Antti Nykänen

EVPN in Private Cellular Networks

Metropolia University of Applied Sciences

Master of Engineering

Information Technology

Master’s Thesis

20 December 2023

Abstract

Author:	Antti Nykänen
Title:	EVPN in Private Cellular Networks
Number of Pages:	80 pages
Date:	20 December 2023
Degree:	Master of Engineering
Degree Programme:	Information Technology
Professional Major:	Networking and Services
Supervisors:	Ville Jääskeläinen, Principal Lecturer

In the rapid evolution of Industry 4.0, the demand for robust and flexible wireless data transmission systems has surged, especially with the growing integration of the Internet of Things (IoT) and other similar initiatives. This thesis addresses the challenge of effectively deploying private 4G and 5G networks in industrial environments, where end-to-end Layer 2 Ethernet connectivity is often desired.

The thesis proposes a solution that leverages the connectivity provided by of private 4G and 5G networks using Border Gateway Protocol (BGP)-based control planes for dynamic configuration of Layer 2 Virtual Private Networks (VPNs). This approach aims to reduce the manual configuration in large-scale deployments of L2 tunnelling.

To begin the study, various networking standards and protocols relevant for problem at hand were examined in detail. A solution was then proposed, and a Proof of Concept (PoC) was developed to validate it. The results demonstrated the feasibility of dynamically configuring Layer 2 tunnels over 4G and 5G networks using BGP EVPN with VXLAN tunnelling, achieving significant reduction in manual configuration and maintaining operational flexibility.

In summary, the results show that the VXLAN EVPN with BGP-based control plane in private 4G and 5G networks is a viable solution for wireless data transmission in industry.

Keywords: Private Wireless, 5G, 4G, VXLAN, BGP, EVPN

Contents

List of Abbreviations

3GPP	3rd Generation Partnership Program
4G	Fourth Generation (mobile network)
5G	Fifth Generation (mobile network)
A-D	Auto-Discovery
AFI	Address Family Identifier
AI	Artificial Intelligence
AMF	Access and Mobility Management Function
AS	Autonomous System
ASIC	Application Specific Integrated Circuits
ATM	Asynchronous Transfer Mode
BGP	Border Gateway Protocol
BSD	Berkeley Software Distribution
BUM	Broadcast, Unknown and Multicast
BoS	Bottom of Stack
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSMA/CD	Carrier Sense, Multiple Access with Collision Detect
CUPS	Control and User Plane Separation
DN	Data Network
DOVE	Distributed Overlay Virtual nEtwork Architecture
DPDK	Data Plane Development Kit
DSL	Digital Subscriber Line
E-UTRAN	Evolved Universal Terrestrial Radio Access Network
ECMP	Equal Cost Multipath
EDGE	Enhanced Data Rates for GSM Evolution
EGP	Exterior Gateway Protocol
eMMC	Embedded Multimedia Card
EPC	Evolved Packet Core
ESI	Ethernet Segment ID
EVPN	Ethernet Virtual Private Network

EoGRE	Ethernet over GRE
FCS	Frame Check Sequence
FRR	Free Range Routing
FTP	File Transfer Protocol
GENEVE	Generic Network Virtualization Encapsulation
GPE	Generic Protocol Extension
GPRS	General Packet Radio Service
GRE	Generic Routing Encapsulation
GSM	Global System for Mobile Communications
GTP	GPRS Tunneling Protocol
HDLC	High-Level Data Link Control
HSDPA	High-Speed Downlink Packet Access
HSS	Home Subscription Server
HTTP	HyperText Transfer Protocol
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol
IMET	Inclusive Multicast Ethernet Tag
IMS	IP Multimedia Subsystem
IP	Internet Protocol
IPR	Intellectual Property Rights
IPv4	Internet Protocol, version 4
IPv6	Internet Protocol, version 6
IRQ	Interrupt Request
ISDN	Integrated Services Digital Network
ISO	International Organization for Standardization
IoT	Internet of Things
L2TP	Layer 2 Tunneling Protocol
L2TPv2	L2TP version 2
L2TPv3	L2TP version 3
LAN	Local Area Network
LSR	Label Switching Router

LTE	Long-Term Evolution
MAC	Media Access Control
MAN	Metropolitan Area Network
MME	Mobility Management Entity
MPLS	Multiprotocol Label Switching
NAPT	Network Address Port Translation
NAS	Non-Access Stratum
NAT	Network Address Translation
NF	Network Function
NIC	Network Interface Controller
NLRI	Network Layer Reachability Information
NR	New Radio
NVGRE	Network Virtualization Using Generic Routing Encapsulation
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
OvS	Open vSwitch
P-GW	Packet Gateway
PA	Path Attributes
PCEF	Policy and Charging Enforcement Function
PCF	Policy and Charging Function
PCI	Peripheral Component Interconnect
PCIe	PCI Express
PCRF	Policy and Charging Rules Function
PDU	Protocol Data Unit
PPP	Point-To-Point Protocol
PSTN	Public Switched Telephone Network
PoC	Proof-of-Concept
RAM	Random Access Memory
RD	Route Distinguisher
RFC	Request for Comments
RIP	Routing Information Protocol
RT	Route Target
S-GW	Serving Gateway
SAFI	Subsequent Address Family Identifier

SD	Secure Digital
SDN	Software Defined Networking
SFP	Small Form-factor Pluggable
SIM	Subscriber Identity Module
SMF	Session Management Function
SSH	Secure Shell
STP	Spanning Tree Protocol
TC	Traffic Control
TCP	Transmission Control Protocol
TDF	Traffic Detection Function
ToR	Top of Rack
UDM	Unified Data Management
UDP	User Datagram Protocol
UDR	User Data Repository
UE	User Equipment
UMTS	Universal Mobile Telecommunications System
UPF	User Plane Function
UR	Unfeasible Routes
USIM	Universal SIM
VID	Virtual Network Identifier
VLAN	Virtual Local Area Network
VNI	VXLAN Network Identifier
VPN	Virtual Private Network
VSID	Virtual Subnet ID
VXLAN	Virtual eXtensible LAN
WLAN	Wireless Local Area Network
XDP	eXpress Data Path

1 Introduction

In recent years, industries have embraced advanced technologies, such as the Internet of Things (IoT), Artificial Intelligence (AI), and data analytics, as part of their Industry 4.0 initiatives, leading to a significant increase in the need for wireless data transmission. Wireless data transmission enables flexible communication and data exchange between connected devices and systems, allowing factories and industrial facilities to collect and analyse vast amounts of data. This information can be used to improve operations, enhance efficiency, and increase productivity.

Private fourth and fifth generation (LTE/4G and 5G) mobile networks are a solution that many companies are currently interested in using for their Industry 4.0 communication requirements. The case company is a leading supplier of private 5G and 4G solutions. The case company is a multinational telecommunications company, information technology, and consumer electronics corporation. The company operates in 130 countries, providing critical communications infrastructure for telecommunications operators, industries, and the public sector.

While there are many benefits from using 4G / 5G instead of other wireless technologies, the 3rd Generation Partnership Program (3GPP) architecture they're based on is paradigmatically rather different from e.g. Wireless Local Area Network (WLAN) technologies. This means that deployments where existing Local Area Network (LAN)-based architectures are made wireless by utilizing 5G or 4G, the resulting new architecture must be carefully planned.

Especially in industrial applications, some of the network protocols that are used do not work over pure Internet Protocol (IP)-based communication paths that the 3GPP architecture typically provides, but they operate directly on the Layer 2 of the Open Systems Interconnection (OSI) model. With these protocols, a Layer 2 tunnelling protocol is typically used to provide connectivity over IP.

While the 3GPP specifications outline Ethernet PDU functionality for Ethernet transport directly over cellular networks, without an additional tunnelling layer, this feature hasn't gained widespread adoption yet. And for 4G, it's extremely dubious whether any such functionality will ever be implemented – although the specifications do seem to allow for backporting it.

The configuration and maintenance of large deployments involving the use of tunnelling protocols by manual configuration is quite labour intensive, as each tunnel endpoint and changes to them require individual manual configuration. Since configuration is static, tunnel endpoint IP addresses, need to remain static on both ends. This requires further configuration and planning and reduces operational flexibility.

Internet Engineering Task Force (IETF) Request for Comments (RFC) documents 7432 [1] and 8365 [2], among others, describe how a Border Gateway Protocol (BGP)-based control plane can be utilized for dynamic configuration of Layer 2 Virtual Private Networks (VPNs). The goal of this thesis is to evaluate these solutions for use over 5G and 4G networks, to reduce the amount of manual configuration required for industrial private wireless tunnelling deployments, and to enable the use of other functionality provided by the BGP EVPN control plane, such as the selection of a back-up route in case of failure.

This thesis is divided into six main sections. The first section introduces the topic, giving readers a background on the issues this thesis aims to address. In section two, we outline the research design and study methodology, explaining the approaches and materials used to gather information for this thesis.

The third section breaks down various crucial standards and protocols in networking. It starts with an introduction to the OSI Model for Networking, then moves to a discussion on Ethernet, IP, transport protocols, and IP routing. We also look at IP packet delivery in 3GPP LTE and 5G systems. The section ends with an overview of Layer 2 Ethernet tunnelling protocols BGP-based EVPN solutions.

Section four looks at the hardware offloading capabilities available for layer 2 tunnelling protocols, helping readers understand the requirements and limitations present in current technology setups.

In section five, a PoC proposal and its implementation are presented, where the knowledge from the earlier sections is used to develop a potential solution to the problem at hand.

Finally, section six concludes the thesis, summarizing the findings and offering a look into potential future developments in the field.

2 Methods and Material

As stated above, the goal of this thesis is to come up with a proposal for a protocol suite upon which dynamic configuration of Layer 2 tunnels can be based, and then build and test a proof-of-concept setup based on the selected technologies.

2.1 Research Design

The research design is outlined in Figure 1.

