the operational architecture mismatch. For example, two commonly used operational architectures are:

- **Event based** – data are being transferred only after specific even occurrence.
- **Time based** – data transmission takes place only at a specific time intervals.[5]

III. Amalgam of underlay communication

IoT requirements [16] for network identifies simple model of intercommunication that lacks any sorts of recommendation towards applicability in existing Internet infrastructure supposedly aiming for future proof concept. We distinguish only two significant networking paradigms to plug IoT into – Legacy (packet routing and switching) networks and SDN (flow matching and packet forwarding). Both of which can leverage Edge (geographically close to LAN located miniature cloud) computing for fulfilment of IoT concept objectives discussed in section II.

A. Hierarchical model of Legacy network

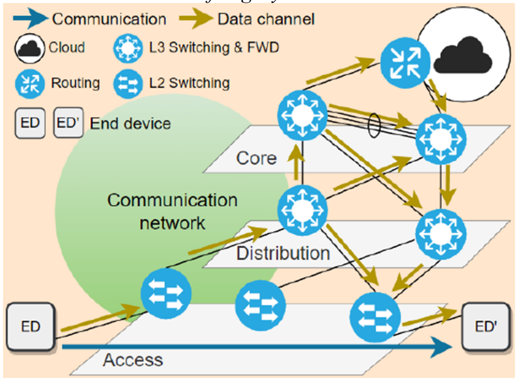


Fig. 2. Hierarchical network model

Hierarchical network model in fig. 2. is used for legacy telecommunication networks. Telecom conglomerate Cisco uses this model as bases in ordering their products for use in each of these three layers:

- **Access layer** – Connects end devices (computers, printers etc.) to the network. Usually, L2 switching devices each connect multiple end devices and via higher-speed interface aka uplink L2 switches connect end devices to either distribution layer or in a two-tier hierarchy directly to core layer devices.
- **Distribution layer** – Functions as policy executor by application of Access Control Lists (ACL) and other security mechanisms. It aggregates received data from access layer for its transmission to core layer. It also provides route aggregation and IP subnet summarization boundaries for uplinks towards core layer devices.
- **Core layer** – Functions as fast-delivery packed switching fabrics and is therefore referred to as the network's backbone. It's the interconnection point for Wide Area Network (WAN) Edges or datacenters and the Internet. It provides reliability and fault tolerance with use of multiple links between devices. Scaling in this layer is done by not using more devices rather replacing existing ones with more powerful and faster devices. [20]

The Internet backbone relies on BGP routing – a legacy network. All devices in this model can work autonomously.

In fig. 2. depicted data channel represents possible network traffic directions versus intended communication vector. Data channel used depends on physical and virtual distance between end devices and network devices. Telecommunication companies holds on to hierarchical network model as it is easy to deploy and upgrade.

B. Software defined networking

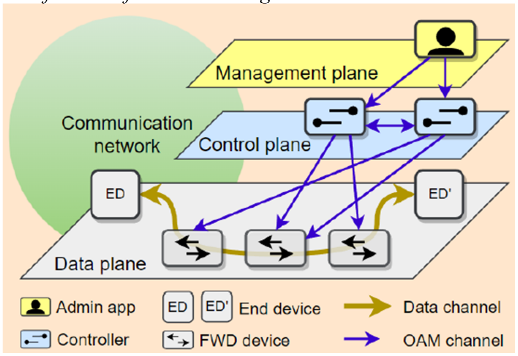


Fig. 3. Software defined network model

Fig. 3. shows SDN model. The idea behind SDN is the ability to have a single point management with which to control all network devices. Unlike in a hierarchical network model used in legacy networks not all SDN layers called planes are used for data channel. Upper planes are used only for network traffic control and not forwarding. Functions of SDN planes are as follows:

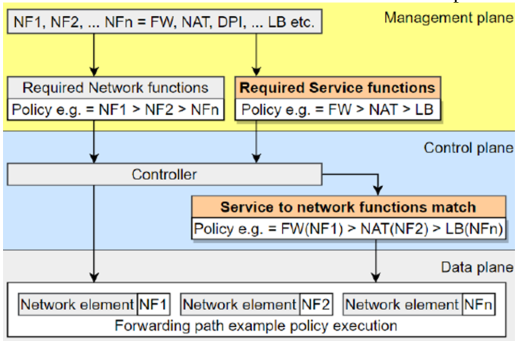
- **Data plane** – Facilitates network traffic forwarding devices, most often, white switches which does not follow typical routing and switching principles. They execute orders and forwarding policies setup by the controller. White switches like Open Virtual Switch (OVS) connect end devices to the network.[8]
- **Control plane** – The main purpose of a network controller is to oversee data plane and translate application tasks into flow policies for execution in forwarding devices (white switches). A byproduct of such control mechanism is an ability to proactively generate forwarding policy on demand.
- **Application plane** - Functionates as an interpreter between user input and controller. There are various ways to communicate but common is representation state transfer (REST) application programming interface (API). [9]

In OVS each forwarding policy consist of packet matching information, action instructions and policy entry state information. In SDN there are three flow rule base installation types:

- **Proactive** – White switches already hold the necessary information for arriving packet match.
- **Reactive** – Necessary information about correct packet forwarding is generated at the time of packet arrival with controllers help.
- **Hybrid** – this type can function both proactively and reactively. That is achieved by administrator loading network traffic classification policy on data switches manually but allowing controller to change them dynamically as well.[5]

A cost saving solution is to not use out of band controller and use the same links for both the date and control. The control channel (OAM cannel in fig. 3.) in centralized control SDN can become a bottleneck. [12]

Diverse communication technologies may rely on various security mechanisms creating enclosed security domains. This rises global trust issues among the whole network. Therefore an unified trust mechanism is required.



NF1, NF2, ... NFn = FW, NAT, DPI, ... LB etc.
Management plane
Required Network functions
Policy e.g. = NF1 > NF2 > NFn
Required Service functions
Policy e.g. = FW > NAT > LB
Control plane
Controller
Service to network functions match
Policy e.g. = FW(NF1) > NAT(NF2) > LB(NFn)
Data plane
Network element NF1
Network element NF2
Network element NFn
Forwarding path example policy execution

Fig. 4. Service Function Chaining in SDN [13]

Service Function Chaining (SFC) shown in fig. 4. is a way of decoupling service function path from physical network topology in SDN.

In legacy networks network services like Firewall (FW), Deep Packet Inspection (DPI). Network Address Translation (NAT) or Load Balancing (LB) and others are provided as proprietary solutions of specific vendor. Therefore, they are constrained by vendor specific update policy and capabilities. Virtualization and SFC allows to escape this dependency.

SFC allows network traffic to be steered trough different than physical path route to change service application order on directed network traffic. We have covered this SDN capability in detail in our previous research [13].

C. Edge computing

To minimize delay from cloud to IoT an edge cloudlet can be used. Edge computing is the mechanism where intensive computational processes are performed that IoT devices themselves cannot do locally. This computation is an efficient way to avoid peek loads.

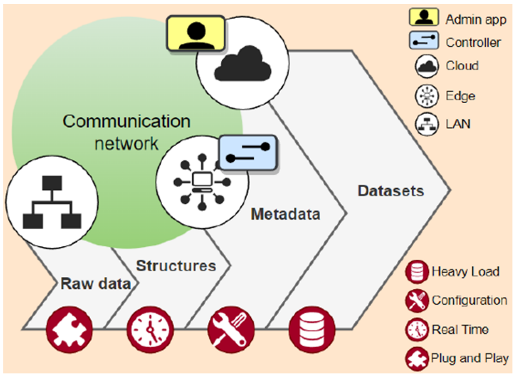


Fig. 5. IoT data processing with Edge computing

IoT data processing with use of Edge computing is shown in fig. 5. IoT devices like sensors and actuators generate various date that not only differentiate with their content but also with their intent. For example, thermal sensor data can be used for instant room temperature regulation via actuator nearby therefore there would be no need for further data processing and it shouldn't be transmitted outside local area network (LAN).

However the same data could be used for autonomous climate regulation in future by basing necessary ventilation system workflow on previously gathered data. In such case from raw data a dataset needs to be created. Such heavy computational process can only happen in cloud data centers.

Between LAN and Cloud is Edge, which is a miniature version of cloud physically placed someplace closer to the LAN and intended for real time operation handling. If data processing requires higher computational performance it travels up from the edge to the cloud.

In example of thermal management the use of edge would be trigger and alert system deployment as for disaster mechanisms to work effectively raw data constructed in structures of predefined events would travel to the edge for quick processing. Edge therefore holds both real time processing and configuration duties as it has to be able to act upon operation, administration and maintenance (OAM) messages as well. OAM messages are a form of metadata. In their simplest form metadata are data explaining other data. In this case metadata would be configuration policies.

Smartwatches, phones, and health monitoring bracelets all constantly produce and transmit data. For proper functioning a communication edge must meet a sufficient level of quality and service QoS. Transferring most of the computational processes directly to cloud would increase latency therefore an edge is in use.

Enabling edge is a complex task that includes fault tolerance, mobility management, and device authentication subtasks. To ease the completion of these subtasks edge virtualization is introduced. Such objective is fulfilled by network function virtualization (NFV).

In edge computing scenarios other issues arises like user handover, service discovery and mobility management. IoT development is disjoint and therefore lacks standardization leading to interoperability issues and security challenges.

IV. Thread Networking Technology

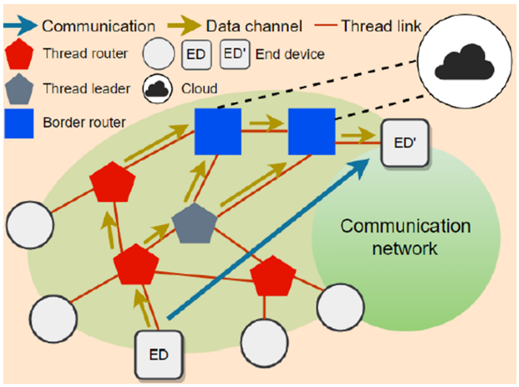


Fig. 6. Thread network design

Thread network design is shown in fig. 6. Thread is a mesh network. End devices can communicate through different thread routers. Any of thread links can be used for data transfer therefore communication path shown with yellow arrows can varied from communication vector show with the blue arrow. It is foreseen that end devices themselves can fall under variable constraint challenges like battery power, bandwidth, CPU processing limitations, therefore they only communicate with parent routers and not directly. [19]

Thread network by itself has no single point of failure. As the role of a router can be obtained by election process sleepy nonresponsive devices can lose their status of a router and the leader of a network segment can assign another device to perform routing task. Leaders in Thread network most commonly are border routers unlike in fig. 6.

In a case where network topology spans over more than one communication technology and there is only one border router in between this border router becomes a single point of failure as there is no other device that can take its place. Thread devices are:

- **Border Router** – A device that provides connectivity from Thread network to adjacent networks of different physical layers like Wi-Fi or Ethernet etc.
- **Router** – A device providing routing services to other network devices. Additionally, routers secure the network and allow or deny new devices to join it. Routers are not supposed to be sleepy end devices (SEDs), but they have an ability to downgrade to Router-Eligible end devices REEDs.
- **REED** – a device eligible to become a router but due to the network topology does not. REED is not a device type but a state of readiness to become a router if necessary. Thread network can menage the transition of a device from REED state to a router state without any requirement of user interaction.
- **Sleepy End Devices** – an IoT devices which only communicate with router and does not facilitate message forwarding capability of forwarding other device messages.

Not only all communication in Thread network is secured with a network key but it also has a process of joining a new device which requires authentication, authorization, and completion of a key agreement mechanism.

Thread uses IPv6 at a network layer therefore it is capable of routing its traffic to and from the Internet although Thread is specifically designed that it would not require the Internet for end device communication.

6LoWPAN is more suitable as network encapsulation protocol than Routing Protocol for Low-Power and Lossy Networks (RPL) as it better facilitates scalability.

IoT devices communicating with one another should agree upon most effective data transmission for both memory and power preservation as most often there are constraints in these resources. [4]

IEEE 802.15.4 low-power radio is supported. This data transfer technology is specially designed for end devices like door locks or window alarm sensors that are built to have their battery to last for several years.

V. MATTER PROTOCOL

Over time IoT protocol stack has grown and expanded as older protocols has been found inefficient or unsecure.[4] Protocol diversification has lead to a requirement for a need of a unified solution as making border routers to support multiple protocols would be excruciating task.

Smart home is one of the most promising of IoT markets in the whole ecosystem as it has a market space that has a potential consumers where traditional home appliances are in use already. While Industrial IoT consumers can and does have a budget for custom solution foundation for their industrial operations, home users on the other hand does have to relay on products developed for universal use. Matter is a standard for delivering interconnection between these universal use products. Matter is an industry-driven protocol developed by many industry leading companies. [18]

For communication Matter uses such technologies as Ethernet, Thread, Wi-Fi. It can seamlessly span over all three with use of capable boarder routers.

For automated device authentication and commissioning in the users home network Matter uses Bluetooth LE and NFC. Manual device commissioning is also supported via product number or QR code input by users themselves.

For developers Matter provides a specifications and white papers on application creation with Matter protocol support on devices like lighting switch, smart TV etc. in great detail describing even such actions as press and long press.

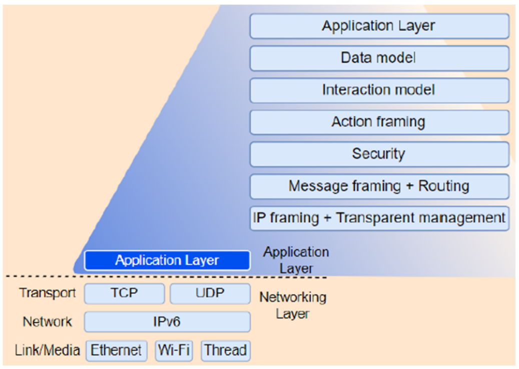


Fig. 8. Matter protocol stack

In fig. 8. Matter protocol is depicted. Matter protocol resides in TCP/IP protocol stacks application layer. Formerly known as Connected Home over IP (CHIP) Matter is meant for application in consumer products for smart home IoT devices. Its purpose is to serve a common communication channel for applications from various manufacturers.

Matter is divided in flowing sublayers:

- **Application Layer** – Consists of business logic aka things functionality (climate control, lighting etc.) having sublayers name as remark of functionalities previous placement in TCP/IP stack.
- **Data model** – Sort of a presentation layer aiding functionalities translation into a common data and verb namespace.
- **Interaction model** – Specifies the performable interactions between a client and server device. These actions can be carried out on elements defined by data model.
- **Action framing** – Constructed action specified in interaction model is serialized and form an encoded packet binary format ready for transmission.

- **Security** – Encryption is done on the reamed action and additional signing of payload is applied for insurance of secure communication between sender and the recipient.
- **Message framing and Routing** – Defines message properties by specifying additional payload fields and routing information.
- **IP framing and Transparent management** – with constructed payload message is sent down to underlaying transport layer for further processing and finally transmission via communication network.

Matter has no network ownership regulations therefore it considers network as a sharable space. This allows for many Matter overlays to coexist among same IP devices. Matter does not require use of the Internet for its operation.

VI. Conclusions

This study serves as broad overview of the IoT concept and its current applicability in consumer-based products. Previous works tend to focus on either comparison between different IoT protocols or introduction of customized IoT communication technologies but not on the general IoT concept. The lack of research of IoT conceptional evolvement is what motivates this study.

We found that IoT concept holds no clear boundaries of its technical deployment and functional purpose. There is no clear understanding of what devices belong to IoT concept. It's not defined whether underly communication network itself and its means of data transfer belong to IoT network.

Most of IoT and SDN fusion proposals relay on obsolete OpenFlow protocol usage and therefore would require redesign for application on state-of-the-art programmable network devices or similar future technologies.

Thread networking technology for consumer products facilitates no SDN functionality and therefore is not sufficient for fulfilment of complete IoT requirement set defined in section II.

Thread is supposedly only proposed for use in IoT area of the whole IoT network defined in recommendations.

With no consensus of a unified IoT application layer framework, manufacturers agreed upon an alternate solution - Matter intermediate layer, thus favoring a support of already existing IoT ecosystems. This puts additional unnecessary resource load on already constrained devices.

This study indicates a necessity for IoT concept framework proposal and its communication domain boundary exploration in future research.

VII. References

[1] T. Adesina, O. Osasona “A Novel Cognitive IoT Gateway Framework: Towards a Holistic Approach to IoT Interoperability” in 5th World Forum on Internet of Things (WF-IoT), 2019

[2] M. Wei, E. Liang and Z. Nie “A SDN-based IoT Fine-grained Access Control Method” in International Conference on Information Networking (ICOIN), 2020

[3] F. Montori, L. Bedogni and F. Morselli, L. Bononi “Achieving IoT Interoperability through a Service Oriented In-Home Appliance” in Global Communications Conference, 2017

[4] C. Sharma, N. Gondhi “Communication Protocol Stack for Constrained IoT Systems” in 3rd International Conference On Internet of Things: Smart Innovation and Usages, 2018

[5] W. Rafique, L. Qi, I. Yaqoob and M. Imran, R. Rasool, W. Dou “Complementing IoT Services Through Software Defined Networking and Edge Computing: A Comprehensive Survey” in Communications Surveys & Tutorials Volume: 22, Issue: 3, 2020

[6] F. Yang, C. Ma and C. Chang, J. Huang “Constrained Optimization of Critical Node Detection with Cost Model in Communication Networks” in 5th Information Technology, Networking, Electronic and Automation Control Conference, 2021

[7] T. Theodorou and L. Mamatas “CORAL-SDN: A Software-Defined Networking Solution for the Internet of Things” in Conference on Network Function Virtualization and Software Defined Networks, 2017

[8] H. Yoon, S. Kim and T. Nam, J. Kim “Dynamic Flow Steering for IoT Monitoring Data in SDN-coordinated IoT-Cloud Services” in International Conference on Information Networking, 2017

[9] I. Kassem, A. Sleit “Elapsed Time of IoT Application Protocol for ECG: A Comparative Study Between CoAP and MQTT” in International Conference on Electrical, Communication, and Computer Engineering, 2020

[10] R. Beniwal, K. Nikolova and G. Iliev “Energy Efficient Routing Protocols in Resource Constrained IoT Networks” in 27th National Conference with International Participation, 2019

[11] C. Tselios, I. Politis and S. Kotsopoulos “Enhancing SDN Security for IoT-related deployments through Blockchain” in Third International Workshop on Security in NFV-SDN, 2017

[12] L. Ogrodowczyk, B. Belter and M. LeClerc “IoT ecosystem over programmable SDN infrastructure for Smart City applications” in Fifth European Workshop on Software-Defined Networks, 2016

[13] M. Mihaeljans and A. Skrastins, “Network Topology-aware Service Function Chaining in Software Defined Network”, in 28th Telecommunications forum TELFOR, 2020

[14] Recommendation ITU-T Y.2060 “Overview of the Internet of things”, 2016

[15] Recommendation ITU-T Y.2068 “Functional framework and capabilities of the Internet of things”, 2015

[16] Recommendation ITU-T Y.4113 “Requirements of the network for the Internet of things”, 2016

[17] Recommendation ITU-T F.748.0 “Common requirements for Internet of things (IoT) applications”, 2014

[18] Connectivity Standards Alliance “Matter 1.0 Core Specification”, 2022 [online] Available: https://csa-iot.org/developer-resource/specifications-download-request/

[19] Thread Group “Thread 1.1.1 Specification”, 2022 [online] Available: https://www.threadgroup.org/ThreadSpec

[20] Cisco Press “Cisco Networking Academy Connecting Networks Companion Guide: Hierarchical Network Design”, 2014 [online] Available: https://www.ciscopress.com/articles/article.asp?p=2202410&seqNum=4

Appendix 5

M. Mihaeljans and A. Skrastins
Openthread Network Density Evaluation: Quantitative Analysis
2023 Symposium on Internet of Things (SIoT)
São Paulo, Brazil, 2023

Openthread network density evaluation: Quantitative analysis

Martins Mihaeljans
Institute of Telecommunications
Riga Technical University
Riga, Latvia
martins.mihaeljans@edu.rtu.lv

Andris Skrastins
Institute of Telecommunications
Riga Technical University
Riga, Latvia
andris.skrastins@rtu.lv

Abstract—**In this study we explore Openthread technology. From simulations run in Openthread network simulator (OTNS) we examine its working principle. Simulation results shows technologies suitability for constrained IoT network devices as it generates relatively low overhead. Which is 20 times smaller than the throughput (250kbit/s) available on average. During repetitive simulations we discovered that Openthread routers tend to lose connectivity with end devices especially on topology changes including breakage of links unrelated to changes themselves.**

Keywords—CoAP, IoT, SDN, Thread, Openthread

I. Introduction

Internet of Things (IoT) is foreseen to become widely adopted due to advancements in communication technologies enabling the Internet's ubiquitous availability. This adoption would ensure numerous benefits in human day-to-day life.

Numerous attempts of inhouse proprietary technology development and marketing by tech giants like Samsung, Google and Apple has left the consumer market scattered, where smaller manufacturers tend to support either but not all platforms leaving customers in the grey area of managing the interoperability manually or not having such at all.

Recently this issue has been recognized and addressed by the telecom industry leaders and a two-fold solution has found its way on store shelves. A Thread network technology would guaranty interoperability between devices physically while Matter protocol would enable it at the application layer.

In this study we explore Openthread technologies working principle via examination of multiple simulations all executed within Openthread Network Simulator (OTNS). Openthread by Google is a version of Thread technology free of charge to facilitate more rapid industry adaptation. Results from our simulations shows that Openthread is suitable for use with constrained IoT devices. However, on topology changes a sort of chaotic broadcasts of network status update requests causes link disjoints prolonging topologies recovery process.

II. Related works

Authors of [1] state that the Internet has not been designed to accommodate resource constrained IoT devices. Named Data Networking as an alternative to the TCP/IP model is discussed for support of these devices. Although not in context of Thread mesh networking, this study highlights topologies effect on IoT device communication highlighting motivation for our study goal of figuring out the Thread overheads value in different density meshes.

Not only in Thread based but in IoT networks in general power consumption is a dominant matter. Study [2] discuses IoT based application energy efficiency for wireless sensor networks (WSN). Modified Low Energy Adaptive Clustering Hierarchy (LEACH) protocol is introduced. Achieving longer network lifespan and more reliable communication. This study goes together with our motivation of figuring out how efficient it is to have more routers in Thread network.

A study [3] of Thread mesh network efficiency in a harsh environment facilitates findings in our study of Thread links being easily detachable. Even more so, a test had been carried out in real world environment and concluded with claim of Thread be fit for non-real time measurement solutions.

Authors of study [4] built a test environment of Openthread network in an office building and discovered that due to occasional Wi-Fi signal interference Thread mesh is forced to do rerouting. Impact of these accuracies we have noticed in our simulations as well. They concluded that an end-to-end feedback mechanism could decrease packet loss evident in larger networks. We argue that such feedback mechanism would only be possible via enablement of SDN functionality discussed in paper [5].

Study [6] found that a noticeable issue in routing for Thread networks is their inability to handle multi personal area networks (PANs) scenarios. Authors developed flaying IoT test network where occasionally drones would separate more than the radio radius of their antennas therefore losing direct communication which would cause these nodes to regenerate their PANs and once drones would be in one another's range before they could communicate another PAN regeneration should take place. During our simulations specifically in nine router topology shown in fig. 6 this PAN regeneration caused a disruption in FED communication on multiple runs.

In science paper [7] from the comparison of various IoT communication protocols and technologies authors stated that a universal IoT solution is not to be found as existing ones already are tailored for best performance according to their use case. Thread protocol got categorized as fit for dense small are networks which reflects also on our conclusion as the main Thread drawback is the inability of multiple PAN routing which might not play a huge role in small area networks such as a household.

Thread protocol was used for health monitoring systems for elderly peoples in old age homes in paper [8]. This paper emphases Threads mesh as a fit solution as data from sensors is not a single point dependable but can reach sink node via multiple paths. During our simulations we witnessed how secondary Thread mesh links greatly compensate router outages and improves reachability of end devices.

An in-depth review of both Thread and its partial predecessor Routing Protocol for Low-Power and Lossy Network (RPL) has been given in [9]. Authors point out that while the design assumptions of RPL are correct the use cases in real life scenarios have proven that IoT networks does not require as much flexibility in variation of physical link layer structure as they do at a transport and application layers. Therefore, IoT networks require adequate reliability both from and to sensor nodes, as with that not only data sensing is enabled but also actuation. This is reflected in our study as by simulation of Ping traffic that is in general only is successful when a response reaches the sender. Thread has been built to measure bidirectional link quality and use it as a metric for safe data delivery.

III. Thread Technology Bases

Thread [10] is a network technology designed for use in consumer products for smart home applications. It focuses on providing connectivity cost-efficiently and securely in a IPv6 network. Fig. 1. displays a Thread network where topology is a mesh and multiple data channels are available. The topology consists of routers and end devices.

There are different types of end devices. For instance, full end devices (FEDs) will always have their radio on, but sleepy end devices (SEDs) will disable their radio while idle.

Thread leader maintains the construction of Personal Area Network (PAN) by assigning IP addressing and security details to its child devices. In Thread directly connected devices form child parent relationship creating sort of a two-tier mesh topology. There can be up to 32 routers in a PAN.

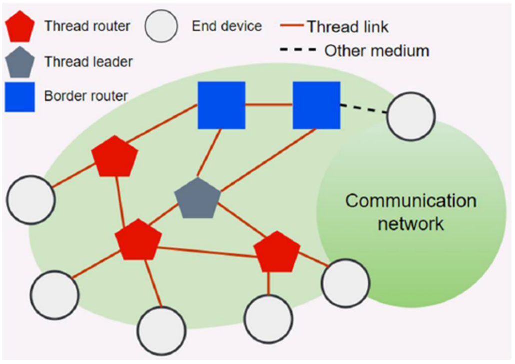


Fig. 1. Thread network design

Thread protocol stack is shown in fig. 2 along TCP/IP model for comparison. Although the accent is put on use of wireless communication means Thread can also be used over ethernet. When multiple paths are available the best path is calculated for bidirectional signal quality by measuring the Received Signal Strength Indicator (RSSI) value. At network layer a distance vector routing RIPng for IPv6 with reduced headers is in use in Thread [7]. At transport layer Thread uses UDP but it is capable to use TCP Low Power (TCPlp) as well.

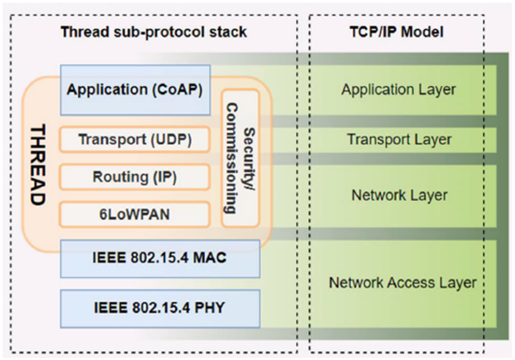
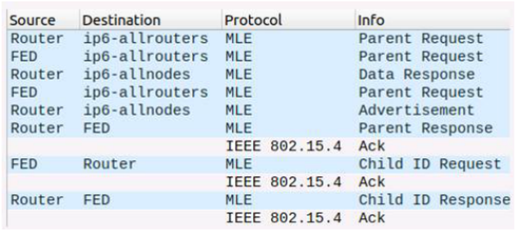


Fig. 2. Protocol stack

As shown in fig. 3 most of Thread communication consists of Mesh Link Establishment (MLE) messages that contain in themselves a Type-length-value (TLV) fields. Commonly as an acknowledgment replay a MAC layer Ack is in use. In fig. 3 a process of link establishment between Thread router and full end device (FED) is shown.



```
Source   Destination     Protocol            Info
Router   ip6-allrouters  MLE                 Parent Request
FED      ip6-allrouters  MLE                 Parent Request
Router   ip6-allnodes    MLE                 Data Response
FED      ip6-allrouters  MLE                 Parent Request
Router   ip6-allnodes    MLE                 Advertisement
Router   FED             MLE                 Parent Response
                         IEEE 802.15.4       Ack
FED      Router          MLE                 Child ID Request
                         IEEE 802.15.4       Ack
Router   FED             MLE                 Child ID Response
                         IEEE 802.15.4       Ack
```

Fig. 3. Communication basics

IV. Network Simulation

In this study we used Openthread Network Simulator (OTNS) available at Openthread GitHub [11], even though it has not as intuitive design in comparison to other network simulators, it's a polished enough version in comparison to its own predecessors found in studies [12].

Overhead and reachability are two parameters of interest of our analyses. We measured Thread technologies generated overhead from MLE and MAC layer packet network traffic amount. We expressed it as operational traffic intensity measured in packets per second (pps). Reachability was measured by count of successful ping responses. Duration for each simulation was 10 minutes (600 sec.). During our analyses more than 120 simulations under various topologies were run. From each simulation a packet capture was collected and analyzed via Wireshark. Capture example can be seen in fig. 3.

Each topology variates from one-another with mesh depth, respectively Five-router topology (shown in fig. 5) has less possible data paths than nine-router topology (shown in fig. 6), and thirteen-router topology (shown in fig. 7) has more data paths in comparison to both nine-router and five-router topologies.

Each case scenario (shown in fig. 4) holds different Thread network activity according to processes run during simulation time and are as follows:

- Scenario 1. – Silent FEDs – only Thread mesh is generated and maintained, and no user data traffic at all from full end devices (FEDs) is generated.
- Scenario 2. – Active FEDs – Thread mesh is generated and maintained, and ping packets (user data traffic) are sent across the network.
- Scenario 3. – Single router failure – Additionally to Thread and Ping processes one router gets disconnected causing topology changes.
- Scenario 4. – Two routers failure – Additionally to Thread and Ping processes this time two routers are disconnected from topology causing bigger topology changes in comparison to those in scenario 3.

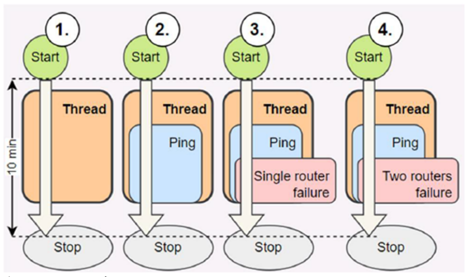


Fig. 4. Case scenarios

Simulated mesh topologies are shown in figures 5, 6 and 7. Each of these figures are divided in four numbered quadrants as follows:

- Quad. 1. – Full mesh topology.
- Quad. 2. – Network traffic path in full topology.
- Quad. 3. – Impact of one router disconnect.
- Quad. 4. – Impact of two router disconnect.

As stated in table I router count in each mesh is different, therefore, the impact of removing a single or two routers differs between topologies as well. Router count changes the overall network mesh density.

Fig. 5 shows topology with five routers. In quad. 3 of fig. 5 communication between two of the FEDs is lost as one router has been disconnected. In quad. 4 of fig. 5 due to two routers disconnect communication in the whole network is disrupted and previous mash topology has segmented itself in four independent PANs. These PANs do not communicate with one another. Quad. 4 of fig. 5 also demonstrates how time-based Thread wireless networks are as the remaining routers have not formed proportionally sized PANs but the child-parent relationships have been established on first chance. Respectively, two-tier mesh forms where routers talk with one another, but end devices only talk to routers but not directly to one another. Evidently one of the FEDs has been left alone as in its radio range there are no other routers.

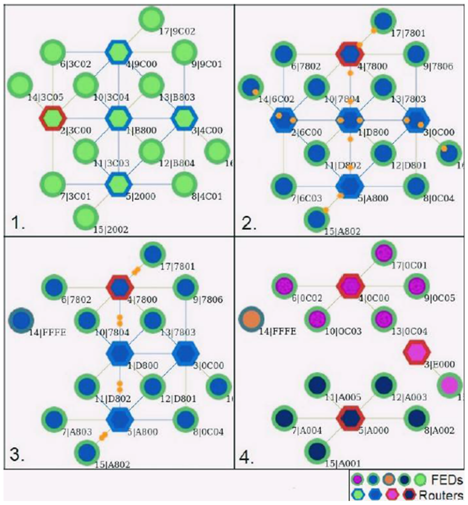


Fig. 5. Five-router topology

In fig. 6 a topology with nine routers in it is shown. On a single router disconnect depicted in quad. 3 of fig. 6 it is evident that mesh network has managed to continue both communications. However, in quad. 4 both communications have been disrupted fully. Full disruption has happened because PAN has been regenerated for all nodes and not only for those that had a direct link with the disconnected routers. So, while PAN regeneration process took place FEDs ended their retry process after several failed attempts. Interestingly, such full PAN regeneration did happen only occasionally. Unfortunately, we can't make any claim on leader placement or user data traffic path relation to these occurrences.

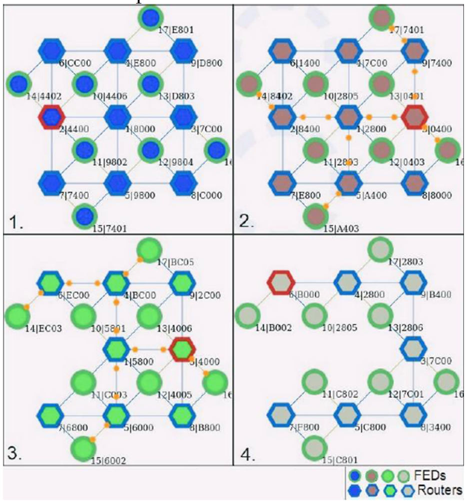


Fig. 6. Nine-router topology

Thirteen router topology is shown in figure 7. Quad. 3 and quad. 4 shows that no communication has been interrupted after topology changes. This represents a selfheal capability improvement over topologies with less routing capable nodes in them. Figure 7 shows the lack of topology control as both communications has at least one hop of their paths overlapping.

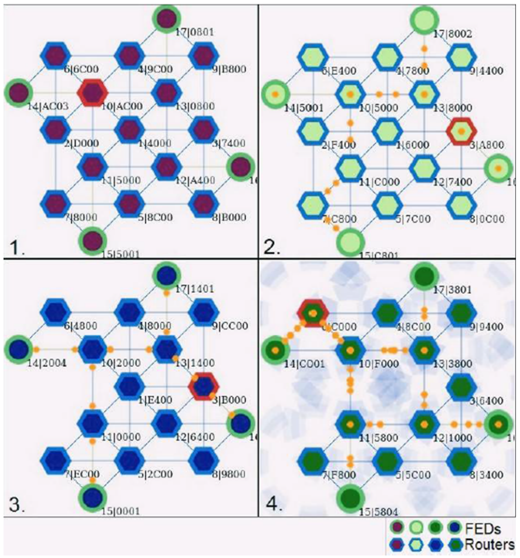


Fig. 7. Thirteen-router topology

V. Results

From over 120 simulations we calculated mean points and linear regression for each case scenario shown in figure 4. We chose linear regression to be a fit parameter for observation of non-simulated values as the increase or decrease in operational traffic intensity is directly proportional and dependent to node roles, data transfer and node liveliness.

A linear regression formula (1) is as follows:

$$a * x + b = y \qquad (1)$$

Where x in this study is normalized router count, y is operational intensity, but a and b are coefficients generated from mean points with use of ***polyval function*** in Octave.

Calculation results are shown in table I. Results are sorted according to topologies. Router count reflects on topology i.e., value five – five-router topology. As the values of linear regression are close in proximity to those of mean points, we accept it as a fit parameter for observation of sequential topology variations forming overall study results.

TABLE I. OVERHEAD CALCULATION RESULTS

Nr.	Thread network operational intensity in packets per second			
	Router count	***Scenario from fig. 4***	***Mean point [pps]***	***Linear reg. [pps]***
1.	5	1. Silent FEDs	1.1052 (±0.081)	1.164
2.	5	2. Active FEDs	6.4765 (±0.109)	6.816
3.	5	3. Single router fail.	5.4613 (±0.432)	6.042
4.	5	4. Two routers fail.	4.2067 (±0.296)	4.788
5.	9	1. Silent FEDs	1.6955 (±0.257)	1.576
6.	9	2. Active FEDs	8.2865 (±0.477)	7.606
7.	9	3. Single router fail.	8.8160 (±1.397)	7.654
8.	9	4. Two routers fail.	8.1837 (±2.538)	7.019
9.	13	1. Silent FEDs	1.9275 (±0.182)	1.987
10.	13	2. Active FEDs	8.0557 (±0.593)	7.606
11.	13	3. Single router fail.	8.6867 (±1.053)	9.267
12.	13	4. Two routers fail.	8.6680 (±1.035)	9.250

For the analyses of data obtained from the simulations we chose to use Octave as it is a freeware meant for numerical computations. Thanks to its rich mathematical library we obtained Thread network overhead graphs shown in figures 8 and 9, and network reachability graph shown in figure 10.

Steps for overhead calculation are as follows:

1. Getting mean points from simulation data with use of ***mean function***.
2. Getting standard deviation from simulation data with use of ***std function***.
3. Generating normalized router count vector by dividing router count within each simulation with maximal router count in topology which is 17.
4. From vector of mean points respectively to each measured case scenario and vector of normalized router count create coefficient a and b values with use of ***polyval function***.
5. Generating new router count vector from all values that are possible normalized by maximum router count in topology.
6. Creating a ***linear regression*** from newly created router count vector and previously gained a and b coefficients according to formula (1).

Reachability (shown in fig. 10) calculation required only acquiring average values from simulation data with use of ***mean function***.