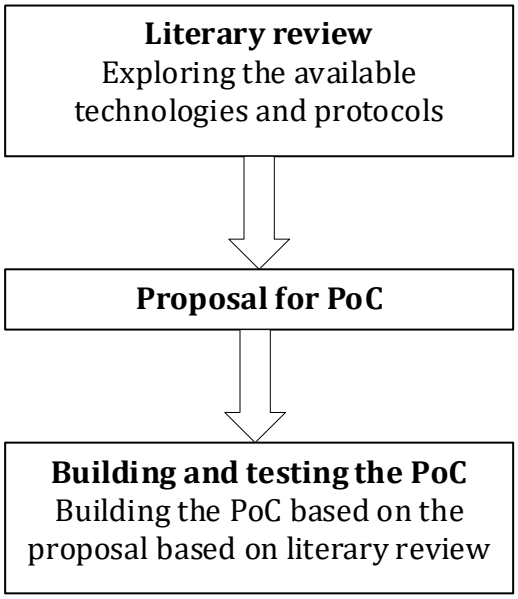

Figure 1. Research Design

The standards and protocols section introduces the terminology, technologies and protocols that are required to serve as an introduction to the relevant topics for a general reader with a technical background. Furthermore, they explore the technologies, protocols and products that are available to solve the outlined problem.

Based on the features and qualities of the technologies that are introduced, an argument is made in section 5 regarding the choice of the technology stack that is utilized for the final delivered proof of concept. The proof-of-concept setup is then built, described, and evaluated based on the proposal, which in turn is based on decisions derived from the evaluation of alternative technologies outlined in the literary review.

2.2 Study Methodology

The methodology of this study can be thought of as following a mixed-methods approach, as the proposal can be said to be a qualitative result of the literary review, and further, the evaluation of the finished proof of concept combines both quantitative measurement results and qualitative conclusions.

According to Creswell and Creswell [3 pp 549], in a mixed methods research,

> The researcher:
>
> • Collects two types of data—qualitative (open-ended) and quantitative (closed-ended) data—to study the research problem or question
>
> • Connects the two data sets, called integration
>
> • Uses procedures to integrate the two data sets, called a mixed methods design
>
> • Analyzes integration typically in a table, called a joint display, by representing the two databases together in the procedures
>
> • Draws conclusions, insight, or interpretation from the integration analysis, called metainferences
>
> • Frames the study with the researcher’s beliefs, values (worldview), and explanations drawn from the literature (theories).

In this study, the literary review serves as the open-ended data collection step, and test results from the PoC are the closed-ended data collection step. The middle step where the PoC proposal is made based on decisions drawn upon the literary review is then clearly the integration step. Any comparison between what is specified in the standards and specifications and how the devices and software behave could be thought of as integration analysis. After all, there are often differences in specification and implementation.

Regarding the last point of framing the study, an important guiding principle in this study is the adherence to solutions that are described in publicly available standards from organizations such as the 3GPP, Institute of Electrical and Electronics Engineers (IEEE), IETF, International Organization for Standardization (ISO), and so on. This principle stems partially from it being clearer from an IPR (Intellectual Property Rights) standpoint to write about topics that are documented in publicly available standards. Secondly, it reflects my personal perspective that standardized solutions often lead to more consistent, interoperable, and long-lasting outcomes. While proprietary solutions may offer short-term advantages, the long-term utility and widespread applicability often reside in implementing and enhancing standardized protocols and technologies.

3 Standards and Protocols

To be able to propose an architecture for the PoC, it's essential to first understand the technological building blocks and frameworks used in IP-based computer networking in general, and how packet delivery works in the 3GPP 5G system. This section provides an exploration of the key standards and protocols that serve as the underpinning for our PoC architecture.

Section 3.1 begins with the OSI Model for Networking, that is fundamental for understanding the layered concept of how different network protocols interact and function in relation to each other.

Section 3.2 delves into the specifics of Ethernet, a key data link protocol. Ethernet is the dominant local area networking technology that provides the basic structure for a large majority of physical networks.

Section 3.3 explores IP and its associated transport protocols, Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

Building on top of the information in sections 3.1 – 3.3, section 3.4 dissects the process of IP packet delivery in the 3GPP 4G and 5G systems, which is critical knowledge for understanding how the proposed solution will operate in a mobile packet-data network environment.

Section 3.5 introduces some of the most common Layer 2 Tunnelling protocols, which provide the means of transporting Ethernet frames on top of IP networks. And finally, section 3.5 investigates Ethernet VPN (EVPN) solutions that provide a control plane mechanism for the dynamic configuration of Layer 2 tunnels.

3.1 The OSI Model for Networking

To manage the complexity of communication systems, various architectural models have been developed over the years. These models compartmentalize pieces of networking functionality into smaller, more manageable parts, often by dividing functionality into layers.

A commonly utilized layer model for networking is the Open Systems Interconnection (OSI) model, a seven-layer model defined by the International Organization for Standardization (ISO). [4 pp 8] The layers of the OSI model are described in Figure 2.

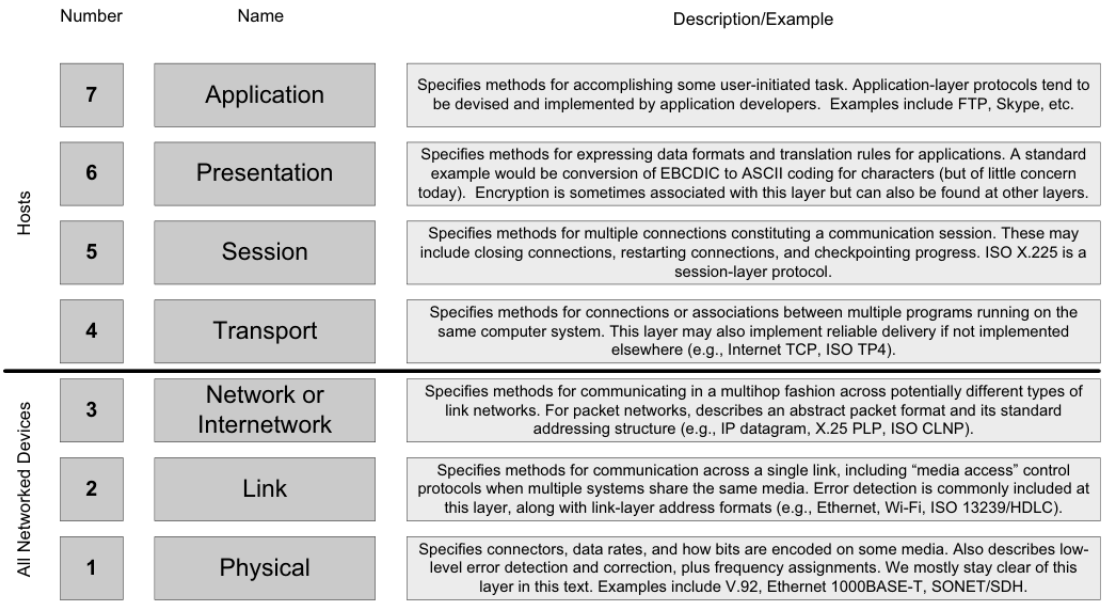

Figure 2. OSI Model Layers [4 pp 9]

Layer 1 defines the physical methods for transferring information over a communications medium. Standards such as the various Ethernet physical standards such as 10GBASE-T, 1000BASE-T, and the physical layers of wireless networking standards, such as Wi-Fi, and the 3GPP standards for cellular network radio access, reside on layer 1. [4 pp 9]

Layer 2, also known as the Link or Data Link layer, specifies protocols for communication with a peer that shares the same physical medium. Conversely, Layer 3, the Network or Internetwork layer, facilitates communication between networks utilizing different transport mediums. The most frequently utilized protocols on Layer 3 are IPv4 (IP version 4) and IPv6 (IP version 6). [4 pp 9]

Layer 4, the transport layer, provides services to the layers above it, such as reliable delivery and multiplexing for different services residing on the same IP host. The most common Layer 4 protocols utilized are TCP and UDP. [4 pp 10]

In IP-based networking, Layers 5 and 6, as specified by the OSI model — the Session and Presentation layers — do not form distinct protocol layers. Instead, their functionality, if needed, is provided by the Layer 7, or Application protocol. [4 pp 10]

There is a wide variety of Layer 7 protocols for various use cases – such as the Hypertext Transfer Protocol (HTTP) used for transmitting websites, File Transfer Protocol (FTP) for file transfers, Secure Shell Protocol (SSH) for secure remote access, and so forth.

3.2 Ethernet

Ethernet, initially developed by Xerox Palo Alto Research Center in the 1970s has endured the evolving landscape of networking to remain a vital component of modern-day communication. [5 pp 119] Originally used for local network communication, Ethernet has since evolved beyond LANs and found widespread adoption in Metropolitan Area Networks (MANs) and even Wide Area Networks (WANs). This has led to the gradual phasing out of legacy technologies such as Frame Relay and Asynchronous Transfer Mode (ATM) in many network infrastructures [6].

Historically, Ethernet began as a shared medium, using a coaxial cable to interconnect devices in a bus topology. The fundamental technology behind Ethernet is known as Carrier Sense, Multiple Access with Collision Detect (CSMA/CD). This method enables multiple devices to share the same network medium by listening to the network before transmitting and detecting if a collision occurs.

While shared-medium Ethernet marked a significant advancement in terms of speed and reliability over previous network technologies, its limitations became particularly noticeable in large and congested networks. To mitigate this, Ethernet evolved to utilize twisted pair or fibre optic cables in a star topology, where each device has a dedicated connection to a network switch, creating individual collision domains. [5 pp 119] As a result, collisions have become an extremely rare occurrence in today's switched Ethernet networks.