Thread activity expressed in operational traffic intensity and measured in packets per second (pps) during both silent and active communication periods between FEDs is shown in fig. 8. The tipping point at router count value 1 is an anomaly that could be eradicated using more advanced mathematical expressions, or by including value 0 in initial polyval vector. We chose to not to as it would only create calculation error because in Thread networks there is no communication between FEDs directly.

The approximate 6 second gap between active and silent values can be explained with mesh being created wirelessly which causes Thread to send MAC layer ACKs on each data traffic packet transmission (No ping packets are included in calculation of values in graphs).

It is evident that silent FEDs linear regression slope is narrower (starting from below 1 to above 2 secs.) than active (starting at 6 sec. to above 9 sec.). This could be explained by ACK packet count overweighing MLE packet count as MLE packet are only sent on routing path updates and during topology creation as well as keepalive.

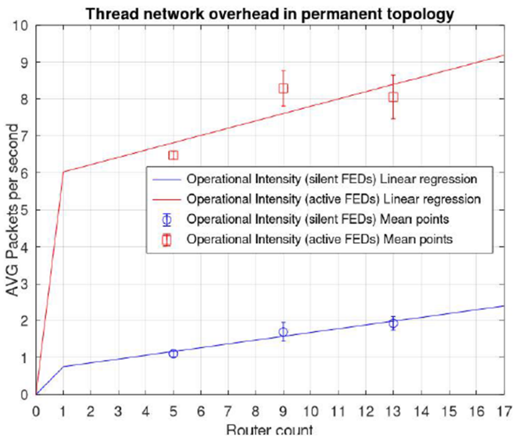


Fig. 8. Thread network overhead in permanent topology

Fig. 9. depicts operational traffic intensity for case scenarios where topology changes are present. These changes cause routing table update requests and for some instances even regeneration of PAN itself. This causes visible increase in slope of linear regressions and larger standard deviation from mean points.

Linear regression of single router failure at a router count 1 is higher than two routers failure as one communication is disrupted, as more MAC layer ACK traffic is generated. Situation is reversed from router count 13 as the impact of two routers failure is more present than impact of a single router failure as in denser mesh more links would disappear.

Prominent standard deviation for nine-router topology is due to specific node placement as shown in fig. 6. On router failure many central links are lost resulting in data traffic retransmission causing additional MAC layer ACK traffic as well as MLE routing table update messages and, in some case, even new PAN formation.

The slope of Single router failure case scenario shown in fig. 9 in comparison to Active FEDs scenario shown in fig. 8. Has more than 2 times steeper rise (increase in pps) which is caused by broadcast of routing table update.

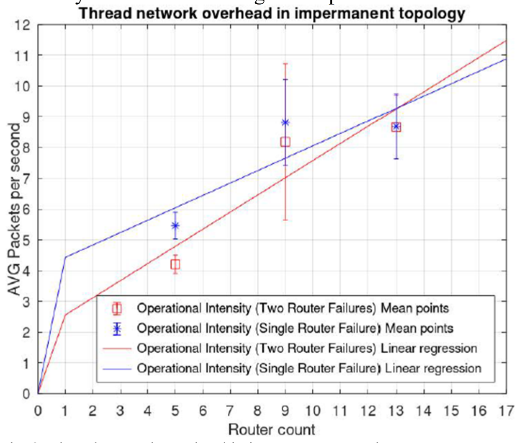


Fig. 9. Thread network overhead in impermanent topology

Thread network reachability is shown in fig. 10. Thirteenrouter topology managed to successfully deliver most of the data traffic (ping packets) both at a Single router failure and Two routers failure case scenarios. Similarly proportioned decreases in successful data traffic delivery among all three topologies evidently reflects on transmission failures doe to relatively slow routing table updates and nodes not being swift on rerouting via secondary paths.

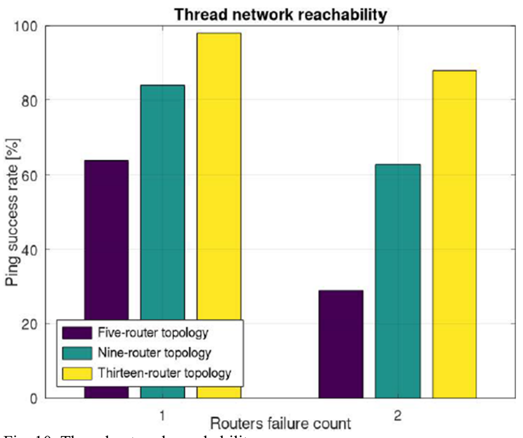


Fig. 10. Thread network reachability

As we aimed to simulate Thread resistance to topology changes, we kept disconnecting centrally positioned routers during all simulations. Consequently, causing maximal effect on a router failure. Thus, reachability is a parameter from which we can deduct that with more routing capable nodes in the network an overall reachability will be higher than with less routers. Seemingly, thirteen-router topology kept reachability above 80% even after two routers disconnect due to secondary link availability.

CONCLUSIONS

This study is an analyses of an industry accepted Thread IoT communication network technology meant for use in consumer products of smart home applications. We used Openthread network simulator for measurement of Threads generated overhead both in silent and active communication periods.

Results indicate a small overhead, which is 20 times smaller than available throughput (250kbits/s) and good reachability (above 80%) for topologies with more routing capable nodes.

The parent-child relationship between routers and their adjacent end devices should not be overlooked and router count in topologies should be kept relatively high for mesh with at least one secondary path to each end device. From our simulated topologies five-router topology (shown in fig. 5) could serve as a good example of not having enough routers for a proper mesh network formation.

A noticeable shortcoming is Threads inability to route network traffic over multiple personal area networks causing additional link regeneration processes, however, this might not have a sever impact at small household environments, Therefore, we acknowledge Thread (Openthread) to be fit for use in home applications due its low overhead and good network reachability even under topology changes.

REFERENCES

[1] C. Langa, P. Tarwireyi and M. Adigun, "Evaluating Named Data Networking forwarding strategies in different IoT topologies," 2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD), 2020
[2] T. N. Nguyen, C. V. Ho and T. T. T. Le, "A Topology Control Algorithm in Wireless Sensor Networks for IoT-based Applications," 2019 International Symposium on Electrical and Electronics Engineering (ISEE), 2019
[3] W. Rzepecki, Ł. Iwanecki and P. Ryba, "IEEE 802.15.4 Thread Mesh Network – Data Transmission in Harsh Environment," 2018 6th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), 2018
[4] A. I. Grohmann, D. Nophut, M. Sobe, A. B. Perez and F. H. P. Fitzek, "Interference resilience of Thread: A practical performance evaluation," 2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC), 2021
[5] M. Mihaeljans and A. Skrastins, "Evaluation of reactive service function path discovery in symmetrical environment," Telfor Journal, vol. 14, no. 1, pp. 2-7, 2022
[6] D. Jaisinghani, T. U. Rehman, R. Mulkey and A. Berns, "IoT in the Air: Thread-Enabled Flying IoT Network for Indoor Environments," 2023 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops), 2023
[7] W. Rzepecki and P. Ryba, "IoTSP: Thread Mesh vs Other Widely used Wireless Protocols – Comparison and use Cases Study," 2019 7th International Conference on Future Internet of Things and Cloud (FiCloud), 2019
[8] A. G. Biradar, S. Johari, S. S. Kulkarni, A. Maheshwari and K. Venkatesh, "OpenThread Based Mesh Enabled IoT Smart Device Cluster for Health Monitoring of the Elderly in Old Age Homes," 2020 4th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech), 2020
[9] H. -S. Kim, S. Kumar and D. E. Culler, "Thread/OpenThread: A Compromise in Low-Power Wireless Multihop Network Architecture for the Internet of Things," in IEEE Communications Magazine, vol. 57, no. 7, pp. 55-61, July 2019
[10] Thread Group "Thread 1.1.1 Specification", 2022 [online] Available: https://www.threadgroup.org/ThreadSpec, last viewed August 2023
[11] Openthread Network Simulator (OTNS), [online] Available: https://github.com/openthread/ot-ns, last viewed August 2023
[12] J. Gopaluni, I. Unwala, J. Lu and X. Yang, "Graphical User Interface for OpenThread," 2019 IEEE 16th International Conference on Smart Cities: Improving Quality of Life Using ICT & IoT and AI (HONETICT), 2019

Appendix 6

M. Mihaeljans and A. Skrastins
Efficient Multipath Service Function Chaining in Inter-Data Center Networks
2023 31st Telecommunications Forum (TELFOR)
Belgrade, Serbia, 2023

31st Telecommunications forum TELFOR 2023 Serbia, Belgrade, November 21-22, 2023.

Efficient Multipath Service Function Chaining in Inter-Data Center Networks

Martins Mihaeljans
Institute of Telecommunications
Riga Technical University
Riga, Latvia
martins.mihaeljans@edu.rtu.lv

Andris Skrastins
Institute of Telecommunications
Riga Technical University
Riga, Latvia
andris.skrastins@rtu.lv

***Abstract*—This paper is complimentary research to our previous paper [1] on Service Function Chaining (SFC) symmetries effect on network function (NF) such as firewall (FW) application successfulness. This studies focus is on service function (SF) usability improvements in inter-data center networks. We introduce a method of network traffic simultaneous steering through multiple SFs in SFC domain stretching over two data centers (DCs), resulting in resource sharing among both DCs. We tested such methods efficiency via conducting a series of simulations in GNS3. Results indicate an overall increase in successful available bandwidth utilization (from 0.3 to 0.65) measuring AVG throughput.**

Keywords—MPTCP, SDN, SFC, Inter-data center networking

I. Introduction

Software Defined Networks (SDN) have become a de facto standard in data center (DC) core design for at least a decade. SDN allows to menage resources a lot more efficiently in comparison to classical L2 packet switching or L3 routing. SDN holds a wide variety of functionality, but a standout solution is its packet forwarding ability that enables Service Function Chaining (SFC) usage.

SFC is a network traffic steering mechanism allowing virtualized network path customization with use of additional packet encapsulation, therefore, avoiding physical topology changes or complex routing policies. We have previously covered basics of SFC in paper [2] where we argued that SFC domain should be able to span over geographically disperse networks.

In recent years more and more information technology companies have expanded their resource capacity leveraging technological provision of multiple DCs. In networks where these DCs are placed relatively close to one another a telecommunication operator can provide direct link by means of leased line or dark fiber. Hence creating inter-data center networks.

We consider leased line connections as a great underlay for SFC domain expansion. Having a single SFC domain over multiple DCs would allow service function (SF) sharing. SFs are such network functions (NF) that by use of additional encapsulation like Network Service Header (NSH) for traffic forwarding create an overlay network topology of SFC paths. Enablement of SFC path spanning across multiple DCs allows also to not only share SFs themselves but use them simultaneously for the same network traffic.

Configuration alike would allow Multipath (TCP) MPTCP network traffic to traverse SFC domain without hiccups due to bottlenecks created by use of a single SF for both flows.

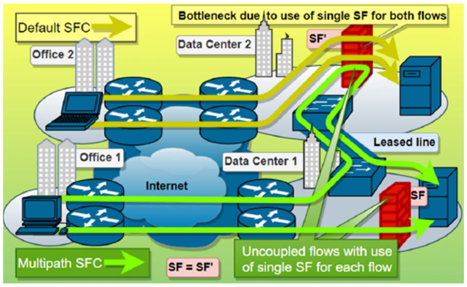
In this study we introduce a novel method of multipath capability leverage for SF sharing among multiple DCs. Results of this study show that it is possible to increase overall SFC domain throughput by balancing network traffic among multiple SFs.

II. Related wors

Inter-connected SFs in Fat-tree data centers are described in paper [3]. Authors claim to provide an algorithm for calculation of required SF placement in DCs according to needs of their tenants. This paper indicates SF placement complicity as well as incompatibility of needs from DCs tenants and DCs capabilities. Problem compliments our statement of necessity for SF share among DCs.
In paper [4] authors introduce a novel classification mechanism that leverages knowledge of inter-corelated flows therefore making more cost-sensitive solution. This paper introduces a need for measurement of different flows for proper measurements both mice and elephant.
Authors of study [5] showcase a mechanism of flow entry preservation as such mechanism is necessary to be imposed on inter-datacenter networks. Authors claim these kinds of networks are vastly used for cloud service delivery. Therefore, facilitating our claim of necessity for development of such network.
Throughput improvement necessity for inter-connected data center networks have been discussed in paper [6]. Authors state that with proper network classification it is possible to increase throughput in optical connections. An adaptive classification is proposed.

III. Joint domain working principles

Leased lines allow for multiple DCs to be joint in a single more capable data processing entity. In figure 1 an interconnected DC network is depicted. While both – sufficient network characteristics as well as computational speeds, could be present, still at a peak hours network functions can become oversaturated and therefore lower their performance and decrees overall networks capability. Fig. 1 also introduces our proposal for SF sharing in cases where such share can improve service delivery. In example office 1 has their flows redirected through adjacent SF. This helps to escape a bottleneck effect that tenants of office 2 are having.


Default SFC
Bottleneck due to use of single SF for both flows
Office 2
Data Center 2
SF'
Leased line
Office 1
Data Center 1
Internet
SF
Multipath SFC
SF = SF'
Uncoupled flows with use of single SF for each flow
Fig. 1. Interconnected data center network utilization

In this paper we replicate proposed network with use of GNS3 network simulator and multiple Open-virtual Switch (OvS) instances virtualized in Virtual Box virtual machines (VMs). Simulation network is shown in fig. 6.

A. Elements of proposed approach

Service function chaining is a mechanism that allows for network traffic redirection towards desired SFs by use of additional packet encapsulation. Fig. 2 shows difference between pre classified and accordingly encapsulated traffic forwarding and forwarding according to original packet information. With encapsulation network traffic can bypass SFs which are not necessary to be imposed on the data.

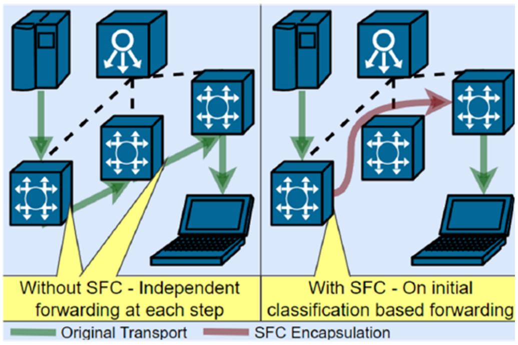

Without SFC - Independent forwarding at each step
With SFC - On initial classification based forwarding
Original Transport
SFC Encapsulation

Fig. 2. Service Function Chaining encapsulation [2]

Figure 3. depicts a comparison between standard TCP and Multi-path TCP (MPTCP). MPTCP by itself is not a standalone transport protocol. It only exists as an extension to standard TCP protocol. Therefore, data packets facilitating MPTCP can traverse network with unknowingly to network devices as from the Application layer perspective as well as Data link and Network layers have no difference between carrying standard TCP or MPTCP as even IP protocol encapsulated header field value stays equivalent to number 6.

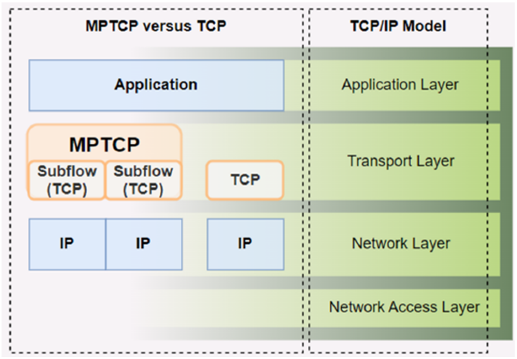


Fig. 3. Standard TCP and MPTCP comparison

B. Topology overview

Within this simulation model we tried to replicate a miniaturized and simplified network working principle of end user receiving service over wide area network (WAN). Therefore, we split the whole topology in three joint segments – LAN, Internet and SFC domain:

- LAN consists of TCP connection source.
- Internet holds multiple routers connected in mesh and running a simple routing protocol over them such as EIGRP.
- SFC domain consist of two sub-domains one for each data centers and mainly consisting of switches and TCP connection destination.

In figure 4 we have depicted described above topology where SF and SF' as well as SFF and SFF' are all run on Open-virtual Switch (OvS) switches as they allow necessary service function path (SFP) creation. Control of these switches was made by Ryu controller. Data link speeds were chosen deliberately small as to not be affected by physical hardware capabilities. Therefore, regular links are ethernet, but Highspeed links are Fast-Ethernet.

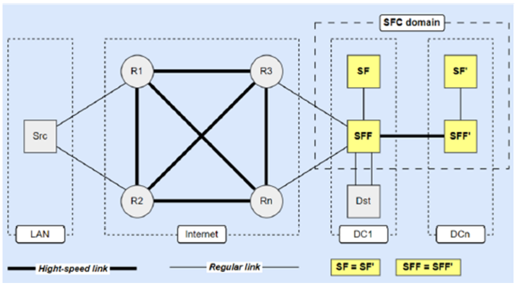


Fig. 4. Service Function Chaining domain

Within SFC-domain we distinguished two different SFPs and they are shown in figure 5. SFP1 and SFP2 both are made up so that MPTCP capability of both source and destination nodes would be preserved. It's important to notice That MPTCP can as well be facilitated even under condition where only one either source or destination has two IP routes available.

There were three case scenarios carried out during our simulations:

1. A path with no SFs – source reached destination node without data ever traversing an SF. This was used for reference of effect having even a single SF in path.
2. A path with 1 SF – source reaches destination traversing one SF that is located in the same DC as destination node.
3. A path with 2 SFs – source reaches destination by having one MPTCP path traversing SF in DC where destination node is placed but the other MPTCP path traversed SF placed in DC connected over leased line connection.

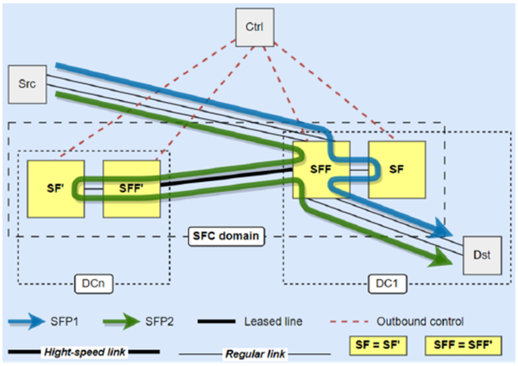


Fig. 5. Service Function Paths

IV. Network simulation

Simulation network consisted of 10 virtual machines (VMs) and 10 Cisco routers. VMs run different operating systems as they had various purposes. For example, data source and destination nodes had to have MPTCP capability and therefore they had to be compiled from older Ubuntu distributions.