At its core, Ethernet operates at the Data Link layer (Layer 2) of the OSI model. It provides a means for nodes in a network to communicate directly with each other. A unique Media Access Control (MAC) address identifies each network interface, serving as the hardware-based address used for communication within a network segment. Ethernet frames, the basic unit of communication in an Ethernet network, encapsulate the payload data. They also contain addressing information that allows the data to be routed and delivered properly. [5 pp 122]

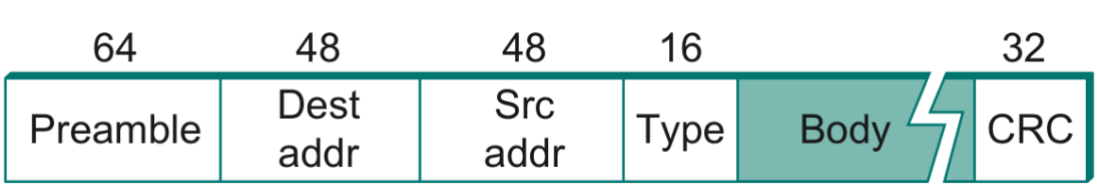

Figure 3. Basic Ethernet frame structure [5 pp 123]

The basic Ethernet frame structure is outlined in Figure 3. The frame begins with a preamble, followed by destination and source MAC addresses. The type/length field is used to indicate either the type of payload protocol or the size of the payload. The payload itself, which is the data being transferred, follows this field. Lastly, the Frame Check Sequence (FCS) is used for error detection. The FCS is a 32-bit Cyclic Redundancy Check (CRC) [5 pp 122-123].

There are additional standards that extend the basic IEEE 802.3 Ethernet Frame Format. For instance, if the type field is set to 0x8100, this indicates that the frame is tagged with a 802.1p/q tag. [4 pp 85] The format of an 802.1q compatible frame is illustrated in Figure 4.

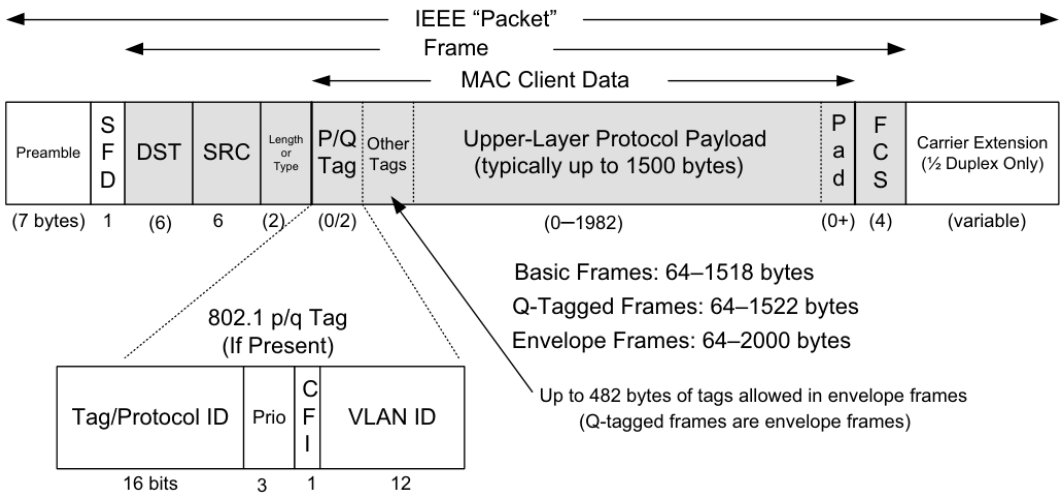

Figure 4. 802.1q Ethernet frame format [4 pp 85]

The purpose of the Virtual LAN (VLAN) tag is to allow a single shared physical LAN to be partitioned into separate logical (or, virtual) LANs. When VLAN tags are used, VLAN-aware switches will accept ingress frames with a particular VLAN tag only from ports that are configured to accept that VLAN tag, and consecutively will forward them as egress frames only out of ports that have the VLAN tag configured on them – of course, with the other normal Ethernet switch forwarding rules applying as well, such as once a MAC address has been learned, only forwarding to the port where that MAC exists. [5 pp 201-202]

3.3 IP and Transport Protocols

To understand the mechanisms and operation of Layer 2 Ethernet tunnelling protocols, it is crucial first to understand the underlying network and transport protocols which they are built upon.

IPv4 is the currently the primarily used OSI layer 3, i.e., network layer protocol of the Internet Protocol Suite. IP provides a mechanism to uniquely identify hosts and route data between them across one or more networks. [5 pp 206]

IPv4 addresses are 32 bits long and are typically represented as four decimal numbers separated by periods (e.g., 192.168.0.1). [5 pp 215] Each packet sent over IP includes a header, which contains important routing and handling instructions. Some key fields include the source and destination IP addresses, the version of IP in use, the packet length, and a field for identifying the transport protocol in use. [5 pp 207-210] The IPv4 header format is illustrated in figure 5.

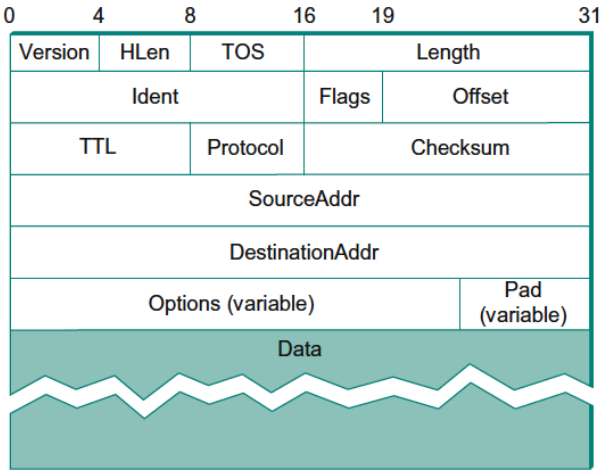

Figure 5. IPv4 header format [5 pp 207]

While IP operates on layer 3 of the OSI model, transport protocols operate on layer 4, providing end-to-end communication services for applications. The most used are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). TCP offers a reliable connection-oriented service, guaranteeing that data sent from one end reaches the other end without errors and in the same order. [9 pp 200] UDP, on the other hand, provides a connectionless service, sacrificing guaranteed delivery for lower overhead and latency. [4 pp 15]

The way TCP has been designed for reliability has its drawbacks, however. In tunnelling setups where TCP is used as both the transport protocol for the tunnel (outer TCP) and the protocol encapsulated within the tunnel (inner TCP), this will lead to a condition called “TCP over TCP”. If packet loss occurs, both the inner and outer TCP connections respond by invoking their congestion control mechanisms, which can result in a drastic overall decline in throughput, known as a "TCP meltdown" [7]. To prevent this situation, protocols for layer 2 tunnelling are most often designed to use non-TCP protocols, such as UDP, for the transport layer of the tunnel.

Since this thesis mainly deals with protocols that use UDP as their transport protocol, it's worth examining the UDP header a bit further. Figure 6 provides an illustration of the UDP header and its various fields.

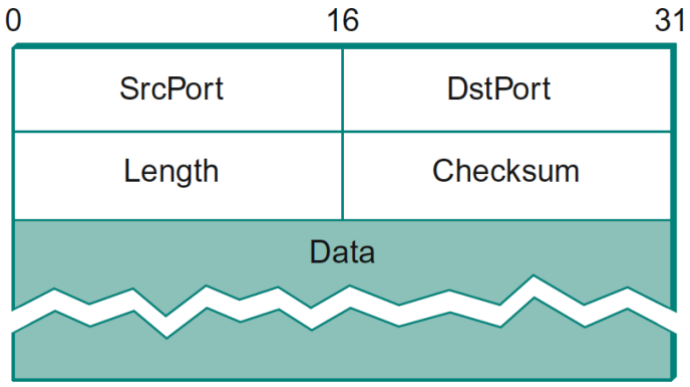

Figure 6. UDP header format [5 pp 394]

The *Source Port* field identifies the source port number, corresponding the application, process, or session inside an application that sent the UDP datagram on the source machine. Together with the *Destination Port,* which identifies the application or process on the destination machine, the ports allow multiplexing and de-multiplexing of communications within a single network layer channel. This means that two machines can simultaneously have multiple streams of communication between themselves, and the combination, or *tuple*, of source and destination ports identifies which packets belong to which communication stream. [5 pp 393-395]

The Length field in the UDP header specifies the length of the entire datagram, including both the header and the data. The *Checksum* field can be used by the receiver to verify the correctness of the received datagram. The checksum is optional in IPv4 but mandatory in IPv6. [5 pp 395]

3.3.1 Overview of IP Routing

For IP connectivity to work between hosts that are not directly in the same layer 2 broadcast domain and IP subnet, IP routing must be utilized. This means that when traffic must be able to travel between two distinct subnets, a router must be placed in between, with connectivity to both subnets, and the required routing information stored in the routing table. The end hosts can suffice with a single piece of routing information, called the *default route*, towards the routing gateway, but the router itself might have a variety of routing information stored in its routing table. [8 pp 8]

When the routing tables start getting more complex and there are multiple routers to keep in sync with each other, it becomes increasingly difficult to maintain the routing table by entering static routing information by hand into each router. Thus, various routing protocols have been developed to facilitate exchange of routing information between routes in a more dynamic fashion. [8 pp 9]

Routing protocols can be roughly categorized into three groups. The simplest routing protocols are so-called *distance vector* protocols, where the only information distributed is the next-hop to a certain destination, and the number of routing hops required to get to the destination. Routing Information Protocol (RIP) is an example of a distance vector protocol. When RIP is used, the routers broadcast their routing table over every connection, and listen to the same information to be received from other routers. Routes received over RIP are then added to the routing table, and broadcast along with other routes known by the router. The hop count increases the further the routers get from the network, and the best route is selected based on the shortest hop count. [8 pp 9-10]

Simple distance vector protocols like RIP can cause certain issues, such as convergence problems caused by the periodic flooding of all routing information by all routers, and not considering the fact that different physical links can have different properties that make them more or less desirable, such as varying link speeds. The second category of routing protocols are *link state protocols*, such as Open Shortest Path First (OSPF). OSPF uses a more complex route

preference algorithm that also considers the cost (i.e. usually the bandwidth) of the link. However, as the Dijkstra algorithm utilized in OSPF is rather complex, route updates are only sent when something changes – such as when a link goes down or comes back up. OSPF routers keep a map of the network topology, and re-compute the map when something changes and an update is transmitted. [8 pp 10]

RIP and OSPF belong to a set of protocols called Interior Gateway Protocols (IGP's). This means that they are meant to operate under the same administration – inside what in BGP terms is called an Autonomous System (AS), among a fairly limited number of routers, and the connectivity is relatively homogenous and doesn't span vast distances. In the case of routing between AS's on the Internet, however, it's not feasible to periodically broadcast all routing information, or keep a complete topology map of the whole network. Thus, for routing between organizations, protocols in a family called Exterior Gateway Protocols (EGP's) are used. BGP version 4, defined in RFC 1771, is the dominant EGP protocol in use today. [8 pp 10]

BGP is a so-called *distance-path protocol*. This means that while it doesn't use just the shortest hop count like RIP, it also doesn't keep track of the complete network topology like OSPF either. Instead, in BGP, reachability information for routes is received from neighbouring routers, and the route to a destination with the shortest path is then included into the routing table, if the routing policy permits it. The path in BGP is a list of AS's between the router and the destination. [8 pp 10]. Section 3.6 investigates BGP itself, and EVPN solutions based on BGP, in more detail.