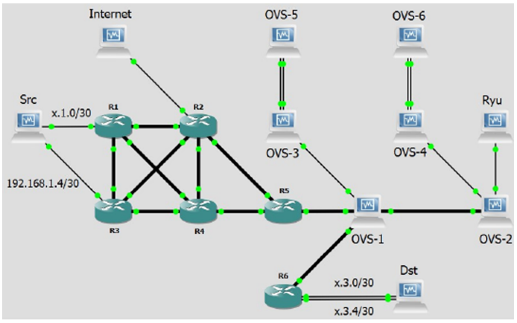
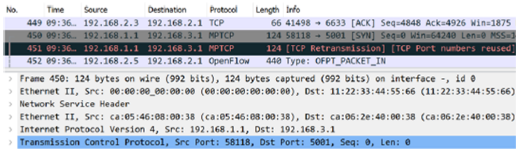


Fig. 6. GNS 3 topology

For network traffic generation we used Iperf bandwidth testing tool. Iperf has capability of creation of various length TCP connections. We used Iperf to generate three different TCP communication patterns described below:

1. Normal – TCP connection with single dialog run for 10 seconds. Command: ***iperf -c 192.168.3.1***.
2. Elephant by length – TCP connection with single dialog run for 60 seconds. Command: ***iperf -c 192.168.3.1 -t 60 -P 1***.
3. Mice by length – TCP connection with 5 dialogs run in parallel for 0.2 seconds repetitively for overall time span equivalent to 60 seconds. Command: ***for ((i=1; i<=60; i++)); do iperf -c 192.168.3.1 -P 5 -t 0.2; done***.

SFC domain utilized Network Service Header (NSH) protocol as SFC encapsulation. We created network traffic classification with use of regular OpenFlow15 capabilities such as encap(), decap(), nsh() and other actions. Therefore, simplifying encapsulation implementation. Previously [2], for adding NSH header, we have used Scapy packet crafting tool. In this case however our encapsulation imposed doubling of Ethernet header as shown in fig. 7. Where from Wireshark capture it can be distinguished that MPTCP protocol is formed from TCP options as well as NSH header is placed outwards.


```
No.   Time      Source       Destination  Protocol  Length Info
  449 09:36…  192.168.2.3  192.168.2.1  TCP          66 41498 → 6633 [ACK] Seq=4848 Ack=4926 Win=1875
  450 09:36…  192.168.1.1  192.168.3.1  MPTCP       124 58118 → 5001 [SYN] Seq=0 Win=64240 Len=0 MSS=1
  451 09:36…  192.168.1.1  192.168.3.1  MPTCP       124 [TCP Retransmission] [TCP Port numbers reused]
  452 09:36…  192.168.2.5  192.168.2.1  OpenFlow    440 Type: OFPT_PACKET_IN

> Frame 450: 124 bytes on wire (992 bits), 124 bytes captured (992 bits) on interface -, id 0
> Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 11:22:33:44:55:66 (11:22:33:44:55:66)
> Network Service Header
> Ethernet II, Src: ca:05:46:08:00:38 (ca:05:46:08:00:38), Dst: ca:06:2e:40:00:38 (ca:06:2e:40:00:38)
> Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168.3.1
> Transmission Control Protocol, Src Port: 58118, Dst Port: 5001, Seq: 0, Len: 0
```
Fig. 7. Packet structure within SFC domain

V. Results

We did various runs and experiments until we found three distinctively different scenarios for testing our proposed utilization of service functions. Once we had found parameters adequate, we run 90 independent tests – 30 for each SF count. Where each 10 tests were done under different Iperf parameters stated above. In results we also included average of measured parameters for generalization of gained data.

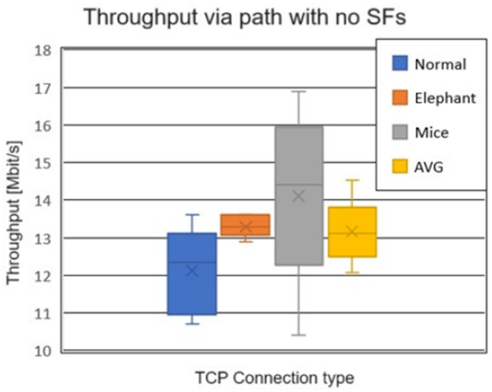


Fig. 8. Measured throughput for path with no SFs

In figure 8 throughput for TCP connections traversing SFC domain with no SFs in the path is show. This measurement was done as reference for how a single SF can become a bottleneck and lower service delivery performance. In figure 8 it can be determined that MPTCP capability of source and destination nodes are being facilitated correctly and throughput for all types of TCP connections mainly stays around 13Mbits/s.

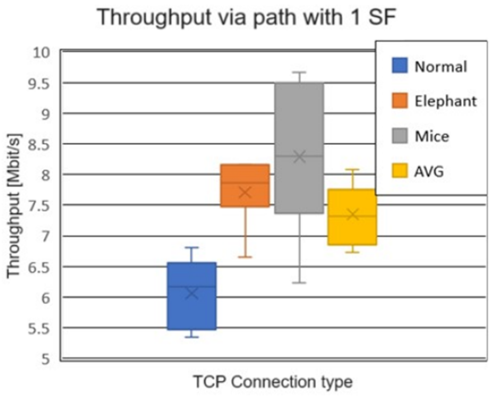


Fig. 9. Measured throughput for path with 1 SF

Figure 9 discloses previously discussed bottle effect where one SF can't deliver speeds required by communication parties. Therefore, throughput has lowered for all TCP connection types. Significant decrease has been captured for Normal connections as their mean point marked with a cross has plumed to roughly 6Mbits per second. Mice connections on the other hand show close to excellent available service utilization as its throughput reaches 9.5Mbits per seconds. Elephant connections have low variation in bandwidth utilization due to longevity of a single connection.

In correlation to reference measurements average throughput has lowered in half from 13Mbits per seconds to 7.5Mbits/s. That shows the inability for communication parties to leverage their MPTCP capability.

In figure 10 our proposed inter-data center network utilization for extended Service Function Path availability results are shown. Evidently throughput has climbed for all TCP connection types and average value is close to reference average that is 13Mbits per second.

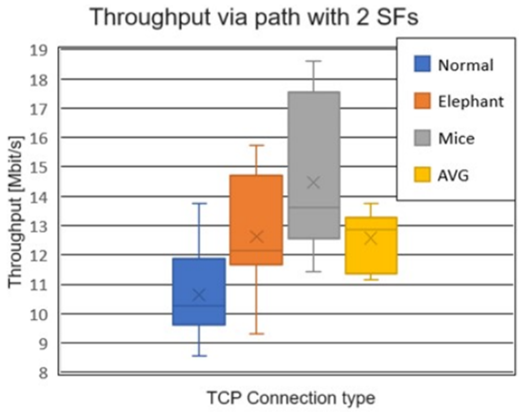


Fig. 10. Measured throughput for path with 2 SFs

Sending half of the MPTCP flows through second SF is a two-pointed sword. The available bandwidth utilization has been increased but the range of measurement variations shows even for Elephant connections. Their measured throughput now is wider then under previous two conditions. This shows sights of unsteady service delivery.

In fig. 11 a comparison between average TCP connection throughput is shown. From this comparison it is evident that spanning service function paths among multiple DCs approach has its potential as leveraging 2 SFs (green in fig. 11) is more advantageous than utilization of a single SF (yellow in fig. 11) placed in the same DC. 2 SFs clearly climb close to reference value (orange in fig. 11) around 13Mbits per second.

However average values do not show how wide is the variation of measured throughput values. This variation can significantly reduce overall service delivery and cause application experience decrease.

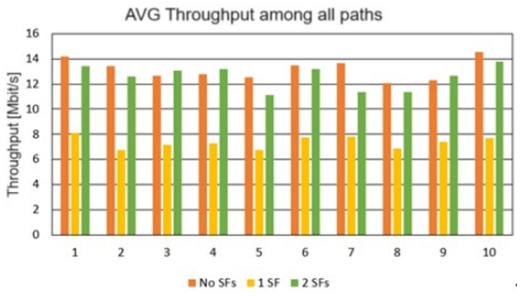


Fig. 11. AVG throughput among all paths

Although results have been gathered from simplified and miniaturized model, still through generalization we can state that approach shows potential as throughput has increased from AVG value of 0.3 to 0.65 from available bandwidth.

CONCLUSIONS

In this paper we discussed a soring issue of network service function disjoint and how network service functions themselves can become the weak point of the network creating a bottleneck effect for multipath capable communication means.

Grouping of data centers in Service Function domains allows their service functions to be used interchangeably. Such communication means as Multipath TCP can leverage this ability.

We tested our approach in simulations run in GNS3 simulator with generating different TCP connection types – Mice (short), Elephant (long) and normal. Test network consisted of both L3 (routers) and L2 (switches) network devices and communication protocol chosen was MPTCP.

Best bandwidth utilization was measured for Mice connections (0.65 to 0.9 from 1) but more stable are Elephant connections (0.6 to 0.75 from 1) as their measured values have more narrow range.

Average throughput values comparison shows that use of our approach of multiple service function utilization from two adjacent DCs increases service delivery capability.

REFERENCES

[1] M. Mihaeljans and A. Skrastins, "Evaluation of reactive service function path discovery in symmetrical environment," Telfor Journal, vol. 14, 2022

[2] M. Mihaeljans and A. Skrastins, "Network Topology-aware Service Function Chaining in Software Defined Network," 2020 28th Telecommunications Forum (TELFOR), Belgrade, Serbia, 2020

[3] G. Moualla, T. Turletti and D. Saucez, "An Availability-Aware SFC Placement Algorithm for Fat-Tree Data Centers," 2018 IEEE 7th International Conference on Cloud Networking (CloudNet), Tokyo, Japan, 2018

[4] M. A. S. Saber, M. Ghorbani, A. Bayati, K. -K. Nguyen and M. Cheriet, "Online Data Center Traffic Classification Based on Inter-Flow Correlations," in IEEE Access, 2020

[5] Yao-Chun Wang, Ying-Dar Lin and Guey-Yun Chang, "SDN-based dynamic multipath forwarding for inter-data center networking," 2017 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN), Osaka, 2017

[6] H. Rastegarfar et al., "TCP flow classification and bandwidth aggregation in optically interconnected data center networks," in Journal of Optical Communications and Networking, Oct. 2016

[7] TCP Extensions for Multipath Operation with Multiple Addresses - RFC 8684, available online at - https://datatracker.ietf.org/doc/rfc8684/

Appendix 7

M. Mihaeljans, A. Skrastins and J. Porins
Parmounts of Intent-Based Networking
28th International Conference ELECTRONICS 2024
Palanga, Lithuania, September, 2024

Paramounts of Intent-Based Networking: Overview

Martins Mihaeljans*, Andris Skrastins, Jurgis Porins
Institute of Photonics, Electronics and Telecommunications, Riga Technical University,
10 Zunda Embankment, LV-1048, Riga, Latvia
martins.mihaeljans@edu.rtu.lv; andris.skrastins@rtu.lv; jurgis.porins@rtu.lv*

***Abstract*—This study is an exploration of state-of-the-art Intent-Based networking (IBN) model's design. In IBN communication means are initialized by user's (herein IT staff not end-user) input of requirements and not instructions. Thus, allowing for network's self-organizational abilities to setup communication paths. Form research of academic studies and standardization drafts we conduct IBN structure shown in fig. 1. We determined a necessity for change in the design. Current IBN model detains its adaptation as network assurance requirements of ensuring network security and scalability, and continuity are unfulfillable via conduct of network analysis and track of intent drift. We propose two sub-models - one for autonomous networks and one for supervised networks.**

***Index Terms*— Intent-based networking; Internet of things; Network function virtualization; Software defined networking; Service function chaining.**

I. Introduction

"None of us just live in a silo. Everything is in context," U.S. VP K. Harris states and emphasizes, "You exist in the context of all in which you live and what came before you" [1]. This is pertinent to Intent-Based Networking (IBN), where subject must integrate into and preserve the surrounding structure. In essence, artificial intelligence (AI) serves IBN by augmenting the network control and management processes previously overseen only by man.

So, whilst 5G communication enablement has been in a forefront of recent advancements in telecommunications field slowly but steadily Intent-Based Networking (IBN) gains a broader recognition both in academic and industry.

IBN users not necessary hold knowledge of network's structure. But they can express desired intentions as requests. Interpreter transforms request into configuration. An actuator then enables the newly generated configuration. Finally, assurance ensures intents fulfilment. IBN heavily relies on artificial intelligence and machine learning (ML) capabilities. From profiling of intent via Natural Language processing (NLP) to predicting network accessibility via big data and Generative Adverbial Networks (GAN). [2,3,4,5,6]

In this paper we investigate the basic building blocks of IBN and its working principle. Outcome of this research is a proposal for division of IBN. The need is to differentiate autonomous networks from supervised networks. Network assurance is a stumbling stone for IBN adaptation. [6]

TABLE I. SCOPE OF THE OVERVIEW.

Material type	Count	Reference index
Academic studies	20	8-11,13,15-18,20-30
Recommendations	4	2,3,4,12
Surveys	5	5,6,7,14,19

Studie's background is sorted in Table I. The criteria for selection were accuracy, relevance, and scope of information.

II. Structure of Intent-Based Networking

IBN has five building blocks (1. Profiling, 2. Translation, 3. Resolution, 4. Activation, 5. Assurance). Blocks form two closed loops (combination of all five and combination assurance-resolution-activation) shown in fig. 1.

IBN is supposedly set to achieve a trinity of assessment - self-organization, self-reconfiguration, and self-optimization. One of the loops facilitates whole lifespan of intent while the other provides assurance (a perpetual enhancement process of intent's assessment via analyses of network state). [5]

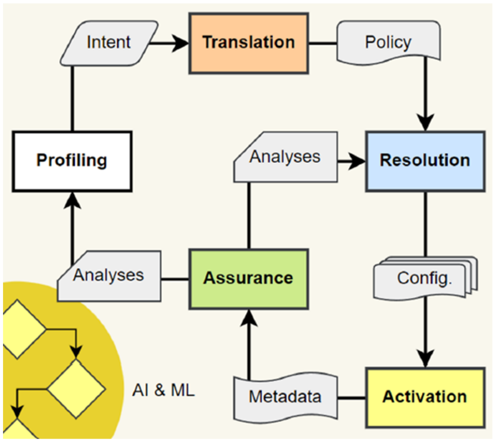


Fig. 1. Intent-Based networking model

The survey [5] has also meticulously examined applicable underlay technologies and extracted a set of IBN's open issues in conjunction with possible solutions which are:

1. The alignment of intents expression with translation could be resolved by classifying user's experience.
2. Business specific intents can be fulfilled if IBN is auto-adaptable and auto-configurable.
3. Improve of neural language processing vocabulary via use of chatbots and incremental learning for IBN to adopt user's language and not vice versa.
4. Network assurance and available datasets could leverage from combination of ML profiling and transfer learning method.
5. Zero-touch aka autonomous networks can only be achieved via use of AI in control theory application.
6. Multi-domain network management is obtainable by structuralizing network controllers in a hierarchy.

Given problem-solution statement list is non-exhaustive and at hand states the general areas of interest for IBN model's deployment.

A. Profiling of user's request/desire

Often an intent is not something that is directly feasible from an amount of information presented to interpreting system. [7] Users of IBN can be with, with some or even without any knowledge of actual network structure and capabilities. Therefore, users proposed intents would require undergoing an assessment and correction mechanisms. These mechanisms might be incorporated in profiling techniques shown in fig. 2.

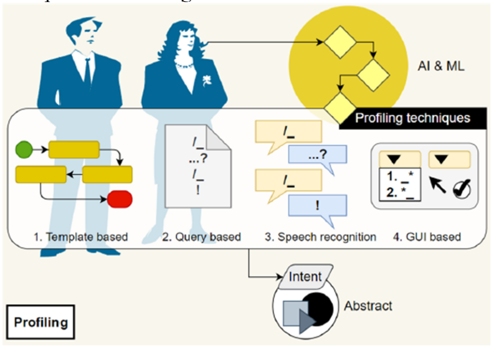


Fig. 2. Intent profiling building block

1. Template based – user is taken through set of predefined attributes and preferences to setup a necessary intent.
2. Query based – users' intent is discovered via initial input and acknowledgement cycle where additional information is requested via secondary input.
3. Speech recognition – users' intent is discovered via keyword extraction from natural language speech or text-based input with help of Natural Language processing (NLP). [8]
4. GUI based – user is presented with dashboard of network pseud-configuration settings.

Profiling techniques in use depend not as much on users' preferences as on their ability to understand technicalities. Some examples of users and their intents are listed below:

- Beginner (helpdesk operator) – Alert on connection issues at branch offices.
- Competent (service technician) – Optimize link utilization for High-Quality voice communication needs in headquarters for weekdays nine-to-five.
- Advanced (project manager) – Enforce government regulation compliance on all logged and backed-up data for past quarter.
- Proficient (software developer) – Secure transactions and system calls made to and from test environments.
- Expert (network administrator) – Prioritize business critical application workflow continuity for minimal network traffic flow migration during peak hours.

As shown in the examples above intent itself is an abstraction and not a set of performable actions. In addition, more experienced users might be granted with privileges of supervision as well as abilities to overthrow other user intents.

B. Translating intent into underly appropriate policy

Application of an intent cannot begin without fitting the task to sub attributes of event, action, condition (EAC). EAC is a basic declaration of a rule in policy-based management systems. Such systems have been in action to menage network traffic in network functions like firewalls for more than a decade. [2]

While intent is a high-level abstract descriptor a policy is low-level descriptor due its nature of need for structure and configuration. Intent translation is shown in fig. 3.

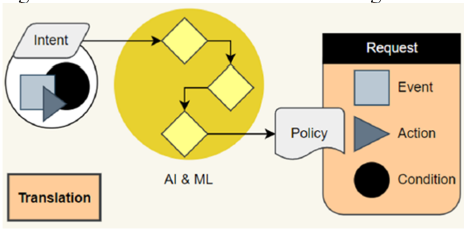


Fig. 3. Intent translation building block

For translation different mechanisms can be utilized like blueprint, mapping, refining, graph-based, ML etc. Whichever mechanism is in use the outcome must form an appliable network configuration and must hold no abstraction. A standardization draft [4] defines an IBN language specifically for intent modelling. A study [9] extends it with ability to define policy definitions reactively.

C. Resolution of intents footprint

When intent is descriptive enough to be accommodated into the underly system a mechanism of try/catch must be run for disclosure of the effect its application might generate. Touched systems must persist serving all previously enacted intents if an overwrite is not the purpose of applicable intent.

A prediction of possible intent drift is possible with combination of intent database and previous network state snapshots. These predictions could be ML generated or could also be clause based logical decisions.

Network slice load prediction and resource forecast as well as load balancing and anomaly detection, or user mobility prediction are just some examples of IBN tasks performed by AI and ML. [10]

In case of resolution fail IBN can notify user and restart the profiling process. Although IBN is intended to be an autonomous, close to a one-touch environment that does not mean it incorporates one-shot policy. Users are required to input their intents once and the rest of intents lifecycle is facilitated by IBN. However, users are required to oversee intents creation process as corrective actions may be needed. Intent resolution is shown in figure 4.

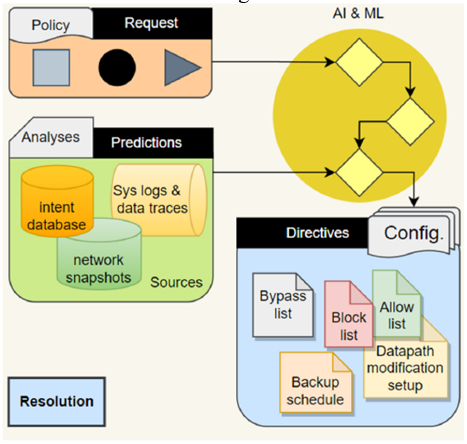


Fig. 4. Intent resolution building block

D. Actuating the necessary modifications

In a state of compliance with the existing system intent can be deployed. This IBN building block is where actual underly network gets affected. Changes to existing network fabric can include but are not limited to network traffic flow migration, Quality of Service (QoS) assessment, Datapath modification or even server migration to other premises.

Service function chaining (SFC) and network function virtualization (NFV) are main underlay technologies leveraged for accomplishment of intent deployment. Still, no changes are error-prone therefore upon making any modifications a snapshot of network configuration as well as rest of the intent's states are required to be taken. This can serve for both immediate rollback and as a reference point for intent drift remedy upon subsequent discoveries.

Intent actuation building block is shown in fig. 5. Where configuration from resolution block is inserted into network fabric and metadata of network state is collected afterwards.

It's important to notice that network fabric can rely on both IBN orchestration and local orchestration from within placed network controllers.

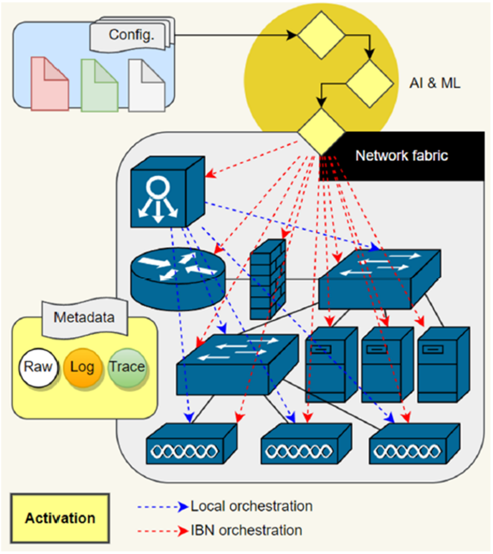


Fig. 5. Intent activation building block

An essential component of IBN is Single Source of Truth. This component handles intent's informational structure and ensures that all related parties (user, network controllers, and network fabric) has the same understanding of intent and its application goals and current state.

E. Assurance of operational successfulness

A continuous system monitoring enables detection of intents state and whether original user's request is fulfilled. Intent fulfilment is not a stationary state therefore upon successful deployment assurance keeps track of necessary optimizations and adjustments.

Assurance is the key element in IBN which enables functionality of rest of the building blocks. Its prime objective is passive and active system monitoring. Monitoring types can variate between different parts of network fabric in use for each intent's deployment. Followings are some of examples:

- On access network – end user mobility and density, interference, and signal to noise ratio (SNR).
- In data centres (DCs) – CPU, RAM, and storage utilization, Input/Output system stats, virtual machine (VM) states.
- For optical networks – quality of transmission, optical power, optical SNR.
- Generic values – round trip time, jitter, packet delivery/error ratio or queue size. [6]

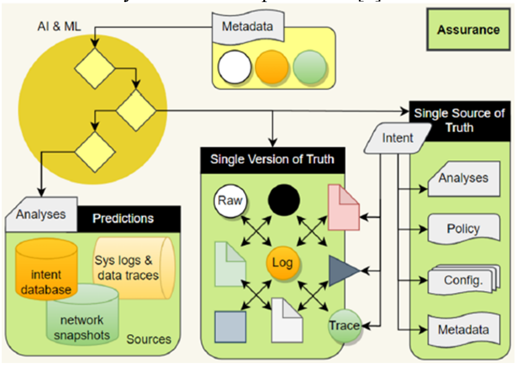


Fig. 6. Intent assurance building block

Intent assurance building block shown in fig. 6 also ensures Single Version of Truth which is a component used for database querying, information pulling, and data filtering for correlations related to single intent within multiple parts of network fabric.

Another important aspect for maintaining abstraction at a profiling building block is security as providing users with network transparency raise potential risks of atrocities. [11]

III. Underlay technologies

IBN does not target to redefine existing networking principles of switching, routing, and forwarding or any other packet processing mechanisms. It also does not intend to reinvent operational, administration and management (OAM) application techniques. Its primordial objective is to take control over all these functions whilst leaving capability of higher-level supervision to its users. Therefore, an underlay technology used for intent structurization can be but is not limited to policy-based network management (PBNM). Even more so, IBN potentially would utilize same existing mechanisms as policy decision point (PDP) and policy extraction point (PEP) for intent instantiation.

A. Artificial intelligence

As reported in [12] Artificial intelligence (AI) does have a set of problems that needs to be overcome for its application in network management. A non-exhaustive list is as follows:

1. Huge solution space – try-catch of every possible aid is impractical.
2. Unpredictability – a continuous change in network state makes forecasts unreliable.
3. Demand window – solutions must arrive timely to be made use of for problem management.
4. Dataset dependence – solutions are as accurate as is ML training sets and if they hold applicable cases.
5. Integration with existing management systems.

6. Cost-efficiency – when distributed AI is used in edge or at the forwarding plane processing resources are limited.

As a subset of AI, machine learning (ML) and big data are a cornerstone of intent-based networking development. Together these technologies are utilized in all five building blocks of the model accordingly:

- Profiling – Natural Language processing (NLP) allows for human speech recognition is how interpreter could function much alike voice assistant.
- Translation – Template-based fitting translation as well as keyword and phrase structure extraction for policy generation.
- Resolution – Comparison of multiple datasets of monitoring data in conjunction with user's intent and network configuration database check.
- Actuating – Resource management and allocation trade-off operation automatization and more complicated task handling as network traffic flow migration planning and execution.
- Assurance – Continuous monitoring optimization according to network availability fluctuation and autonomous state snapshot generation for future intent lifespan analysis including but not limited to intent drift extraction.

Authors of [13] used ML models of Long Short-term Memory (LSTM) and Convolutional Neural Network (CNN) for prediction of CPU usage in virtual machines (VM) for near future timeframe. It's also proposed that assurance engine design may variate for different use case scenarios.

Some areas are less mature than others. For example, learning of network states and predicting possible hiccups is limited by quantity of qualitative data sets gathered from network troubleshooting. In aid a generative adverbial network (GAN) can help generate missing cases. [6]

B. Software Defined Networking

Unlike legacy network devices in which communication configuration capabilities are directly imprinted and burnt into local memory (switches, routers, firewalls etc.) software defined networking (SDN) devices facilitate centralized management and configuration approach. SDN is one of the crucial technologies under IBN concept as it is organized in three easily distinguishable planes as follows:

- Application plane – Accommodates users (usually network administrator or similar) interaction with the network through policy-based network management (PBNM) which is imposed on underlay control plane via application programming interface (API) or RESTful functions.
- Control plane – Receives users declaratives and transcribes them into network configuration which is broadcasted to all implicit forwarding devices that resides in underlay forwarding plane.
- Forwarding/data plane – Executes network traffic forwarding according to configuration imposed by control plane. Unlike L2 switching or L3 routing, forwarding is packet processing mechanism that can extract any packet header information to match network traffic with specific flow entry for distinguishing the right packet transmission path.

SDN design model is shown in fig. 7. where operational, administration and management (OAM) channel directs network policy implementation while data channel showcases end device communication. Featured in our recent research on IoT and SDN fusion [14] this SDN model represents the very own logic of indirect device configuration as user (resides in application plane) only directly accesses network controllers. Study [15] notes that SDN by itself does lacks data analytics (herein assurance-resolution-activation) closed loop that is needed for autonomous networks and is a part of IBN. Research [15] states that partial IBN implementation is due to challenges that a comprehensive framework faces - how to align intent with user's requirements, how to translate intent into network configuration, how to ensure intents fulfilment.

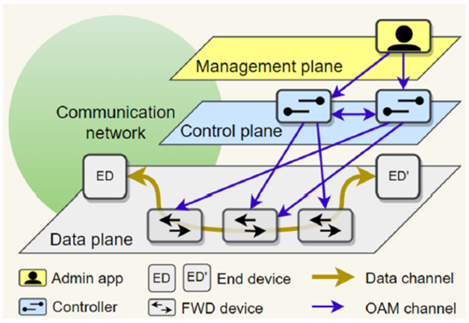


Fig. 7. SDN design model [14]

It is common to link SDN with only data centre (DC) networks as most of the backhaul technologies relies on legacy network capabilities (L2 switching at network access and L3 BGP at core backbone).

Due to rapid industrialization enterprise has spawned its industries into multi-domain tenants. For example, multiple companies can share same data centre, rack or even server but at the same time have a secondary placement in their headquarters and/or branch offices. GUI based multi-domain IBN model for virtual NF scaling for Evolved Packet core (EPC) has been introduced in [16].

SD-WAN solutions such as Cisco Meraki (centralized network access device management) or Quagga routing suite (SDN edge software for legacy network protocol control from Linux OS) allows to limit the gap between multi-tenant needs and core network capabilities. For example, in [17] authors studied how the number of SD-WAN branches affects intent deployment time.

Although for many years OpenFlow protocol and its eligible hardware/software like Open Virtual Switches (OVS) were an industries standard for an SDN implementation, nowadays it is being replaced by more robust approach – Programming Protocol-Independent Packet Processors (P4).

OpenFlow's limitation lies in its inability to redefine device's function set (match, set IP source field, push VLAN, pop VLAN etc.) in a get-go. Users still needed to wait on manufacturers to implement necessary functionality via update or upgrade. P4 however allows not only to manipulate with existing functionality but rebuild it or add new one to white switches (networking devices with possibility of reconfiguration) [18].

A plethora of scientific research regarding SDN has been done via use of Ryu controller and Mininet network emulator. Authors of [19] created an IBN framework for AI based intent perception and fulfilment. By conducting a survey authors [19] discovered that AI commonly is used either for intent input or management, but not comprehensively throughout the IBN structure.

C. Network function virtualization

DC networks can become oversaturated with inbound and outbound network traffic not only at a peek hour but also due to unexpected application behaviour or improper data backup generation schedules.

Some data traffic is supposed to leave DC premises but other only needs to travel as far as neighbouring server appliance or even adjacent blade of same server.

There are instances where various applications reside in same server (Kubernetes, Docker, OpenShift etc.) but requires distinct barrier between for security reasons.

In all mentioned cases a virtualized network traffic processing device can come in aid as a trade-off between need for packet switching speed or placement and functionality among closely resided resources.

Any network function (NF) can be virtualized – routing, switching, forwarding, network address translation (NAT), deep packet inspection (DPI) etc. In research [20] virtualized NF placement is studied for an abstract user intent satisfaction in multi-tenant environment.

D. Service Function Chaining

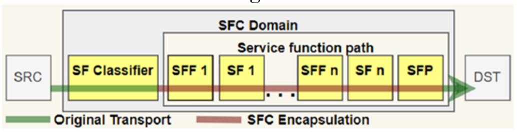


Fig. 8. Service function chaining domain [21]

Due to development of network function virtualization (NFV) a paradigm of steering network traffic through customized path among selection of NFs has emerged in form of Service Function Chaining (SFC).

We have extensively covered SFC in study [21] but in brief - SFC domain shown in fig. 8. consists of following elements:

- SFC network traffic encapsulation-aware network functions (NFs) aka service functions (SFs), SF classifiers and SF forwarders.
- A SF chain is the designated array of SFs that network traffic should traverse for service delivery, but an SF path is actual Datapath taken including SF forwarders and classifiers.
- SF forwarders are nodes that do not modify network traffic any how but only passes the network traffic from one SF to another. SF classifiers impose and remove SFC encapsulation on to network traffic packets.
- Utilization of additional encapsulation allows to alter network traffic path from one that would be taken by L2 switching or L3 routing.

We also covered service function chaining in [22] from perspective of automation for network traffic classification at SFC domain ingress. We proposed reactive-path discovery at a time of traffic arrival instead of proactively defining all possible paths. Main drawback of our method was time needed for appropriate service function selection. AI potentially could mitigate it by generation of sophisticated selection patterns.

In our recent study [23] we used SFC encapsulation abilities to differentiate between multipath TCP sub-flows belonging to same session for bandwidth utilization optimization. Similarly, study [24] approaches necessity for fair resource allocation and tackles with conflict detection among intents generated by multiple users. Study [24] claims that IBN use in SDN rises also a security issue where abstraction of centralized network control enables potentially illegal access.

IV. Implementation Domains

As a multipurpose system IBN can be deployed in a variety of domains and serve not only for communication means but also businesses, healthcare, social security etc. Listed below are some of the domains of interest:

A. Enterprise networks

- Operational technology – Management of intents related to manufacturing, monitoring and control systems in industrial environment.
- Information technology – Management of intents related to LAN, DCs, and Cloud resided end device end-to-end communication enablement.

Research [25] delves into Quality-of-Service delivery capabilities of IBN as it eludes on transition away from traditional networking in enterprise environments.

B. Operator's networks

5G communication autonomy evolvement for ML used in radio access networks (RAN) where importance of a proper antenna tilt is just as high as one for adequate network slicing.

Authors of [26] developed IBN compliant monitoring framework capable for both active and passive RAN monitoring by utilization of network function virtualization (NFV). Tools utilized for data gathering and querying were Prometheus, Elasticsearch, and Grafana.

C. Internet of Things

Vast adaptation of Internet of Things (IoT) technologies in monitoring and control systems like sensor and actuator networks has created a need for SD-WAN solutions for automation and maintenance of these autonomous devices.

Recently IoT conquered another milestone by overcoming a lack of interoperability in consumer products. Matter protocol and Thread communication technology have made smart home IoT devices attractive to new buyers as they no longer must worry whether one device will work with another. We studied these technologies in [14] where we discovered that inadequate ambiguity of IoT standards halts the conceptional model from ever gaining a common framework for IoT device design.

Research [12] states that the lack of human-understandable reasoning process is an aspect that makes AI and ML solutions unattractable for implementers in networking domains. Often guarantees in form of service level agreements (SLA) must be met. Therefore black-box behaviour has no place in assuring a fulfilment of SLA.

V. Proposed IBN Sub-Models

The main block of IBN model is assurance. It is responsible for intent fulfilment by ensuring the security and scalability, and continuity of the network via conduct of network analysis and track of intent drift. The open challenges for assurance are as follows:

1. Bilateral user intent intervention – how to ensure that one user's intent does not negatively affect other users' intent.
2. Resource allocation – how to ensure a fair use and proper distribution of network slices during normal usage, peak load and disaster recovery scenarios.
3. Intent supervision – how to ensure intent fulfilment and escape already existing intent drift whilst accommodating new intents.

To aid in satisfaction of these requirements we propose to split IBN model in two separate models. One of which could be automation-oriented removing multiple user and new intent requirements. The other could be suggestion-oriented leaving network configuration at the user's hands and having assurance for analysis gathering. Therefore, removing resource allocation requirement.

A. IBN for autonomous networks

IoT alike networks does not necessary require human intervention. For example, Thread network technology has self-organization and self-reconfiguration capabilities. But on topology changes Thread networks tend to suffer link breakages due to personal area network regeneration process. The issue had been encountered in our previous research [27] where we studied Thread network technology in a simulation environment. Respectively, intent-based networking (IBN) could assist in stray node pickup.

Other domains like ad hoc or vehicular networks could potentially benefit from IBN capabilities of cloud and edge computing orchestration. Fig. 9. depicts IBN for autonomous networks. We suspect that many IoT solutions targeting smart home equipment will likely be Matter enabled as this newly industry adapted standard is set to unify the segregated consumer-product market. Matter is supposedly aimed towards ease of product development cycle with its core function of cross-platform application layer. This application layer might include intent resolution building block functionality. Where lower layers could reside in subsequent blocks. Leaving only the initial profiling up to developer.

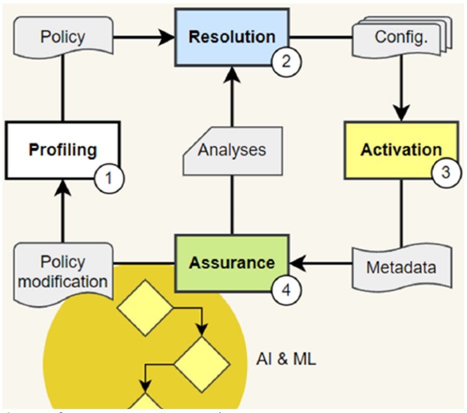


Fig. 9. IBN for autonomous networks

From profiling building block an initial policy (generated by the manufacturer or user) enters resolution block from where second IBN loop is ignited. Once assurance detects a modification necessity it generates policy modification request. Intents would still be an essential part of this model, but they would reside only in assurance building block for intent drift discovery.

B. IBN for supervised networks

Operator's networks most often span across at least one autonomous system (AS) whether it's an internet service provider (ISP) or mobile service provider. AS are supposed to interact with one another requiring supervision as a strict part of management. IBN enables a loosely-coupled inter-working between applications in various autonomous systems whilst leaving direct technical supervision to operators. [28]

A domain accommodating networks of a not so colossal scale but of a high importance is healthcare. Telemedicine and operational medical equipment as well as life support systems could make use of IBN's capabilities. [29]

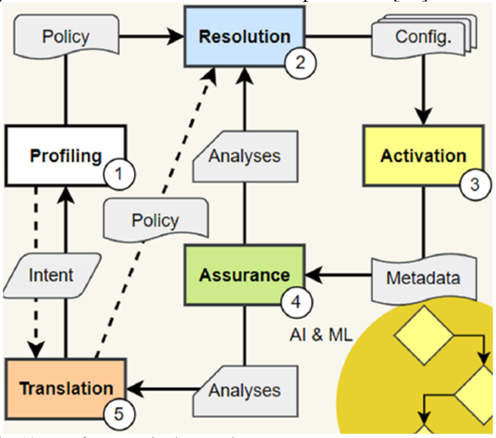


Fig. 10. IBN for supervised networks

Fig. 10. depicts IBN for supervised networks. In this model users setup network policy and from assurance receive intent suggestions for network optimization. However, an ability to ingest an intent also stays possible as a secondary option for non-expert level user.

VI. Conclusion

In this paper we studied Intent-based network. IBN is a networking control model where governance, optimization and maintenance of communication network is achieved with artificial intelligence (AI) and machine learning assistance.

Intent is a user's expressed request of necessary outcome in an abstractive manner without formulating how the outcome should be achieved. Intents ingestion, realization and optimization is a human to artificial intelligence dialog.

To our knowledge, we outlined IBN structure based on two-loop intent lifecycle (fig. 1.) making this paper to be the first refinement of IBN's graphical illustration since early Cisco Press materials.