3.4 IP Packet Delivery in the 3GPP LTE and 5G Systems

Private cellular networks aim to bring the benefits of mobile communication into specific enterprise or localized environments. They heavily leverage standards and technologies defined by the 3GPP. Currently, the two most widely deployed network architectures are Long-Term Evolution (LTE), also known as 4G, and 5G. For the purposes of this thesis, understanding the mechanics of IP packet delivery within these frameworks is crucial.

Subsection 3.4.1 delves into the LTE system, dissecting the components and procedures integral to IP packet transportation. Following this, subsection 3.4.2 focuses on the 5G system, highlighting the evolutionary steps and innovative features that shape its IP packet delivery pathways.

3.4.1 Packet Delivery in LTE

Second generation and earlier mobile networks – such as Global System for Mobile Communications (GSM) – were originally designed for carrying voice traffic only. The ability to transmit packet data directly from the Mobile Station (MS) was a later to GSM in the form of General Packet Radio Service (GPRS), which was further enhanced in Enhanced Data Rates for GSM Evolution (EDGE). In 3rd generation Universal Mobile Telecommunications System (UMTS) standards, packet data capabilities were introduced from the onset. The introduction of High-Speed Downlink Packet Access (HSDPA) resulted in a considerable increase of packet data subscribers and traffic. In Finland, the packet data volume in 3G networks exceeded voice traffic in 2008, and in 2009 there was already a 10-fold amount of packet data being transmitted compared to voice. [9 pp 1]

Compared to 3G systems, Long-Term-Evolution (LTE), also called the 4th Generation Mobile Network (4G), is optimized to provide higher throughput, lower latency, and an overall simplified architecture for packet data delivery [9 pp 23]. In fact, contrary to earlier networks, there are no circuit switched elements at all in the 4G Evolved Packet Core (EPC). Connections to legacy circuit-switched

networks such as Public Switched Telephone Network (PSTN) or Integrated Services Digital Network (ISDN) may be provided, for example, by the IP Multimedia Subsystem (IMS) via distinct Media Gateways. [8 pp 26]

The LTE network functions, and other elements involved in packet delivery between the User Equipment (UE) and external networks in the simplest possible network configuration are illustrated in figure 7.

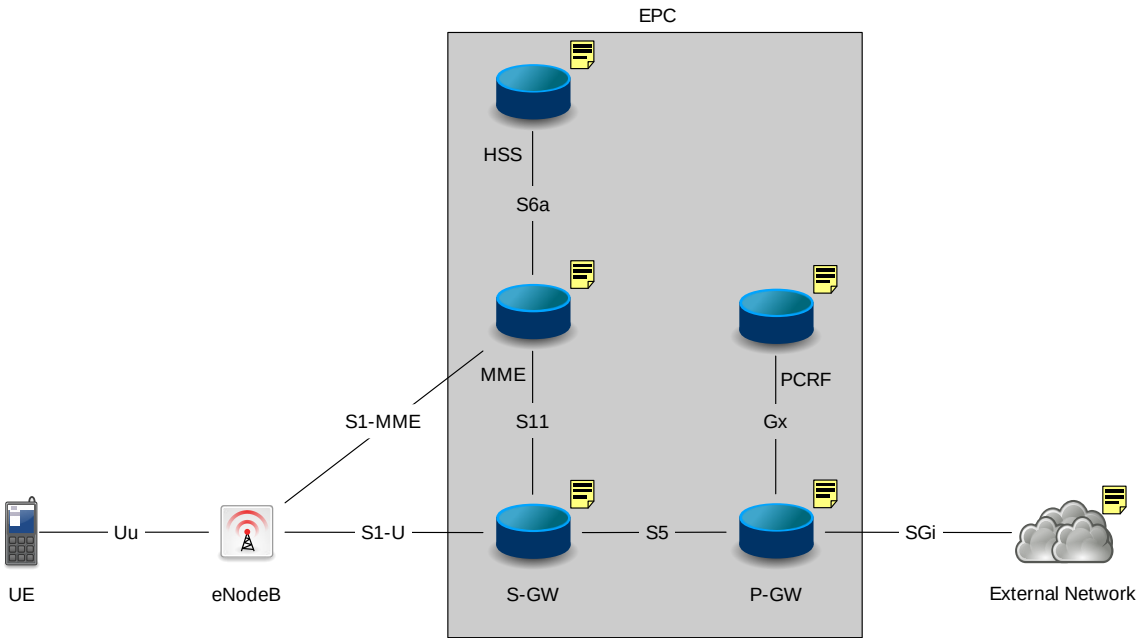

Figure 7. Packet Delivery architecture in LTE [9 pp 25]

The User Equipment (UE) is a device that's used by the end user for communication. Besides handheld devices such as smartphones or tablets and wearables such as smart watches, the UE can also be a modem card embedded into a router or a laptop, for example. The UE has a USIM (Universal Subscriber Identity Module) that is used to identify and authenticate the user, and to derive security keys used for encrypting the radio transmission. [9 pp 26-27]

The Home Subscription Server (HSS) contains information about the UE, such as which services are allowed for the particular subscriber. The HSS also contains the Authentication Center (AuC) functionality, which is used on the network side to derive the authentication vectors and encryption keys required for secure communication [9 pp 34]

The UE connects wirelessly to a radio base station over an interface called Uu. The radio base station in LTE is called the Evolved Universal Terrestrial Radio Access Network (E-UTRAN) Node B (eNodeB). The eNodeB plays many important roles in the LTE network – however when looking at simply the perspective of the user data plane, its main function is to relay data it receives from the UE over the Uu interface towards the Serving Gateway (S-GW) over the S1-U interface, and vice versa. Furthermore, the eNodeB is used to transmit and receive control-plane signalling information in the form of Radio Resource Control (RRC) messages, and relays Non-Access-Stratum (NAS) messages containing control and signalling information between the Mobility Management Entity (MME) and the UE. This exchange of signalling information between the eNodeB and MME happens over the S1-MME interface. [9 pp 25]

The Serving Gateway (S-GW) relays traffic between the eNodeB and the Packet Gateway (P-GW). When GPRS Tunneling Protocol (GTP) is used on the S1 and S5 interfaces, and the UE is in connected mode, the S-GW just relays GTP packets between the eNodeB and the P-GW. If a UE is in idle mode, the eNodeB has no radio resources allocated to it. If in such a case the S-GW receives downlink data packets from the P-GW that belong to a UE in idle mode, the S-GW will buffer the data, and request the MME to initiate paging for the UE, causing it to re-connect and be able to receive the buffered packets.

The S-GW also acts as the local mobility anchor point when UE is moving from one eNodeB to another, being instructed by the MME to switch the GTP tunnel from the old eNodeB, where the UE is moving away from, to the new one, where the UE is moving to. It's also possible for an UE to move from one S-GW to another, or a single S-GW to relay data towards multiple P-GW's. [9 pp 30-31]

The primary function of the Packet Gateway (P-GW) is to route traffic between the mobile network and external networks. The P-GW allocates the IP addresses for the UE's and sets up the GTP tunnels which represent the data bearers for the connected UE's. Thus, as a simplification, the P-GW can be often thought of acting as an IP router between the external IP network, and the IP address pool that the UE's utilize. The P-GW also includes the PCEF (Policy and Charging Enforcement Function), which performs traffic gating and filtering functions, according to either statically configured rules, or rules supplied dynamically from the PCRF (Policy and Charging Rules Function).

One downside of the "classical" LTE architecture is that the network functions have mixed responsibilities in handling control- and user-plane tasks. For example, the P-GW is responsible for both the IP allocation, which can be thought of as a control-plane task, and the actual routing and forwarding of the user-plane traffic. Thus, in release 14 of the 3GPP standards, the Control and User Plane Separation (CUPS) specification was introduced, which splits the P-GW into PGW-C and PGW-U, S-GW into SGW-C and SGW-U, and the Traffic Detection Function (TDF), not discussed here, into TDF-C and TDF-U. This split of functionalities enables more flexible deployment models of the packet core functionality. For example, the control-plane nodes can be hosted in a cloud platform for enhanced scalability, but the user-plane nodes can be brought closer to the customer for lower latency. [10]

In summary, the LTE network architecture is designed for effective transmission of IP traffic between UE's and external networks connected to the packet core over E-UTRAN wireless connectivity. The architecture has further been evolved by subsequent changes, e.g., the Control and User Plane Separation (CUPS) introduced in 3GPP release 14. This separation of control-plane and user-plane is a key element in the 5G architecture, which we will look at in the next chapter.