We conduct that IBN has a significant reliance on artificial intelligence and machine learning to an extent that fullfeatured IBN implementation is unfeasible due to lack or insufficiency of network knowledge base for autonomous systems.

The complexity of end-user services and network configuration makes it hard to predict and detect the effect of intents ingestion. To aid the matter necessary steps for problem mitigation are attainable only through sophisticated systems like digital twin of physical network. [30]

We discovered that a partial implementation of IBN is common among studied materials. Studies suggest that AI is used for either input of the intent or network management, but not both. The design of IBN architecture is based on network requirements, expected outcomes, and users' priorities. [3]

We propose a transitional solution applicable up until maturity of AI and ML is raised to adequacy in complex task execution. A separate IBN sub-model implementation for autonomous networks (fig. 9.) and supervised networks (fig. 10.). This division could relax assurance building block challenges such as intent drift tracing, intent conflict prediction, network state and resource utilization forecasting.

Our proposal also touches on bridging the gap between feature-enrichment equality between both types of networks. Where nowadays it is supervised networks that has a Swiss-knife of toolset in comparison to autonomous networks.

REFERENCES

[1] K. Harris, "Remarks at White House event," Vice President of the United States, May 10, 2023, pp. 48-49, available at: https://sites.ed.gov/hispanicinitiative/files/2023/09/2023.5.10-FINAL-Commission-Transcript.pdf, last seen Aug. 10, 2024.

[2] A. Clemm, L. Ciavaglia, and L. Z. Granville, "Intent-Based Networking - Concepts and Definitions," IRTF RFC 9315, 2022, DOI:10.17487/RFC9315

[3] C. Li et al., " Intent Classification draft-li-nmrg-intent-classification-00," IETF Network Working Group, 2019

[4] S. Hares, "Intent-Based Nemo Overview draft-hares-ibnemooverview-01," IETF Internet-Draft, 2016

[5] A. Leivadeas and M. Falkner, "A Survey on Intent-Based Networking," in IEEE Communications Surveys & Tutorials, vol. 25, no. 1, pp. 625-655, Firstquarter 2023, doi: 10.1109/COMST.2022.3215919.

[6] A. Leivadeas and M. Falkner, "Autonomous Network Assurance in Intent Based Networking: Vision and Challenges," 2023 32nd International Conference on Computer Communications and Networks (ICCCN), Honolulu, HI, USA, 2023, pp. 1-10, doi: 10.1109/ICCCN58024.2023.10230112.

[7] B. E. Ujcich, A. Bates and W. H. Sanders, "Provenance for Intent-Based Networking," 2020 6th IEEE Conference on Network Softwarization (NetSoft), Ghent, Belgium, 2020, pp. 195-199, doi: 10.1109/NetSoft48620.2020.9165519.

[8] Y. Xiao, W. Quan, H. Zhou, M. Liu and K. Liu, "Lightweight Natural Language Driven Intent Translation Mechanism for Intent Based Networking," 2022 7th International Conference on Computer and Communication Systems (ICCCS), Wuhan, China, 2022, pp. 46-51, doi: 10.1109/ICCCS55155.2022.9845995.

[9] Y. Tsuzaki and Y. Okabe, "Reactive configuration updating for Intent-Based Networking," 2017 International Conference on Information Networking (ICOIN), Da Nang, Vietnam, 2017, pp. 97-102, doi: 10.1109/ICOIN.2017.7899484.

[10] K. Abbas, T. A. Khan, M. Afaq and W. -C. Song, "Ensemble Learningbased Network Data Analytics for Network Slice Orchestration and Management: An Intent-Based Networking Mechanism," NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, 2022, pp. 1-5, doi: 10.1109/NOMS54207.2022.9789706.

[11] A. -R. Meijer, L. Boldrini, R. Koning and P. Grosso, "User-driven Path Control through Intent-Based Networking," 2022 IEEE/ACM International Workshop on Innovating the Network for Data-Intensive Science (INDIS), Dallas, TX, USA, 2022, pp. 9-19, doi: 10.1109/INDIS56561.2022.00007.

[12] J. François, A. Clemm, D. Papadimitriou, S. Fernandes, and S. Schneider, "Research Challenges in Coupling Artificial Intelligence and Network Management," Internet Research Task Force (IRTF), Internet-Draft draft-irtf-nmrg-ai-challenges-03, Mar. 2024. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-irtf-nmrg-aichallenges-03

[13] X. Zheng and A. Leivadeas, "Network Assurance in Intent-Based Networking Data Centers with Machine Learning Techniques," 2021 17th International Conference on Network and Service Management (CNSM), Izmir, Turkey, 2021, pp. 14-20, doi: 10.23919/CNSM52442.2021.9615580.

[14] M. Mihaeljans and A. Skrastins, "IoT concept and SDN fusion in consumer products: Overview," 2023 3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), Tenerife, Canary Islands, Spain, 2023, pp. 1-6, doi: 10.1109/ICECCME57830.2023.10252518.

[15] H. Yu, H. Rahimi, C. Janz, D. Wang, C. Yang and Y. Zhao, "A Comprehensive Framework for Intent-Based Networking, Standards-Based and Open-Source," NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium, Miami, FL, USA, 2023, pp. 1-6, doi: 10.1109/NOMS56928.2023.10154454.

[16] T. Ahmed Khan, K. Abbas, J. J. Diaz Rivera, A. Muhammad and W. -c. Song, "Applying RouteNet and LSTM to Achieve Network Automation: An Intent-based Networking Approach," 2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS), Tainan, Taiwan, 2021, pp. 254-257, doi: 10.23919/APNOMS52696.2021.9562499.

[17] R. Perez, A. Zabala and A. Banchs, "Alviu: An Intent-Based SD-WAN Orchestrator of Network Slices for Enterprise Networks," 2021 IEEE 7th International Conference on Network Softwarization (NetSoft), Tokyo, Japan, 2021, pp. 211-215, doi: 10.1109/NetSoft51509.2021.9492534.

[18] M. Riftadi and F. Kuipers, "P4I/O: Intent-Based Networking with P4," 2019 IEEE Conference on Network Softwarization (NetSoft), Paris, France, 2019

[19] J. Huang, C. Yang, S. Kou and Y. Song, "A Brief Survey and Implementation on AI for Intent-Driven Network," 2022 27th Asia Pacific Conference on Communications (APCC), Jeju Island, Korea, Republic of, 2022, pp. 413-418, doi: 10.1109/APCC55198.2022.9943612.

[20] A. Leivadeas and M. Falkner, "VNF Placement Problem: A Multi-Tenant Intent-Based Networking Approach," 2021 24th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), Paris, France, 2021, pp. 143-150, doi: 10.1109/ICIN51074.2021.9385553.

[21] M. Mihaeljans and A. Skrastins, "Network Topology-aware Service Function Chaining in Software Defined Network," 2020 28th Telecommunications Forum (TELFOR), Belgrade, Serbia, 2020, pp. 1-4, doi: 10.1109/TELFOR51502.2020.9306554.

[22] M. Mihaeljans and A. Skrastins, "Reactive Service Function Path Discovery Approach in Software Defined Network," 2021 29th Telecommunications Forum (TELFOR), Belgrade, Serbia, 2021, pp. 1-4, doi: 10.1109/TELFOR52709.2021.9653356.

[23] M. Mihaeljans and A. Skrastins, "Efficient Multipath Service Function Chaining in Inter-Data Center Networks," 2023 31st Telecommunications Forum (TELFOR), Belgrade, Serbia, 2023, pp. 1-4, doi: 10.1109/TELFOR59449.2023.10372615.

[24] J. Zhang et al., "A Conflict Resolution Scheme in Intent-Driven Network," 2021 IEEE/CIC International Conference on Communications in China (ICCC), Xiamen, China, 2021, pp. 23-28, doi: 10.1109/ICCC52777.2021.9580197.

[25] M. Beshley, A. Pryslupskyi, O. Panchenko and H. Beshley, "SDN/Cloud Solutions for Intent-Based Networking," 2019 3rd International Conference on Advanced Information and Communications Technologies (AICT), Lviv, Ukraine, 2019, pp. 22-25, doi: 10.1109/AIACT.2019.8847731.

[26] C. Fernández, A. Cárdenas, S. Giménez, J. Uriol, M. Serón and C. Giraldo-Rodríguez, "Application of Multi-Pronged Monitoring and Intent-Based Networking to Verticals in Self-Organising Networks," 2022 5th International Conference on Advanced Communication Technologies and Networking (CommNet), Marrakech, Morocco, 2022, pp. 1-10, doi: 10.1109/CommNet56067.2022.9993854.

[27] M. Mihaeljans and A. Skrastins, "Openthread Network Density Evaluation: Quantitative Analysis," 2023 Symposium on Internet of Things (SIoT), São Paulo, Brazil, 2023, pp. 1-5, doi: 10.1109/SIoT60039.2023.10390236.

[28] R. Caldelli, P. Castoldi, M. Gharbaoui, B. Martini, M. Matarazzo and F. Sciarrone, "On helping users in writing network slice intents through NLP and User Profiling," 2023 IEEE 9th International Conference on Network Softwarization (NetSoft), Madrid, Spain, 2023, pp. 545-550, doi: 10.1109/NetSoft57336.2023.10175400.

[29] Y. Njah, A. Leivadeas, J. Violos and M. Falkner, "Toward Intent-Based Network Automation for Smart Environments: A Healthcare 4.0 Use Case," in IEEE Access, vol. 11, pp. 136565-136576, 2023, doi: 10.1109/ACCESS.2023.3338189.

[30] M. Gharbaoui, B. Martini and P. Castoldi, "Intent-Based Networking: Current Advances, Open Challenges, and Future Directions," 2023 23rd International Conference on Transparent Optical Networks (ICTON), Bucharest, Romania, 2023, pp. 1-5, doi: 10.1109/ICTON59386.2023.10207407.

Appendix 8

M. Mihaeljans, A. Skrastins and J. Porins
Service Function Chaining via SR-MPLS over P4: A rudimentary analysis
The International Conference on Intelligent Computing, Communication, Networking and Services (ICCNS2024)
Dubrovnik, Croatia, September, 2024

Service Function Chaining via SR-MPLS over P4: A rudimentary analysis

Martins Mihaeljans
Institute of Photonics, Electronics and Telecommunications
Riga Technical University
Riga, Latvia
martins.mihaeljans@edu.rtu.lv

Andris Skrastins
Institute of Photonics, Electronics and Telecommunications
Riga Technical University
Riga, Latvia
andris.skrastins@rtu.lv

Jurgis Porins
Institute of Photonics, Electronics and Telecommunications
Riga Technical University
Riga, Latvia
jurgis.porins@rtu.lv

***Abstract*—This study is a rudimentary analysis of segment routing with MPLS (SR-MPLS) use case for service function chaining (SFC) on protocol-independent switch architecture (PISA). Segment routing is a packet forwarding method that leverages both source routing and packet encapsulation. SFC is packet steering through ordered service function (SF) path technique where SFs are firewall, deep packet inspection, network address translation, etc.**

Goal of this study is to investigate the service function chaining domain use case scenario for PISA data plane incorporating SR-MPLS as transport. We developed program code for SFC domain elements in a programmable protocol independent packet processing (P4) and python languages.

The outcome of this study is functionating SFC domain emulation in Mininet environment. Emulation results shows that SR-MPLS makes minimal overhead for all emulated SF paths in generalized proportion of approximately 1/1000 from overall transmission rate. Results also reveal a minimalistic transmission rate dependency on path length in generalized proportion of approximately 1/40 of transmission rate drop between shorter and longer SF paths. It's evident that SRMPLS is fit to be used for SFC on PISA switches.

***Keywords*—Intent-based networking, Programmable protocol-independent packet processing, Software defined networking, Service function chaining, Segment Routing with MPLS**

I. Introduction

The Internet – a perpetual interconnectivity service held under man's providence is a puzzle of many pieces. One of which namely Software Defined Networking (SDN) has approached a significant shift in its underlay hardware structure. Over past years a protocol-independent switch architecture (PISA) has made its way to both academical research and commercial products. PISA enables network equipment re-programmability. Turning SDN functionality development from proprietary vendor provision to an opensource device administrator driven task [1].

Service function chaining (SFC) is data traffic steering concept with objective on differentiating packet processing without modification of underlay network topology [2]. SFC gained traction due recent advancement of segment routing.

Segment routing with MPLS (SR-MPLS) is a routing method that uses source routing in combination with MPLS for encapsulation enabling reactive SFC initialization [3].

In this study we explore mentioned technologies in a Mininet emulation environment. We develop program code for SFC domain elements for PISA switches with use of programmable protocol-independent packet processing (P4) language and python program code for topology host nodes acting as service functions (SFs). Results show SR-MPLS as an adequate choice for SFC encapsulation as it generates minimal overhead. SFC length to transmission ratio shows insignificant dependency.

II. Related works

Authors of [4] points out the difference between OpenFlow and P4 switches. OpenFlow was introduced to program control plane. Thus, switch match-action pipeline stayed protocol dependent. P4 however was introduced to program data plane and make match-action pipeline protocol-independent. Which also allows in our experiment to utilize custom SF encapsulation processing.

Study [5] utilizes P4 registers to store address resolution protocol (ARP) information for enablement of autonomous forwarding. An ability for self-organization in P4 pipelines can reduce information exchange with network controller. Our study also reveals a need for control channels between SFC enabled P4 switches.

Authors of [6] propose congestion aware multi-path label switching algorithm for detection of elephant flows in data center networks. Like our study, P4 ability of customized packet processing is used for model development in Mininet.

Research [7] states that stateful SFs are common in real world deployments but rarely does an existing P4 literature examines them. We do cover the necessity for use of stateful SFs in section VII.

Authors [8] of study introduce a flexible system for NF offloading to P4 switches themselves. Which in SFC terminology would be equal to making an SF forwarding element do the service functions work. For example, P4 switch could work as a network address translator.

Like in our study, authors of [9] also find segment routing generated overhead for SFC excessive and propose a heuristic algorithm for SF encapsulation compression for SR-IPv6 routing. Our proposal of single segment identifier (SID) use is applicable either if proactive classification is available or path taken by the packet has been reactively backtracked as in their study.

III. Programmable packet processing

A. SDN packet processing capability provision shift

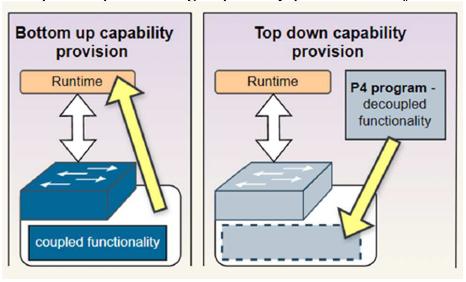

Bottom up capability provision
Runtime
coupled functionality
Top down capability provision
Runtime
P4 program - decoupled functionality

Fig. 1. Capability provision

In legacy as well as in older white switches (e.g. Open Virtual Switches) their functionality is coupled to inbuild logics and is dependent on vendor upgrade cycles. This type of capability provision is bottom up as information about available configuration of the switch is gathered from the switch itself. It is shown in Fig. 1.

Programmable packet processing allows to transform switch's functionality according to network administrator's instructions by compiling P4 programs. [10] This type of capability provision is called top down as functionality is enforced from upper control layer. It is shown in Fig. 1.

1) PISA architecture building blocks

PISA architectures building blocks and its derivation into applicable switch model is show in Figure 2 where:

- Parser – Incoming packets evaluation through multiple state filter according to its content for further processing. States are mechanisms that determine whether to continue packet processing or not and with which action to proceed.
- Match and action pipeline – Parsed header content match to predefined custom to users' requirements logic for appropriate modification and processing. Memory holds users' predefined logic in table values while arithmetic logic unit (ALU) allows for packet header content modification actions.
- Deparser – An exit point where transmittable packet is constructed from processed packet headers and original or modified payload before its egress from the appropriate switch interface.

Portable switch architecture (PSA) [11] introduces traffic manager (TM) building block. TM is responsible for fixed functions like packet buffering, reordering, and queuing, etc. TM can differentiate between hardware and its working principle is hardware vendor defined. V1model is PISA derivation implemented in bmv2 software switch of Mininet emulator that we used in our experiment.

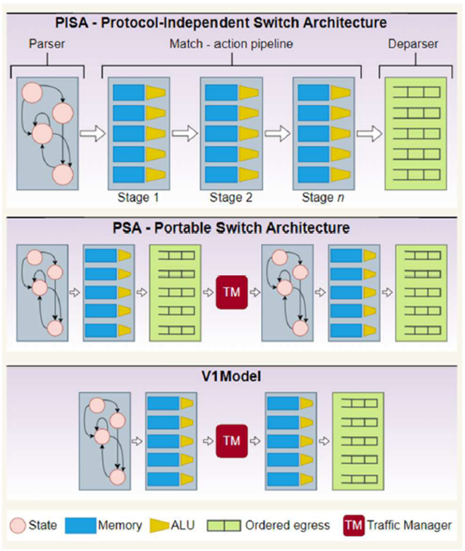


Fig. 2. PISA model derivation

IV. Segment Routing with MPLS

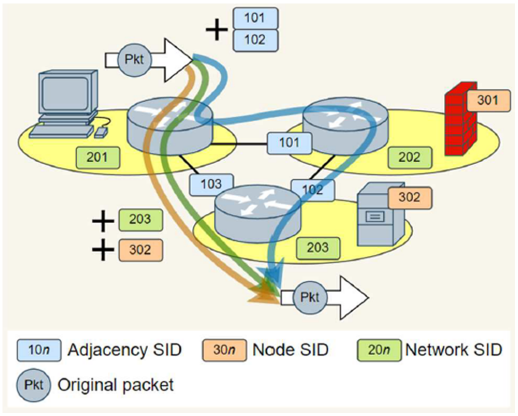


Fig. 3. SR-MPLS example

Segment routing with MPLS (SR-MPLS) utilizes labels as segment identifiers (SIDs). Shown in Fig. 3. is SR-MPLS with three possible routes. When original packet enters SRMPLS enabled routing domain an encapsulation is pushed on top of packet. This encapsulation can consist of a single label or label stack. One label is equivalent to one SID. There are three types of SIDs:

- Adjacency SID – used if the packet needs to be routed along a very specific path as these SIDs represent directly connected neighbors.
- Node SID – used if the packet needs to be routed to a specific network node.
- Network SID – used if the packet needs to be routed to a specific network.

The encapsulation can be popped upon exiting the SRMPLS enabled routing domain. A swap is a composite action made from push and pop action combination. Swap is performed inside a SR-MPLS domain for when packet path consists of multiple stacked labels or if label stacking is prohibited and swap is done on each packet hop.

In Fig. 3. a computer can reach server via use of three different SID types. A stack of adjacency SIDs represent blue path where SID 101 would be popped before packet reached the second router, while SID 102 would be popped before packet reached the third router. Although SIDs 203 and 302 represent different values their path is equivalent therefore they can be used interchangeably.

The difference between SR-MPLS and traditional MPLS is that SR-MPLS does not utilize an additional label distribution protocol (LDP). Instead, label information is exchanged among routers via routing protocol updates. Protocols in use are modified versions of Intermediate System-to-Intermediate System (IS-IS), Border Gateway Protocol (BGP) or Open Shortest Path First (OSPF) [3].

V. Service Function Chaining

Service function chaining (SFC) is designed to ease network configuration by allowing to steer traffic through ordered service function (SF) path without modification of underlay network topology. Originally it was introduced by Cisco conglomerate alongside a Network Service Header (NSH) protocol as applicable SF encapsulation. However, SFC quickly grew out of use in virtualized network functions and is heavily leveraged in 5G networks as well.

A. SFC placement in SDN conception

Network configuration policy setup with and without SFC thorough SDN structural planes is shown in Figure 4. With use of SFC user at a management plane is not required to specifically indicate which network elements should data traffic cross on its way to destination. It is enough to indicate what network functions needs to be in use. It is controller's task to map requested SFs with underlay network topology.

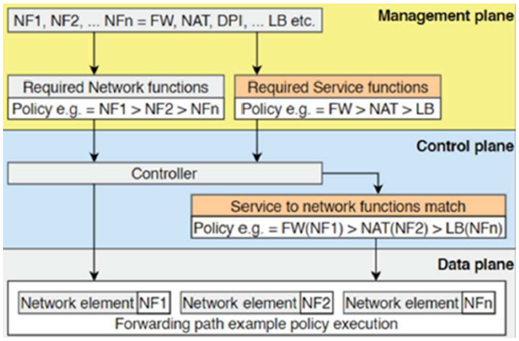

NF1, NF2, ... NFn = FW, NAT, DPI, ... LB etc.
Management plane
Required Network functions
Policy e.g. = NF1 > NF2 > NFn
Required Service functions
Policy e.g. = FW > NAT > LB
Control plane
Controller
Service to network functions match
Policy e.g. = FW(NF1) > NAT(NF2) > LB(NFn)
Data plane
Network element NF1
Network element NF2
Network element NFn
Forwarding path example policy execution

Fig. 4. SFC among SDN structural planes

We have examined SFC domain in our previous research [12] that can be in use if an in-depth explanatory of SFC elements is required.

B. SF path types

SF paths differentiate not only with SFs that they contain but also with how flows are handled. Three SF path types are shown in Figure 5. Service function forwarder (SFF) is SF domain element that is used only to forward network traffic along the path and does not modify it. SF Path types are:

1. SFC is applied for one of the data flow directions.
2. SFC application is asymmetrical for both directions.
3. SFC application is symmetrical for both directions.

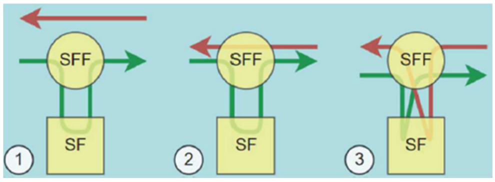


Fig. 5. SFC path types

We have previously studied paths symmetries effect on successful SF application [13]. Symmetrical SFC application might be an obligatory for SF chains with stateful SFs in them like non-transparent proxies, SIP servers, L7 firewalls.

C. Proactive SF path policy

For SFC to work an SF path policy is required. This policy holds the rule base of network traffic classification for appropriate SF encapsulation application. Proactive SF classification process is shown in Fig. 6.

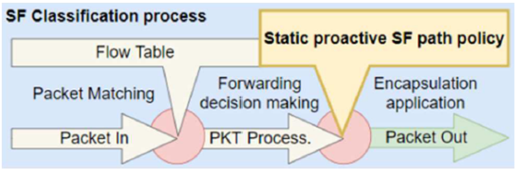

SF Classification process
Static proactive SF path policy
Flow Table
Packet Matching
Forwarding decision making
Encapsulation application
Packet In
PKT Process.
Packet Out

Fig. 6. SF classification process

Static proactive SF path policy is a rule base set in advance of any network traffic arrival by network administrator. Main disadvantage of path policy setup proactively is that only expected network traffic can be classified and enter SFC domain.

In our research we examined an alternative to proactive SF path policy which is reactive policy instantiation.[14] Reactive SF path discovery works on try and catch principle with utilization of SFC control plane channels for network traffic reclassification[15]. With use of reclassification a need for proactively set rule base would be eliminated.

VI. EMULATION SETUP

A. Network topology

To investigate working principle of PISA we developed an SFC domain shown in Fig. 7. Classifier pushes MPLS encapsulation on top of incoming network traffic coming from source. SF forwarders steers network traffic according to outermost SID in MPLS label stack. Service functions are transparent proxies. SF proxy removes bottom of stack label for packet to leave SFC domain and is sent to destination.

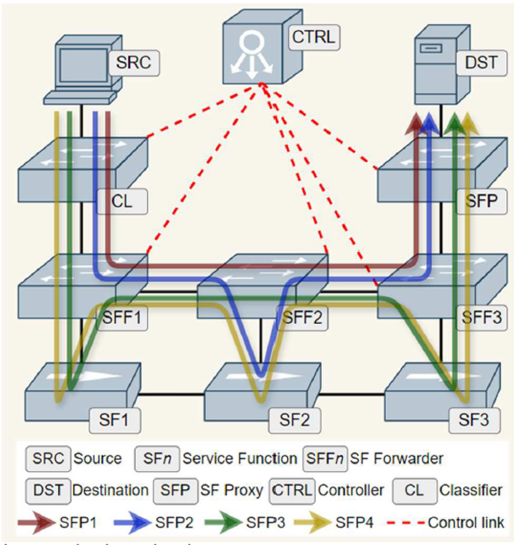


Fig. 7. Emulated SFC domain

There are four SF paths shown in Fig. 7. Each path differs from one another with their length. SFP1 is special. It does not cross any of the SFs, therefore, requires no encapsulation as its path matches one provided by underlay network topology. Controller is the network element in which the proactive SF path policy is stored for redistribution to all other network elements.

Experimental scenario was implemented in Mininet network emulator with use of P4-Utils[16]. P4-Utils provides not only emulation needed software but also environment setup script for dependency installation as well as literature and examples of various P4 enabled use cases.

The underlay network topology of our SFC domain is shown in Fig. 8. All switches are behavioral model version 2 (bmv2) software switches that work according to V1model architecture shown in Fig. 2. Switches use loopback interface as outbound link to the controller. Data path link bandwidth was set to 10Mbits/s.

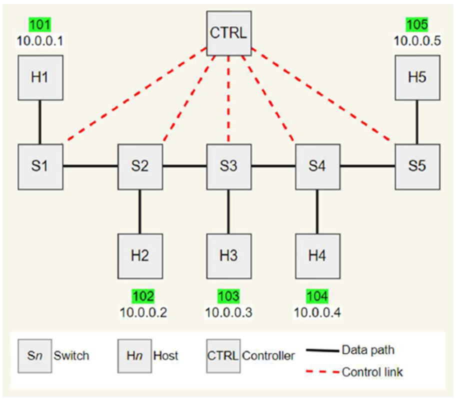


Fig. 8. Underlay network topology

Network element use was as follows:

- H1 – Source host (running Iperf as client).
- H2, H3, H4 – Service Functions (running proxy server python program from Fig. 10).
- S1, S2, S3, S4, S5 – SF forwarding elements (running P4 program from Fig. 9).
- H5 – Destination host (running Iperf as server).

B. Emulation process

The designing of emulation setup was a challenging task. It required dozens of reruns and misconfiguration fixes. The task included not only topology planning in Mininet, but we also developed a separate P4 program for SF forwarding elements, a python program for service functions and a python program for the controller.

When the network emulation environment was ready, we conducted 40 experiments (10 sequential runs for each SF path). The emulation process of each run was as follows:

1. Running network topology program – it starts the Mininet command line interface (CLI) along with bmv2 software switches with the P4 program and controller python program. It also starts packet capture on all switch interfaces.
2. Setting maximum transfer unit parameter for each host in topology from Mininet CLI. Lower for hosts 1 and 5, and higher for hosts 2, 3, and 4. (SFC underlay network topology shown in Fig. 8.).
3. Running service function program – through Xterm terminals on hosts 2, 3, and 4.
4. Starting Iperf server on host 5 – the TCP port to listen on set according to needed path (5565 port – SF path 1, 5566 port – SF path 2, 5567 port – SF path 3, and 5568 port – SF path 4).
5. Starting Iperf client – we used no additional arguments which makes Iperf to test the available throughput for 10 seconds with gradually increasing TCP window.
6. Repeating Iperf client start for 10 iterations.
7. Collecting network snapshot – making a copy of packet captures (each network interface gets a separate capture for ingress and egress traffic which makes in total of 26 captures in this topology). It’s worth mentioning, that realistic real-world cases would have way more network interfaces which potentially could be divided in sub interfaces as virtual local area network (VLAN) technology widely used in data center networks.
8. Repeating previous steps for all SF paths.

We endured a shortcoming of P4-Utils upon development of network emulation setup. The application programming interface (API) does not hold any configuration capability of host node maximum transfer unit (MTU) parameter. We overcame this shortcoming by manually setting MTU values for each host in Mininet CLI after topology initialization and before running any Iperf. This was needed as Scapy packet crafting tool used in SFs did not allow frames bigger than MTU of the host. This problem arose due to Iperf utilizing maximum of allowed packet size and P4 switches added an MPLS label on top of that. Therefore, we needed to limit Iperf MTU and rise SFs MTU.

C. P4 program for SF forwarding elements

Flow diagram of SF forwarding elements (classifier, SF forwarders, and SF proxy) logic is shown in Fig. 9. The logic is implemented in program code written in programmable protocol-independent packet processing (P4) language [17].

As shown in Fig. 9. an incoming packet is examined whether it already has an MPLS or not by parsing its ethernet header for ethernet header type. If encapsulation is present (type 0x8847) packet is matched against SID database and forwarded further trough an appropriate egress port. If encapsulation is not present, then TCP header is parsed. If TCP destination port has a match in proactively setup SF classification policy, then an SFC encapsulation is enforced, and packet is forwarded further through an appropriate egress port. However, if no match in SF classification policy is found, then packet is forwarded according to L2 hardware address.

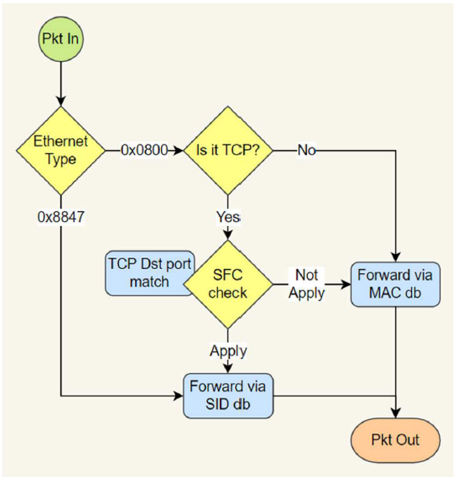


Fig. 9. Logic of SF forwarding elements

D. Python program for service functions

Flow diagram of service functions logic is show in Fig. 10. This logic is used in python program run by hosts which act as SFs. Main functionality of this program is provided by Scapy packet crafting tool [18].

On packet arrival it is filtered whether it is MPLS or not. Afterwards destination hardware address is examined to distinguish whether captured packet is incoming or outgoing. If it is incoming only then time to live (TTL) of MPLS and IP headers is modified, and outermost label match is done.

We encountered an issue with Scapy as a noticeable drop of transmission rate appeared for all paths that crossed SFs. After path examination of packet captures, we noticed that lots of packets delivered to first SF in path are being lost, which causes Iperf to limit its rate. Through online query we settled on potential cause being Scapy's sendp() function which opens and closes a socket for each packet.

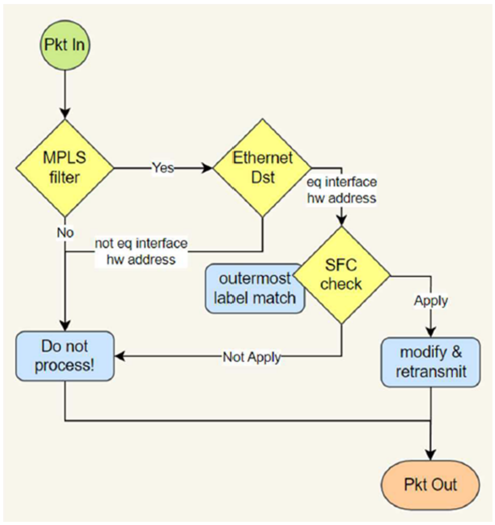


Fig. 10. Logic of service functions

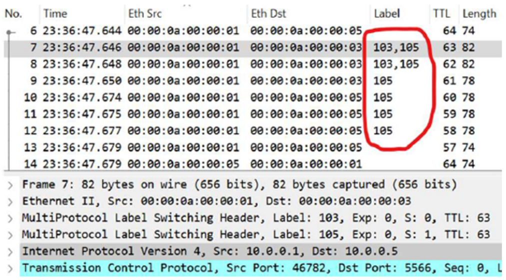
VII. Results and Analysis

A. Emulation results

P4-Utils has an in-built functionality of making a packet capture of each P4 switch interface. In same time P4-Utils also allows active logging for all switches. The combination of both abilities can be referred to as network snapshot creation. Network snapshots relaxes network troubleshooting task as problem solving no longer requires a separate hop at the time approach and enables a cross-examination among all links. This allowed us to understand the traffic rate problem mentioned in the end of previous section.

As shown in Fig. 8. MPLS encapsulation was globally significant and resembled host IP addressing. For example, node SID of 103 instructs switch to forward packet in hosts with IP 10.0.0.3 direction, SID 104 would be packet forwarding to host with IP 10.0.0.4 and so on.

Service function path 1 (red arrow in Fig. 7.) is packet switched by hardware address therefore it gains no SF encapsulation as it does not to cross any SFs. This path was created to test whether SF classification worked correctly.

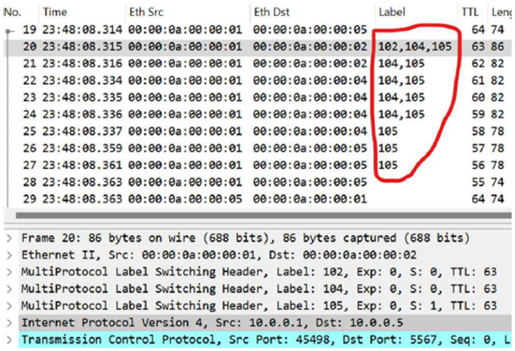


```
No.  Time          Eth Src            Eth Dst            Label     TTL Length
  6 23:36:47.644 00:00:0a:00:00:01 00:00:0a:00:00:05            64 74
  7 23:36:47.646 00:00:0a:00:00:01 00:00:0a:00:00:03 103,105    63 82
  8 23:36:47.648 00:00:0a:00:00:01 00:00:0a:00:00:03 103,105    62 82
  9 23:36:47.650 00:00:0a:00:00:01 00:00:0a:00:00:03 105        61 78
 10 23:36:47.674 00:00:0a:00:00:01 00:00:0a:00:00:05 105        60 78
 11 23:36:47.675 00:00:0a:00:00:01 00:00:0a:00:00:05 105        59 78
 12 23:36:47.677 00:00:0a:00:00:01 00:00:0a:00:00:05 105        58 78
 13 23:36:47.679 00:00:0a:00:00:01 00:00:0a:00:00:05            57 74
 14 23:36:47.679 00:00:0a:00:00:05 00:00:0a:00:00:01            64 74

> Frame 7: 82 bytes on wire (656 bits), 82 bytes captured (656 bits)
> Ethernet II, Src: 00:00:0a:00:00:01, Dst: 00:00:0a:00:00:03
> MultiProtocol Label Switching Header, Label: 103, Exp: 0, S: 0, TTL: 63
> MultiProtocol Label Switching Header, Label: 105, Exp: 0, S: 1, TTL: 63
> Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.5
> Transmission Control Protocol, Src Port: 46782, Dst Port: 5566, Seq: 0, L
```

Fig. 11. SF path 2

Service function path 2 (blue arrow in Fig. 7.) is shown in Fig. 11. From merging packet captures taken in multiple places throughout the topology we were able to track SF encapsulation changes along the taken path. Circled with red is the SIDs in use. Packet nr. 6. and 13. shows that original packet enters and exits SFC domain with no encapsulation. From the TTL we can see that the path is 8 hops long. It's also evident that L2 hardware addressing gets modified but L3 logical does not.