3.4.2 Packet Delivery in the 5G System

The 5th generation 3GPP standards for mobile networks were introduced in release 15 of the standards, the first full release of which was completed in December 2018 [11 pp 8]. Compared to 4G and earlier standards, 5G brings many enhancements, such as new spectrum ranging from 400 MHz all the way up to 90 GHz, beamforming capabilities that are useful especially in the higher frequency bands, dual connectivity and coexistence with LTE, and so on. [11 pp 3-5] As in the previous chapter, here we'll only focus on the IP packet delivery aspects of the 5G system.

In 5G, the core network architecture is often described using a service-based approach, in which the communication between the control plane network functions (NFs) is decoupled from the network interconnections and described using a shared access bus-like topology. [11 pp 71-72] However, the network functions and their interconnections can also be described in a similar reference point-based architecture with point-to-point connections as is done in 4G and earlier generations of mobile networks. The functions involved in IP packet delivery in the simplest possible topology of a 5G system are illustrated using the reference point architecture in figure 8.

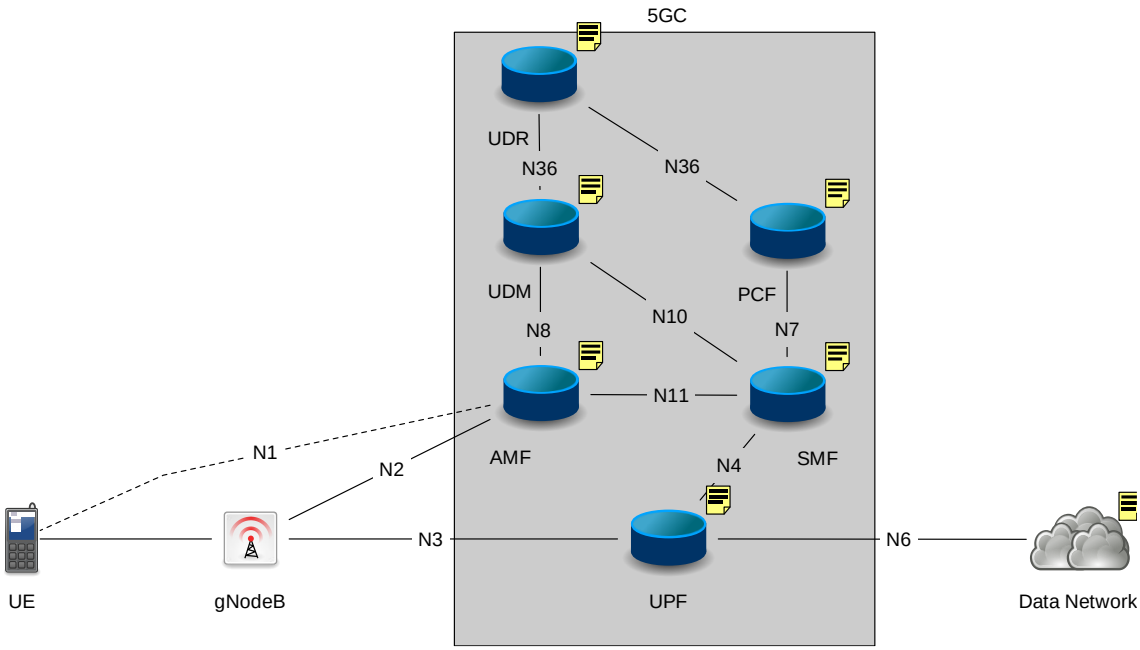

Figure 8. Packet delivery in 5G [12 pp 25]

The User Equipment (UE) in 5G, performs similar functionality as in 4G. It is the end device as used by the network user and contains the USIM module used for authentication.

In 5G, the subscription data, among other standardized structured data, such as policy data, is contained in the User Data Repository (UDR). The Unified Data Management (UDM) function handles the application logic for processing information in the UDM, such as generation of authentication credentials and handling user and subscription identity. The UDM communicates with the UDR over the N36 interface. [11 pp 73]

In 5G, the radio access node is called gNodeB. In 5G, the radio access technologies are typically referred to as New Radio (NR). From the point of view of packet delivery, the gNodeB in 5G serves very similar functionality as the eNodeB in 4G. It communicates with the UE over the air interface, with the Access and Mobility Management Function (AMF) over the N2 interface, and with the User Plane Function (UPF) over the N3 interface. [11 pp 71-72]

The Access and Mobility Management Function (AMF) act as the termination point for the Non-Access Stratum (NAS) signalling between the network and the UE and provides ciphering and integrity protection for the NAS signalling. It also terminates the control plane signalling for the radio access network. Furthermore, the AMF acts as a proxy for session management information between the Session Management Function (SMF) and the UE. [11 pp 72]

The Session Management function takes care of establishment, modification, and release of user sessions. It provides functionality such as UPF selection and IP address allocation, and interfaces with other functions such as the Policy and Charging Function (PCF) to determine the appropriate policy rules for packet delivery. [11 pp 73] Contrasting with 4G – especially without CUPS, there is a clear separation in 5G between functionality related to controlling the user data sessions, which happens in the SMF, and actual delivery of the user-plane data, which happens in the User-Plane Function (UPF).

The User Plane function handles the relaying of user plane data packets between the UE (via the gNodeB) and the Data Network (DN). Furthermore, it performs the actual application of policies such as Quality-of-Service (QoS) marking, packet inspection, and either allowing or dis-allowing traffic, as configured by either static rules in the UPF, or instructed dynamically either by the SMF or the PCF via SMF. The UPF also acts as the mobility anchoring point for the user data tunnels. [11 pp 73] Unlike in 4G, there is no separate network function that correspond to the S-GW – rather, the user plane functionality of the S-GW and P-GW are combined into the UPF as a single network function.

In conclusion, the core network functions in 5G have been designed with the separation of control plane and user plane in mind from the outset. This, combined with the service-based architecture capabilities, makes it easier to enable a variety of flexible core network deployment scenarios, be they fully on-premises, or either partially or fully cloud based. However, as can be seen from this and the previous chapter, the key aspects of the IP packet delivery remain still largely unchanged.

3.5 Layer 2 Ethernet Tunnelling Protocols

As described in section 3.1, layer 2 in the OSI model is the data link layer, which includes protocols that are used for communication between hosts that share the same physical medium. The set of hosts that can communicate with each other using a layer 2 protocol is typically called the broadcast domain. [4 pp 168] And as outlined in section 3.2, Ethernet is a link-layer protocol that operates on layer 2 of the OSI model.

It's still worth noting that while the layer 1 medium itself might change – i.e., Ethernet utilizing copper cables can share a layer 2 broadcast domain with fibre-based Ethernet, the span of a single layer 2 network is still largely limited to a select suite of jointly specified layer 1 and layer 2 technologies. For instance, Digital Subscriber Line (DSL) links are not inherently designed to transmit Ethernet frames. [4 pp 9] Therefore, if an Ethernet broadcast domain needs to span a DSL link, some form of tunnelling is required.

Layer 2 tunnelling protocols define methods for encapsulating Ethernet frames in IP packets. [13 pp 176] Some protocols, such as VXLAN and Multiprotocol Label Switching (MPLS) over UDP, use UDP as the transport protocol for a packet format defined by the protocol [13 pp 176], while other protocols, namely GRE and its derivatives, operate directly on top of IP. [14]

The next subsections explore some of the most common protocols used for Layer 2 Ethernet tunnelling from a purely transport point of view, not including any associated separate control plane protocols, as this discussion is reserved for later.

3.5.1 GRE

Generic Routing Encapsulation (GRE) was developed by Cisco Systems and first specified in IETF RFC 1701 [15] and RFC 1702 [16] in 1994, later further developed into an IETF standards track protocol in RFC 2784 and further extended in RFC 2890 [17].

As the name suggests, GRE is a generic transport protocol for tunnelling / encapsulating packets of a multitude of other protocols to be transmitted in routed IP networks [14]. In addition to IPv4 as the delivery protocol as specified in RFC 1702 [16], support for IPv6 has also been later added as per RFC 7676 [18].

When IPv4 delivery is used, the GRE protocol payload is indicated by the IP protocol type field being set to 47 decimal [16]. When IPv6 delivery is used, a GRE payload is indicated by the Next Header field being set to 47. In IPv6 delivery, the GRE header may immediately follow the delivery IPv6 header, or alternatively, additional extension headers may be utilized between the IPv6 header and the GRE header [18]. The full extended GRE header as defined by RFC 2784 and RFC 2890 is illustrated in Figure 9.

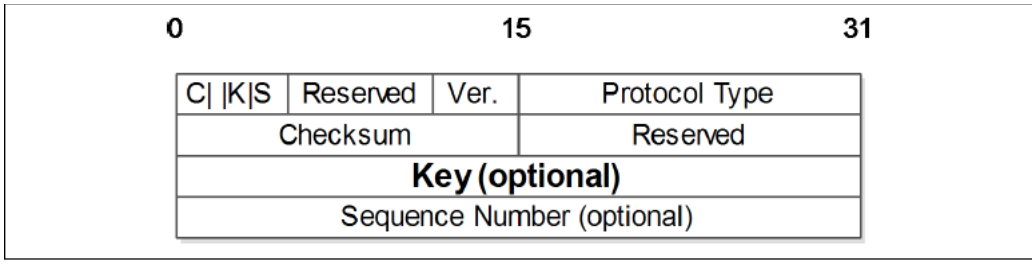

Figure 9. Extended GRE header [13 pp 732]

The first bit of the header at offset 0, *C*, indicates whether the checksum field is present. If C is set to one, then the Checksum field, containing a valid checksum, and the second Reserved field are present in the header. [14]

The second and third bits, *K* and *S,* indicate whether either the Key and/or Sequence Number fields as specified by RFC 2890 are present [17]. These are

then followed by 9 reserved bits, followed by a three-bit version field. The version field is always set to zero [14].