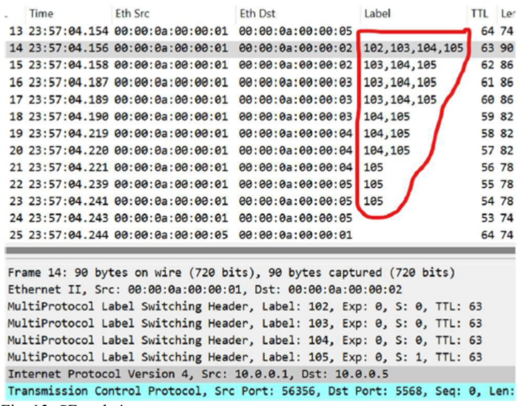


```
No.  Time          Eth Src            Eth Dst            Label         TTL Leng
 19 23:48:08.314 00:00:0a:00:00:01 00:00:0a:00:00:05                64 74
 20 23:48:08.315 00:00:0a:00:00:01 00:00:0a:00:00:02 102,104,105    63 86
 21 23:48:08.316 00:00:0a:00:00:01 00:00:0a:00:00:02 104,105        62 82
 22 23:48:08.334 00:00:0a:00:00:01 00:00:0a:00:00:04 104,105        61 82
 23 23:48:08.335 00:00:0a:00:00:01 00:00:0a:00:00:04 104,105        60 82
 24 23:48:08.336 00:00:0a:00:00:01 00:00:0a:00:00:04 104,105        59 82
 25 23:48:08.337 00:00:0a:00:00:01 00:00:0a:00:00:04 105            58 78
 26 23:48:08.359 00:00:0a:00:00:01 00:00:0a:00:00:05 105            57 78
 27 23:48:08.361 00:00:0a:00:00:01 00:00:0a:00:00:05 105            56 78
 28 23:48:08.363 00:00:0a:00:00:01 00:00:0a:00:00:05                55 74
 29 23:48:08.363 00:00:0a:00:00:05 00:00:0a:00:00:01                64 74

> Frame 20: 86 bytes on wire (688 bits), 86 bytes captured (688 bits)
> Ethernet II, Src: 00:00:0a:00:00:01, Dst: 00:00:0a:00:00:02
> MultiProtocol Label Switching Header, Label: 102, Exp: 0, S: 0, TTL: 63
> MultiProtocol Label Switching Header, Label: 104, Exp: 0, S: 0, TTL: 63
> MultiProtocol Label Switching Header, Label: 105, Exp: 0, S: 1, TTL: 63
> Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.5
> Transmission Control Protocol, Src Port: 45498, Dst Port: 5567, Seq: 0, L
```

Fig. 12. SF path 3

Service function path 3 (green arrow in Fig. 7.) is shown in Fig. 12. This path is 10 hops long. It's also more evident that the L2 addressing changes occur according to next host node in SF path on its egress. The same logic is used for MPLS label stack, but the pop happens before the SF ingress. For example, at nr. 21. packet enters SF1 with L2 destination address 00:00:0a:00:00:02 that is equal to SF1 interfaces hardware address but label with SID 102 has been popped by the previous SFC element SFF1. The remaining SF encapsulation with the outermost label containing SID 104 tells SF1 and next SFC element along the path that packet needs to be routed to SF3.



```
.  Time         Eth Src            Eth Dst            Label            TTL Ler
13 23:57:04.154 00:00:0a:00:00:01  00:00:0a:00:00:05                   64  74
14 23:57:04.156 00:00:0a:00:00:01  00:00:0a:00:00:02  102,103,104,105  63  90
15 23:57:04.158 00:00:0a:00:00:01  00:00:0a:00:00:02  103,104,105      62  86
16 23:57:04.187 00:00:0a:00:00:01  00:00:0a:00:00:03  103,104,105      61  86
17 23:57:04.189 00:00:0a:00:00:01  00:00:0a:00:00:03  103,104,105      60  86
18 23:57:04.190 00:00:0a:00:00:01  00:00:0a:00:00:03  104,105          59  82
19 23:57:04.219 00:00:0a:00:00:01  00:00:0a:00:00:04  104,105          58  82
20 23:57:04.220 00:00:0a:00:00:01  00:00:0a:00:00:04  104,105          57  82
21 23:57:04.221 00:00:0a:00:00:01  00:00:0a:00:00:04  105              56  78
22 23:57:04.239 00:00:0a:00:00:01  00:00:0a:00:00:05  105              55  78
23 23:57:04.241 00:00:0a:00:00:01  00:00:0a:00:00:05  105              54  78
24 23:57:04.243 00:00:0a:00:00:01  00:00:0a:00:00:05                   53  74
25 23:57:04.244 00:00:0a:00:00:05  00:00:0a:00:00:01                   64  74

Frame 14: 90 bytes on wire (720 bits), 90 bytes captured (720 bits)
Ethernet II, Src: 00:00:0a:00:00:01, Dst: 00:00:0a:00:00:02
MultiProtocol Label Switching Header, Label: 102, Exp: 0, S: 0, TTL: 63
MultiProtocol Label Switching Header, Label: 103, Exp: 0, S: 0, TTL: 63
MultiProtocol Label Switching Header, Label: 104, Exp: 0, S: 0, TTL: 63
MultiProtocol Label Switching Header, Label: 105, Exp: 0, S: 1, TTL: 63
Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.5
Transmission Control Protocol, Src Port: 56356, Dst Port: 5568, Seq: 0, Len:
```

Fig. 13. SF path 4

Service function path 3 (yellow arrow in Fig. 7.) is shown in Fig. 13. This path is 12 hops long. Source routing is most evident in this route as the original packet gains a four SID long MPLS label stack at the encapsulation. The label stack slowly depletes as packet crosses all three SFs. Its worth to note that, the encapsulation stacks length in this experiment is directly related to the network topology that is being emulated.

TABLE I. TCP DOWNSTREAM CONNECTION PARAMETERS

Nr.	*Measurement*	*Path 1*	*Path 2*	*Path 3*	*Path 4*
1.	Avg. pkt rate, [pps]	138.2	37.8	38.2	39.7
2.	Avg. bitrate [Kb/s]	9924	1490	1438	1451

Table I shows data transmission rate. Iperf downstream flow is the one going from client to server as it contains the payload with data. Path 1 shows that Mininet bandwidth parameter set to 10Mbit/s was working accordingly and packet switching had insignificant effect on the transmission.

TABLE II. SERVICE FUNCTION INPUT / OUTPUT PACKET COUNT

Nr.	*SFC Path*	*SF 1*		*SF 2*		*SF 3*	
		Rx	*Tx*	*Rx*	*Tx*	*Rx*	*Tx*
1.	Path 2	-	-	3012	2641	-	-
2.	Path 3	3200	2828	-	-	2828	2826
3.	Path 4	3223	2895	2895	2895	2895	2892

Table II shows received (Rx) and transmitted (Tx) packet count difference for each SF and each path. It is evident that packet rate decrees occur at the first crossed SF in each path. For example, Path 2 crosses only service function 2 and the Rx packet count is much bigger than the Tx packet count.

The difference in I/O packet count likely is a cause of the significant data transmission rate difference between path 1 and rest of the paths shown in Table I. As mentioned earlier rate limiting occurs due to working principle of Scapy tool.

We have studied Intent-based networking (IBN) structure and supposed working principle [19]. In IBN network control and maintenance is achieved with human to artificial intelligence (AI) dialog. It is suggested that AI and machine learning (ML) can utilize network snapshots as network state monitoring mechanism for pre and post configuration change. In the current experiment network snapshot aided in troubleshooting of data transmission rate problem.

TABLE III. SF ENCAPSUALTION OVERHEAD

Nr.	MPLS Overhead for TCP downstream connection [bits/s]					
	SFC Path	*CL >*	*SF 1 >*	*SF 2 >*	*SF 3 >*	*Avg.*
1.	Path 2	1551	-	685	-	1118
2.	Path 3	2237	1327	-	663	1409
3.	Path 4	2928	1994	1328	663	1728

Table III shows SF encapsulation overhead in bits per second. At the beginning of each path the overhead is much larger than that at the end. This is the effect of source routing working principle of SR-MPLS. The classifier must attach all necessary SIDs to the original packet for it to be steered along the desired path. Before each SF an SF forwarder (SF) pops outermost label containing the adjacent SFs SID.

B. Conducted analysis

From the gained results shown in previous section we conducted an analysis with use of Octave. Values shown in Table I and Table III of results are mean values calculated from 10 iterations for each path with mean function.

Shown in Fig. 14, Fig. 15., and Fig. 16. measured value points at SFC path lengths 8, 10, and 12 corresponds to path 2, path 3, and path 4 respectively. SF path 1 was not included in the analyses as it did not utilize SR-MPLS routing.

Calculation of encapsulation overhead to path length dependency shown in Fig. 14 was done as follows:

1. Creating a vector of MPLS header overhead in bits per second for each path – data source is analyses of protocol hierarchy with Wireshark for each network snapshot.
2. Calculating of mean values from made vectors with mean function.
3. Calculating standard deviation values from vectors made in point 1. with std function.
4. Creating a vector for x axis of values of 6 to 14 with step 1 with use of linespace function.
5. Creating coefficients of linear regression according to formula (1) from mean points and path lengths respectively with use of polyfit function with curve value 1.
6. Creating a linear regression vector from mean values and x axis vectors with polyval function.
7. Plotting measured values with use of errorbar function and linear regression with use of plot function in single figure.

Similar methodology was repeated for graphs in Figures 15. and 16. Data source for those were network snapshot analyses of endpoint conversations through Wireshark.

Linear regression formula (1) is as follows:

$$a * x + b = y \qquad (1)$$

In this study x represents SF path length in hops for all graphs, y represents measured values for each graph respectively, but coefficients a and b are regressions slope determining factor calculated with use of polyfit and polyval functions.

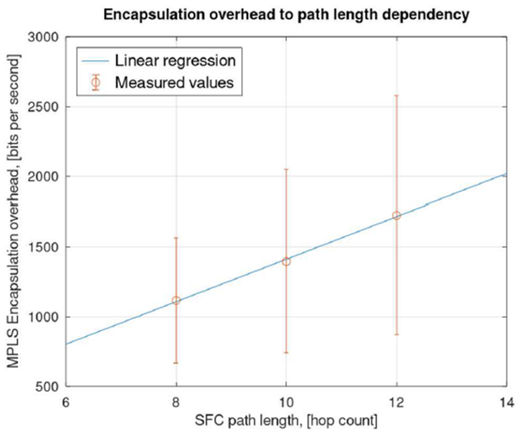


Fig. 14. Encapsulation overhead dependency

SF encapsulation (MPLS labels representing node SIDs) overhead to SF path length (hop count packet takes in SFC domain) dependency is shown in Fig. 14.

Calculated linear regression shows that encapsulation overhead rises with the path length. Path with 8 hops has overhead of approximately 1000 bits per second (bps), but a path with 12 hops has overhead of approximately 1700 bps. This corresponds to 175 bps increase per hop in emulated topology.

An increase is brought in by routing mechanism in use - SR-MPLS. Segment routing uses combination of source routing and MPLS encapsulation. Encapsulation amount required depends on paths complexity. All path's description must be added to the original packet at its ingress in SFC domain for source routing to work.

To eliminate overheads dependency of paths complexity in any topology we propose a use of a single segment ID for each SF path. This could potentially lead to use of hierarchy structured SIDs.

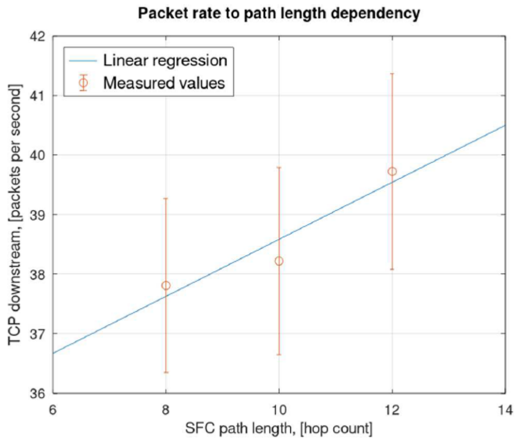


Fig. 15. Packet rate dependency

Packet rate of TCP downstream in packets per second (pps) to SF path length dependency is shown in Fig. 15.

Calculated linear regression shows a packet rate increase with longer SF paths. Packet rate is approximately 38 pps for path of 8 hops and approximately 40 pps for path of 12 hops. This corresponds to 0.5 pps increase per hop in emulated topology.

The packet per second increases was due to 1. SF path type shown in Fig. 5. in use. As the SFs were a transparent proxy servers, they were stateless and held no need for the return (upstream) TCP flow to cross them.

The affected element of flow asymmetry was TCP window. The source and destination nodes unsuccessfully tried for multiple times to arrange a window upscale. So, having more configuration dialog with no payload to carry allude to a slight packet rate increase with no added value.

To avoid the unwanted dialog, we propose use of more granular network traffic classification. For example, enforcing 3. SF path type (shown in Fig. 5.) to be in use for bidirectional communication, and 2. SF path type for use of one-way communication.

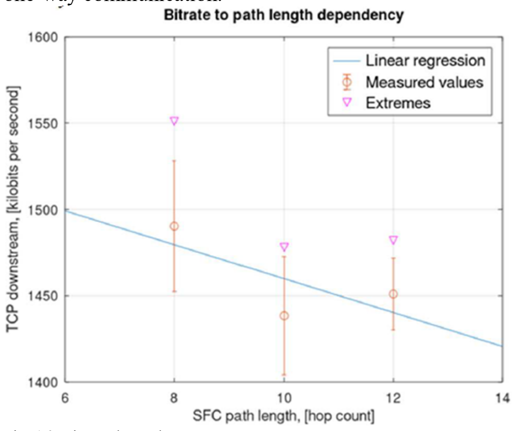


Fig. 16. Bitrate dependency

Bitrate of TCP downstream in kilobits per second (kbps) to SF paths length dependency is shown in Fig. 16. Measured values show mean points from 10 measurements with a standard deviation. Extremes show maximum of the achieved bitrate among same 10 measurements.

Although the calculated linear regression suggests a steady slope downwards, having both extremes and mean value points not following the pattern makes a contradiction.

Bitrate below 1450 kbps for 10 hop long path is the lowest among measured values. Value in this graph can be explained by cross-examining graphs in Fig. 14. and Fig. 15. A common culprit of 10 hop path falling out of linear regression is observable.

The observed occurrence is explainable by having two parameters change among emulated paths where one is paths length change, while the other is count of crossed SFs. Although, both are directly proportional (8 hop path visits 1 SF, 10 hop path visits 2 SFs, and 12 hop path visits 3 SFs), having a multiple parameter change does complexes measurement relation linearity.

Therefore, the uneven bitrate drop is an outcome of both multiple parameters change among paths and TCP window fluctuations discussed under packet rate analyses.

For a more precise performance evaluations, we suggest conduction of fine-grained comparisons. For example, path difference on a single parameter change.

C. Future work

The development did not go without cutting corners:

- The network traffic classification does not reach all SFC elements. Only the classifier (CL in Fig. 7. and s1 in Fig. 8.) can utilize it. This leaves no option for network traffic reclassification.
- The forwarding tables are filled statistically and designed only for emulated topology (Fig. 8). This strains the reusability of our research.
- Even though we succeeded in writing the P4 program code in a way that it is independent from any specific network path descriptors (IP addressing, MAC addressing, segment IDs e. c.), still for egress control we used an action with no tables. That might not be a correct implementation and does requires a revision.
- SF python program also uses try catch as a condition statement for label stack parsing which also should be revisited in future iterations.
- The communication with the controller was through localhost. A more realistic network models would require inbound communication.

An exploration of fundamental characteristics is only a groundwork for a more sophisticated analysis of following:

- Reactive SF path classification implementation – the ability to reclassify network traffic if the SF it's directed at is not fit to fulfil the desired service.
- Path SID forwarding – the ability to map whole path packet needs to be steered along with utilization of a single SID. This could potentially also allow structurization of SID hierarchy.
- SFC control channel implementation – enabling informational message exchange among all SFC elements and the control plane.
- Comprehensive evaluations – a fin-grained division between SF paths should be implemented. SFs individual shortcomings should be excluded from overall SF path analysis.

Conclusion

We used Mininet network emulator with P4-Utils and Scapy to create an emulation environment. This required development of program code in P4 and python languages. We made developed code available through GitHub [20].

Analysis shows that SR-MPLS does create a minimal overhead of 2.5 kbps for a flow of 1451 kbps that constitutes to a ratio of 1/580.

Analysis also revealed that a longer SF path does not contribute to a significant packet drop. As path with 8 hops runs at a rate of 1500 kbps and one with 12 hops runs at a rate of 1450 kbps making a drop of 12.5 kbps per hop. Thus, bitrate drop ratio is 1/120.

To limit encapsulation dependency on path length we propose use of a single SID describing the whole path in contrary of using label stack made from node, network, and adjacency SIDs.

We also propose to direct bidirectional communications both flows via same path to avoid inconsistency in path parameter configuration.

References

[1] E. F. Kfoury, J. Crichigno and E. Bou-Harb, "An Exhaustive Survey on P4 Programmable Data Plane Switches: Taxonomy, Applications, Challenges, and Future Trends," in IEEE Access, vol. 9, pp. 87094-87155, 2021

[2] J. Halpern and C. Pignataro, "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, Available: https://www.rfc-editor.org/info/rfc7665

[3] P. L. Ventre et al., "Segment Routing: A Comprehensive Survey of Research Activities, Standardization Efforts, and Implementation Results," in IEEE Communications Surveys & Tutorials, vol. 23, no. 1, pp. 182-221, Firstquarter 2021

[4] Z. Liu, P. Cui, Y. Hu, Y. Dong, K. Tang and L. Xue, "A programmable data plane that supports definable computing," 2021 International Conference on Advanced Computing and Endogenous Security, Nanjing, China, 2022

[5] J. Alvarez-Horcajo, I. Martķnez-Yelmo, D. Lopez-Pajares, J. A. Carral and M. Savi, "A Hybrid SDN Switch Based on Standard P4 Code," in IEEE Communications Letters, vol. 25, no. 5, pp. 1482-1485, May 2021

[6] Y. -K. Chang, H. -Y. Wang and Y. -H. Lin, "A Congestion Aware Multi-Path Label Switching in Data Centers Using Programmable Switches," 2021 IEEE International Conference on Networking, Architecture and Storage (NAS), Riverside, CA, USA, 2021

[7] X. Zhang, L. Cui, F. P. Tso and W. Jia, "Compiling Service Function Chains via Fine-Grained Composition in the Programmable Data Plane," in IEEE Transactions on Services Computing, vol. 16, no. 4, pp. 2490-2502, 1 July-Aug. 2023

[8] J. Ma, S. Xie and J. Zhao, "Flexible Offloading of Service Function Chains to Programmable Switches," in IEEE Transactions on Services Computing, vol. 16, no. 2, pp. 1198-1211, 1 March-April 2023

[9] Y. Wang, X. Zhang, L. Fan, S. Yu and R. Lin, "Segment Routing Optimization for VNF Chaining," ICC 2019 - 2019 IEEE International Conference on Communications (ICC), Shanghai, China, 2019

[10] Noa Zilberman at University of Cambridge, P4 Tutorial Welcome, [Online] Available: https://github.com/p4lang/tutorials/blob/master/P4_tutorial.pdf, last viewd June 2024

[11] P4.org Architecture Working Group, Portable switch architecture, [Online] Available: https://p4.org/p4-spec/docs/PSA-v1.1.0.html, last viewed June 2024

[12] M. Mihaeljans and A. Skrastins, "Network Topology-aware Service Function Chaining in Software Defined Network," 2020 28th Telecommunications Forum (TELFOR), Belgrade, Serbia, 2020

[13] M. Mihaeljans and A. Skrastins, "Evaluation of reactive service function path discovery in symmetrical environment," Telfor Journal, vol. 14, no. 1, pp. 2-7, 2022

[14] M. Mihaeljans and A. Skrastins, "Reactive Service Function Path Discovery Approach in Software Defined Network," 2021 29th Telecommunications Forum (TELFOR), Belgrade, Serbia, 2021

[15] M. Boucadair, "Service Function Chaining (SFC) Control Plane Components & Requirements," draft-ietf-sfc-control-plane-08 (Informational), October 2016.

[16] Networked Systems Group at ETH Zürich, P4-Utils, [online] Available: https://nsg-ethz.github.io/p4-utils/, last viewed June 2024

[17] P4 Language Consortium, P4 Language Specification, [Online] Available: https://p4.org/p4-spec/docs/P4-16-v1.0.0-spec.html, last viewed June 2024

[18] P. Biondi, Scapy, [online] Available: https://scapy.readthedocs.io/, last viewed June 2024

[19] A. Clemm, L. Ciavaglia, and L. Z. Granville, "Intent-Based Networking - Concepts and Definitions," IRTF RFC 9315, 2022

[20] M. Mihaeljans and A. Skrastins, Program code used in this study, [Online], Available: https://github.com/MartinsMihaeljans/SFC-on-P4/tree/main, last viewed June 2024

Mārtiņš Mihaeljans was born in 1994. He received a Bachelor's degree in Electrical Science (2018) and a Master's degree in Telecommunications (2020) from Riga Technical University. His research interests are related to packet-switched computer networks.