The version field is followed by the *Protocol Type* field, which contains the protocol information of the encapsulated payload packet. The possible protocol types are all valid IEEE 802 Ethertypes [14]. The Internet Assigned Numbers Authority (IANA) maintains a list of the registered Ethertypes [19].

The Protocol Type field is optionally followed by the Checksum field that can be used for integrity verification of the GRE header and the payload packet. If present, the Checksum field is followed by two reserved bytes [14]. Finally, the optional RFC 2890 Key and Sequence number fields follow, if their presence is indicated by the K and/or S bits [17].

When the Protocol Type field is set to the hexadecimal value 6558, this is interpreted as *Transparent Ethernet Switching [15]*. This means that the GRE header is directly followed by an Ethernet header, and the GRE payload is a tunnelled Ethernet frame.

Various networking device manufacturers have implemented Layer 2 GRE tunnelling. While some vendors refer to it as Layer-2 GRE [20], other names such as Ethernet over GRE (EoGRE) are also used [21].

A popular open-source implementation upon which many commercial products are also based is the Linux *gretap* virtual interface driver. Since all the other protocols that follow are supported in a similar fashion, it’s perhaps worth taking a bit of time to look at this implementation and its applications in a bit more depth.

In Linux networking, the Universal TUN/TAP device driver implements virtual network devices that user space programs can use to send and receive packets to and from the kernel networking stack. The difference between *tun* interfaces and *tap* interfaces is that while *tun* can transmit and receive IP packets, *tap* can be utilized to transmit and receive Ethernet frames [22]. Besides Linux, similar functionality exists in other open-source operating systems as well, such as the

Berkeley Software Distribution (BSD)-based free operating systems FreeBSD [23], and NetBSD [24]. So, using a similar naming convention to convey functionality, the *gretap* driver encapsulates and de-encapsulates Layer 2 Ethernet frames inside GRE packets. While *gretap* uses IPv4 for delivery, an equivalent virtual device driver, called *ip6gretap*, exists for IPv6 delivery [25]. Similarly as with the generic *tun* and *tap* interfaces, functionality equivalent to Linux *gretap* exists in other open-source operating systems as well, for instance as the *egre* virtual network interface in OpenBSD. [26]

To be able to transmit data to and from a physical LAN network, the *gretap* interface must be bridged with a physical Ethernet connection somehow. In Linux this can be achieved by using the bridging functionality built into the kernel. The Linux bridge implements a subset of the IEEE 802.1d bridging standard but is more flexible than a hardware bridge because it can also be used to filter and shape traffic. Additionally, the Linux bridge can also bridge frames between physical Ethernet devices and virtual devices such as the *gretap* virtual device. Figure 10 illustrates a use-case where some different physical and virtual devices are connected by a Linux bridge.

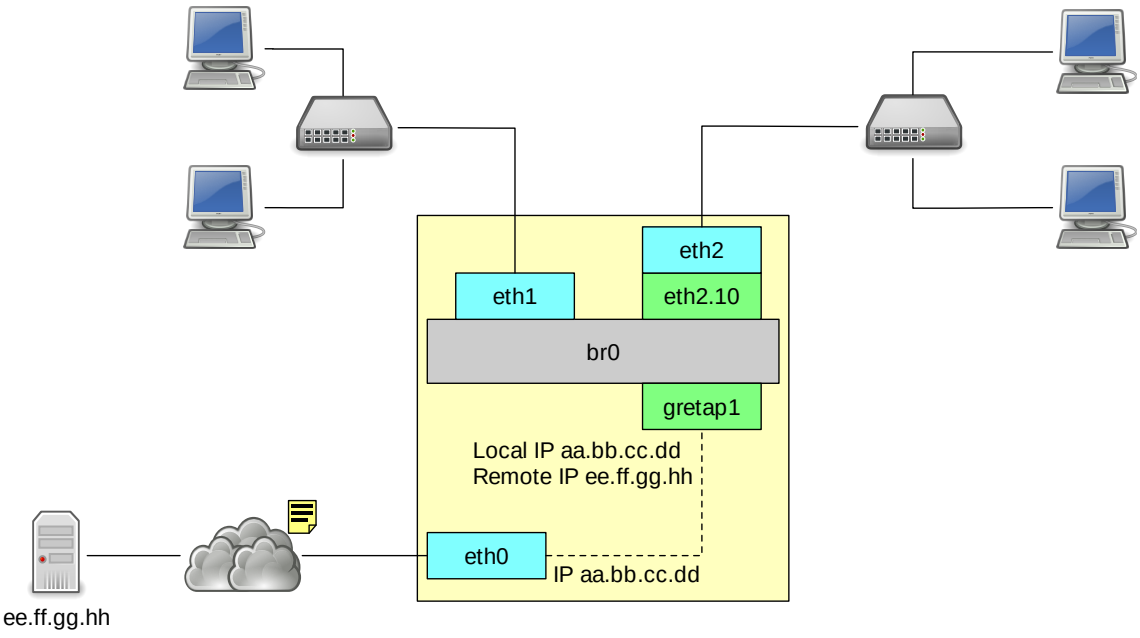

Figure 10. Example Linux bridge setup

In the example depicted above, the contents of the yellow box are within a system running a Linux kernel. *br0* is a Linux virtual bridge, and it has the following interfaces connected to it:

- *eth1*, which is a physical network device
- *eth2.10*, which is a VLAN pseudo device, providing access to VLAN 10 on the physical network device *eth2*
- *gretap1*, which is a virtual *gretap* device with the local IP address *aa.bb.cc.dd*, assigned to *eth0*, and a remote IP address of *ee.ff.gg.hh*.

In this setup, GRE packets with protocol type of 6558 (hex) arriving from *ee.ff.gg.hh* are de-encapsulated and the inner Ethernet frames are sent to other ports on the *br0* bridge, according to the standard Ethernet bridging/switching conventions. Conversely, Ethernet frames deemed to be sent using *gretap1* as the egress port are encapsulated in GRE headers and sent towards the host *ee.ff.gg.hh*.

An alternative software approach to using GRE tunnelling for Ethernet frames is the Open vSwitch (OvS) software switch. Compared to the standard Linux bridge implementation, OvS adds Software Defined Networking (SDN)-related functionality, such as the possibility of automated control via OpenFlow or other management protocols. [27]

One immediately apparent shortcoming of using standard GRE for layer 2 tunnelling is that there can be only one tunnel between a host pair, as there is no tunnel identifier field in the header. One way this issue can be somewhat mitigated is to send VLAN-tagged frames inside the tunnel, which can then be bridged into separate virtual bridges on the receiving side – but this method is somewhat inflexible – for instance, to be able to identify a virtual network, one needs to look inside the tunnelled payload to figure out the VLAN tag, instead of just inspecting the GRE header.

An approach to this problem is outlined in RFC 7637, Network Virtualization Using Generic Routing Encapsulation (NVGRE). In NVGRE, the 32-bit Key field of the GRE header as defined by RFC 2890 is used to carry a 24-bit Virtual Subnet ID (VSID) and an optional 8-bit Flow Identifier. The 24-bit VSID field allows for up to 16 million distinct virtual subnets to be transmitted between one host pair. [28] Similarly to plain GRE, open-source implementations of NVGRE exist as well, at least in OpenBSD as the *nvgre* pseudo-device. [26]

A second important question is what to do with so called Broadcast, Unknown and Multicast (BUM) frames. The standard Ethernet behaviour is to flood frames with Broadcast or Multicast destination addresses and frames for which the recipient port hasn't been learned into each individual port. [5 pp 190-192] When the plain *gretap* implementation is used, this means that each BUM frame is duplicated for each individual GRE tunnel and sent separately to each tunnel endpoint. NVGRE proposes an alternative approach, where the BUM frames would be sent to a multicast IP address that's unique to the virtual network, thus removing the need for per-endpoint duplication in multicast-capable networks. [28]

Another possible issue with GRE is that since it's not based on UDP or TCP, it generally cannot be utilized with Network Address Port Translation (NAPT) as defined in RFC 2663. [29] Also called NAT masquerade or One-to-Many NAT, this method allows the use of a single public IP to be shared by multiple private IP's, and is a common technique used to mitigate the IPv4 address exhaustion problem. But since NAPT typically works only with TCP, UDP, and ICMP, special software would be needed for NAPT traversal of GRE, which generally doesn't exist – or at least it's existence can't be relied upon.

3.5.2 MPLS

Multiprotocol Label Switching (MPLS) was developed around the turn of the millennium to simplify and optimize routing and traffic engineering in packet-switched WAN operator core networks [30 pp 6]. The primary IETF RFC's defining the MPLS architecture are RFC 3031, Multiprotocol Label Switching Architecture, and RFC 3032, MPLS Label Stack Encoding. These have later been further amended by various other RFC's [31], [32].

As the name suggests, the label switching in MPLS is based on labelling packets and then choosing the appropriate path for them based on those labels. The MPLS label is 32 bits long, and its structure is illustrated in Figure 11.

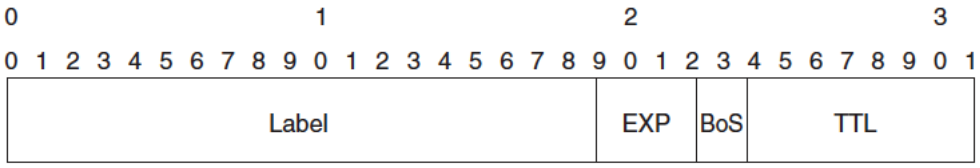

Figure 11. MPLS label format [30 pp 26]

The first 20 bits of the MPLS label are the actual label value. The following 3 bits are "Experimental" (EXP) bits, used for Quality of Service (QoS) information. Bit 23 is the Bottom of Stack (BoS) bit. In MPLS, labels can be stacked, that is, a packet can contain multiple labels that will dictate the path the packet should take in the label-switched network. If the BoS bit is 0, this means that the current label is not the last label in the stack, and more labels will follow – until a label with the BoS bit set to 1 is encountered, which is the last label in the stack. [30 pp 26]

MPLS can be utilized over various Layer 2 data-link protocols, such as Ethernet, Point-To-Point Protocol (PPP), and High-Level Data Link Control (HDLC), to name a few. As detailed in Figure 12, the MPLS label stack is placed between the layer 2 header and the transported payload – thus the MPLS label stack is sometimes called a *shim header*. This also means that a distinct protocol identifier must be used in the layer 2 protocol to indicate that after the layer 2 header, an MPLS label stack follows. In the case of Ethernet, the Ethertype value indicating the MPLS protocol is 8847 hexadecimal. [30 pp 27]

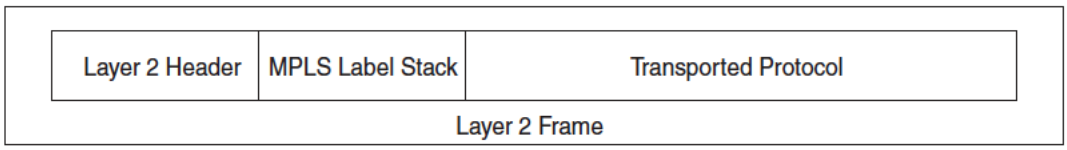

Figure 12. MPLS labelled packet inside a layer 2 frame [30 pp 27]

On the payload side of things, MPLS can be used to transport various protocols, such as IPV4, IPV6, Ethernet, HDLC and PPP. [30 pp 7] The transport aspects of encapsulating Ethernet frames in MPLS packets is defined in RFC 4448, *Encapsulation Methods for Transport of Ethernet over MPLS Networks*. [33] Since RFC 4448 and other similar technologies only allow for a point-to-point link to be defined – leading to similar scalability problems as the static configuration of IP-based tunnelling technologies, the Ethernet encapsulation is typically used as part of a service commonly referred to as Virtual Private Lan Service (VPLS).

As MPLS operates directly on top of the layer 2 data link protocol, a network where MPLS is transmitted generally must support MPLS end-to-end. In cases where MPLS needs to be transmitted over non-MPLS-enabled networks, it needs to be encapsulated somehow. There are multiple methods for encapsulating MPLS in IP packets. RFC 4023, *Encapsulating MPLS in IP or Generic Routing Encapsulation (GRE),* defines methods for encapsulating MPLS in IP packets, and for encapsulation within GRE tunnels over IP. Furthermore, RFC 4817 defines encapsulation of MPLS over Layer 2 Tunneling Protocol Version 3 [34], and RFC 7510 defines encapsulation of MPLS over UDP [35].

In MPLS over IP, the protocol number field in IPv4, or the next header field in IPv6, is set to 137, which indicates an MPLS unicast packet. In MPLS over GRE, the protocol type field of the GRE header is set to hexadecimal 8847 for MPLS unicast, or 8848 for MPLS multicast. In the methods defined in RFC 4023, the IP and/or GRE headers can replace the topmost label in the label stack – as two of the Label Switching Routers (LSR's) are replaced by the routers handling the encapsulation and de-encapsulation [36].

In MPLS over L2TPv3, the MPLS label stack and payload are directly encapsulated in an L2TPv3 header. [34] And finally, in MPLS over UDP, the MPLS label stack and payload are encapsulated in an IP header, followed by a UDP header with the destination port of 6635, which is the port number allocated by IANA to MPLS. [35]

Open-source implementations exist for MPLS as well. For example, in Linux, it's possible to work with MPLS labels by using the Traffic Control (TC) tools and encapsulate / de-encapsulate Ethernet frames in MPLS labels [37]. This can be then combined with the virtual bridge subsystem, as described in subsection 3.5.1, to achieve more complicated tunnelling / bridging setups.

3.5.3 L2TPv3

Layer Two Tunneling Protocol, version 3 (L2TPv3) is specified in RFC 3931 and is based upon the earlier L2TP version 2 (L2TPv2) specified in RFC 2661. [38] While L2TPv2 is used to extend PPP connections over IP – that is, the tunnelling protocol essentially replaces the original layer 2 circuit [39], L2TPv3 is a more generic protocol that can be used to tunnel various types of layer 2 payloads [38].

The L2TP RFCs describe both the format for the data packets and also for control packets. The control packets are used to establish and control an L2TPv3 session. However, it's not mandatory to use the L2TPv3 control protocol, and thus L2TPv3 can be used in a similar fashion to the other described tunnelling protocols. [38] As the L2TPv3 control protocol isn't ultimately of interest in the scope of this thesis, thus it's not discussed here in further detail.

The L2TPv3 data packets, also called session packets, can be transmitted over IP or UDP. When L2TPv3 is used over IP, the IPv4 protocol field is set to 115 decimal. Over IP, the L2TPv3 session header consists merely of a 32-bit session ID that must be zero for control packets and non-zero for session packets, and a variable-length cookie field that's up to 64 bits long, as illustrated by Figure 13. The session ID is used to distinguish multiple tunnels between the same host pair – as was explained previously, standard GRE lacks this functionality. The Cookie field can be further used to ensure that the correct packets end up in the correct tunnel. If the Cookie field is utilized, it is typically set to a random value dictated by the control protocol. [38]

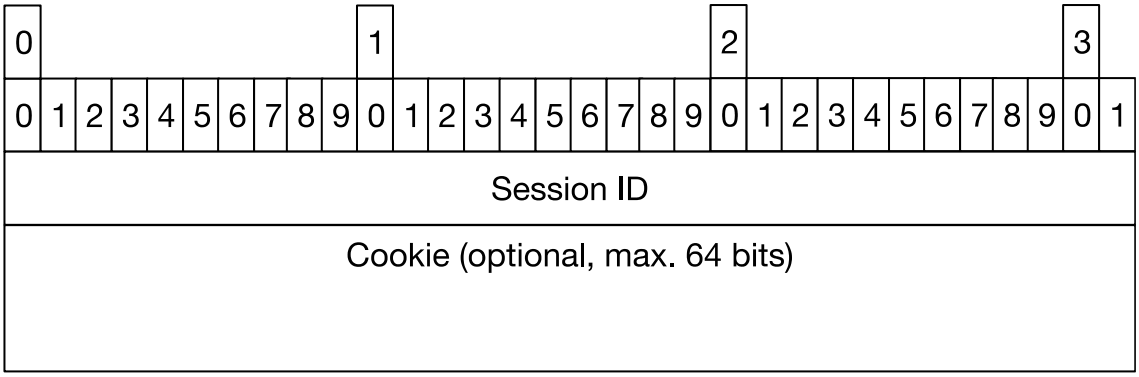

Figure 13. L2TPv3 session header over IP [38]

As stated in RFC 3931, L2TPv3 over UDP can be utilized for better NAT and Firewall traversal compared to L2TPv3 over IP. When utilized over UDP, the session header is different compared to purely IP-based transport, as L2TPv3 over UDP must be able to co-exist with L2TPv2 on the same UDP ports. [38] The L2TPv3 session header over UDP is illustrated in Figure 14.

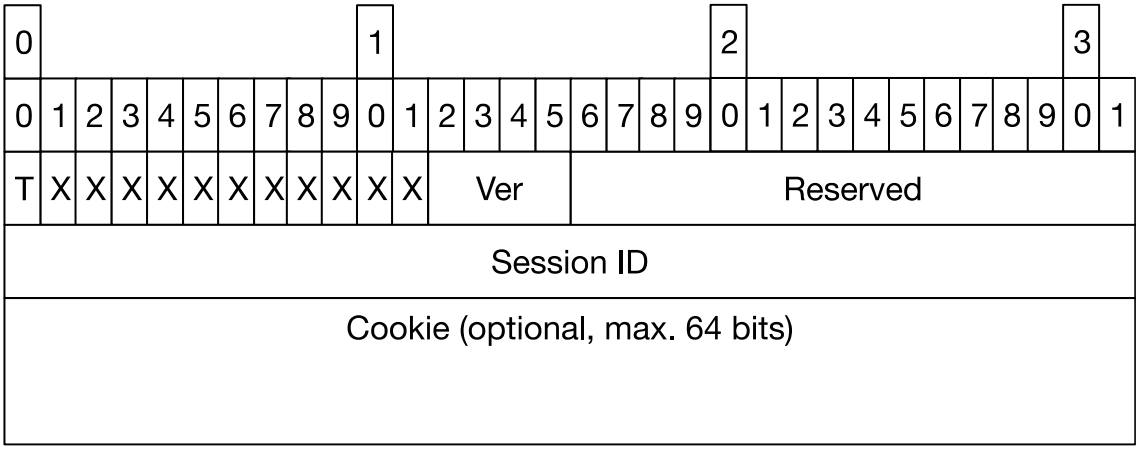

Figure 14. L2TPv3 session header over UDP

Over UDP, the first bit, *T*, is set to zero for a session packet. A value of 1 would mean that the following packet is a control packet. The subsequent *X* bits are set to zero and reserved for future extensions. The *Ver* field must be set to 3, indicating protocol version 3. The Session ID and cookie fields have equivalent functions to the header utilized over IP [38]

As with the other protocols, there are multiple open-source implementations of L2TPv3 - one example being the Linux L2TP driver, which can be utilized in a similar manner as a virtual networking device as outlined in the GRE chapter [40]

3.5.4 VXLAN

Virtual eXtensible LAN (VXLAN) is a fairly recently introduced protocol, and the RFC specifying it, RFC 7348, includes authors from Cumulus Networks (later acquired by Nvidia [41]), Arista, Broadcom, Cisco, VMWare, Intel, and Red Hat. [42]

Compared to the protocols introduced in the previous chapters that are more generic in nature, VXLAN was proposed to solve a very specific task that's also somewhat different in scope – namely the problems that arise from using traditional VLAN's in multi-tenant datacentre networks, such as the fairly limited number of possible VLAN ID's, excessively large MAC tables in Top-of-Rack (ToR) switches, and issues with redundancy provided by the Spanning Tree

Protocol (STP). Thus, VXLAN is geared towards specifically transporting Ethernet frames over IP, being able to handle a high number of distinct virtual networks, and offering good support for functionalities such as Equal-Cost Multipath (ECMP) [42].

VXLAN operates over UDP – thus mitigating the NAT and firewall traversal problems described in the GRE and L2TPv3 chapters, and uses the IANA assigned port number of 4789 as the destination port. For the source port it's recommended to calculate it using a hash based on the packet contents, to ensure a high level of entropy that's useful for load balancing. [42] The VXLAN header is illustrated in Figure 15.

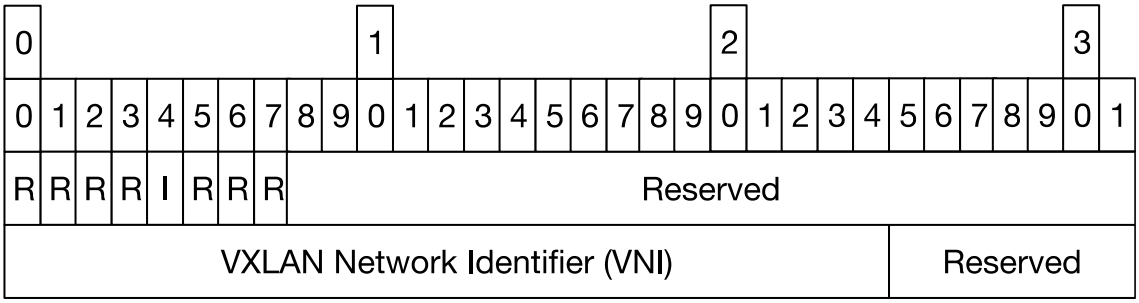

Figure 15. VXLAN header

The first 8 bits in the VXLAN header are the flags, of which the *I* flag must be set to 1, indicating a valid VXLAN network ID, and all the other flags (marked *R*), must be set to 0. The next 24 bits are reserved and must be set to zero. There is an IETF draft titled *Generic Protocol Extension for VXLAN (VXLAN-GPE)*, where it is proposed that some of the reserved bits would be used for a "Next Protocol" field, allowing other protocols besides Ethernet to be transmitted over VXLAN. [43] The reserved bits are followed by the VXLAN Network Identifier (VNI), which identifies the VXLAN segment – as discussed previously, this allows for up to 16 million distinct network segments to co-exist within the same administrative domain. The VNI is then followed by additional 8 bits of reserved padding. [42]

Standard VXLAN, as specified by RFC 7348, relies on multicast IP communication to transmit data destined to BUM (Broadcast, Unknown and Multicast) destinations. In other words, if a frame with an unknown destination

MAC address is to be sent over a VXLAN tunnel, the behaviour specified by RFC 7348 is to send it into an IP multicast group that's specific to the particular VXLAN Network Identifier. However, based on work by the IBM Distributed Overlay Virtual nEtwork Architecture (DOVE) project, it is also possible to use VXLAN without multicast IP, and instead replicate the BUM packets over a configured set of unicast tunnels. [44]

As is the case with the other protocols introduced here, several open-source implementations of the VXLAN protocol exist, such as for Linux [45], FreeBSD [46], and OpenBSD. [47] Open vSwitch also has support for VXLAN. [27]

3.5.5 GENEVE

Generic Network Virtualization Encapsulation (GENEVE), specified in RFC 8926, published in November 2020, with authors from Intel and VMWare, is the newest protocol discussed here. It's also the only one among the more data centre focused protocols (i.e. NVGRE, VXLAN, and GENEVE) that has proposed standard status. As might be fairly evident by this point, and as the GENEVE RFC itself notes, there is already quite an abundance of protocols solving very similar problems [48]:

> The large number of protocols in this space -- for example, ranging all the way from VLANs [IEEE.802.1Q_2018] and MPLS [RFC3031] through the more recent VXLAN (Virtual eXtensible Local Area Network) [RFC7348] and NVGRE (Network Virtualization Using Generic Routing Encapsulation) [RFC7637] -- often leads to questions about the need for new encapsulation formats and what it is about network virtualization in particular that leads to their proliferation. Note that the list of protocols presented above is non-exhaustive.

The main differences between the GENEVE protocol and VXLAN and NVGRE are that GENEVE doesn't specify any control-plane functionality in the RFC describing the data format (whereas VXLAN and NVGRE specify the multicast BUM forwarding / learning mechanism), and the addition of a variable-length options field that allow for extensible metadata in the data packets that is needed to support current and future control planes. [48]

As is the case with VXLAN, GENEVE also operates on top of UDP, using the IANA assigned destination port of 6081, and a source port that should be calculated from a hash based on the encapsulated packet, but should also remain the same for all packets in a flow to prevent re-ordering in transit due to different ECMP paths being taken by packets with different source ports. The GENEVE header is illustrated in Figure 16.

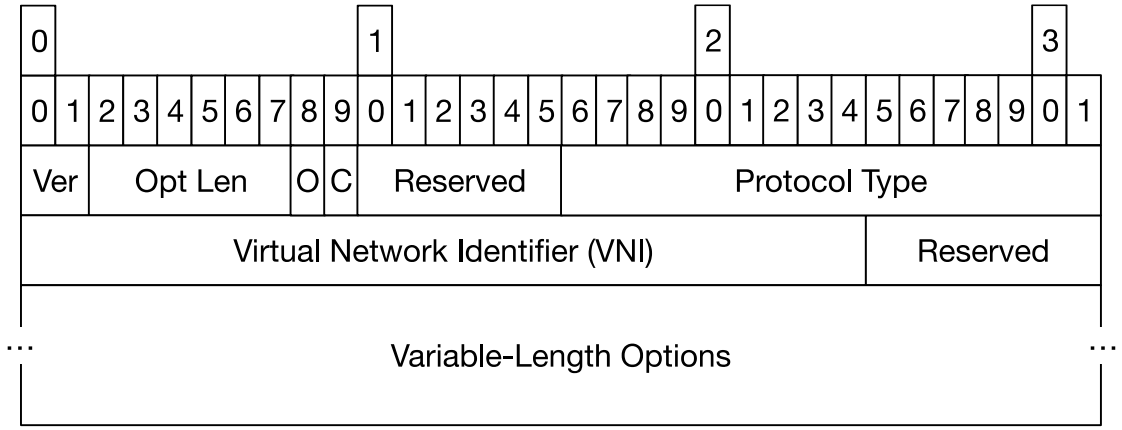

Figure 16. GENEVE header [48]

The first 2-bits are a version field, *Ver,* which is currently set to 0, and tunnel endpoints must drop packets with an unknown version number. The next 6 bits indicate the length of the Variable-Length Options field expressed in multiples of 4 bytes. This leads to a minimum GENEVE header length of 8 bytes (with no options) and maximum length of 260 bytes. The *O* bit indicates that the packet is a control packet, and the *C* field indicates the presence of critical options. [48]

The next 6 bits are reserved and must be set to zero on transmission and ignored upon reception. The following 2-byte *Protocol Type* field indicates the protocol of the encapsulated frame. This is indicated as an Ethertype, and thus, the value of 6558 hexadecimal is used for Ethernet. [15]

The *Virtual Network Identifier* again assigns a 24-bit identifier to each virtual network, making it possible to distinguish up to 16 million virtual networks between a host pair. The following 8 bits are again reserved and must be set to zero upon transmission and ignored on reception. This 8-byte header is then optionally followed by the variable-length options field. The format of the options and their possible proposed use cases are beyond the scope of this discussion. [48]

So, to conclude – compared to VXLAN, GENEVE adds some additional possibilities for future flexibility without increasing the minimum header length from the 8 bytes also utilized for VXLAN. Compared to NVGRE, it runs on top of UDP, thus it doesn't have the practical problems discussed earlier that arise from the use of GRE, or other OSI layer 4 protocols that aren't UDP, TCP, or ICMP, for that matter.

GENEVE is supported in the Linux Kernel. [49] Furthermore, Open vSwitch also has support for it. [27]

To conclude, several Layer 2 tunnelling protocols have been examined in this section. Each protocol offers distinct advantages and challenges. Protocols like GRE and L2TPv3, established in networking, provide fundamental tunnelling capabilities but face limitations in handling modern network virtualization needs. Newer protocols such as VXLAN, NVGRE, and GENEVE, developed with network virtualization in mind, effectively address these limitations. They offer improved scalability and flexibility, particularly in managing the Virtual Network Identifier (VID) issue, demonstrating advancements over older protocols.

3.6 BGP-Based EVPN Solutions

Border Gateway Protocol (BGP) is one of the foundational protocols of the Internet, used primarily for the dynamic exchange of IP routing information between distinct networks. Over the years, the use of BGP has evolved from purely IP routing to also provide dynamic configuration facilities for Ethernet Virtual Private Network (EVPN) solutions.

In subsection 3.6.1, we briefly investigate BGP as it's used for IP routing, as understanding the core functionality of the protocol is crucial to understand how it has evolved to serve as the control plane for EVPN solutions.

Following that, subsection 3.6.2 looks into the use of BGP as the control plane for EVPN networks as it is utilized with IP-based layer 2 tunnelling protocols – most often VXLAN.

It should perhaps be noted that BGP is not the only control plane protocol available for building EVPN's – for instance RFC 4762 specifies VPLS using MPLS Label Distribution Protocol (LDP) signalling [50]. However, since BGP is by far the most common protocol used for EVPN control plane purposes in IP-based networks, it's been chosen as the sole focus in this discussion.

3.6.1 Border Gateway Protocol

Before we delve into the EVPN use cases for BGP, it's perhaps worth taking a quick look at BGP in general, and the most common use case for it – that is, exchange of IP routing information.

BGP runs over TCP port 179. All BGP messages contain the message header. The header is illustrated in Figure 17.

Marker	Length	Type	Message Contents
16 Bytes	2 Bytes	1 Byte	0 - 4077 Bytes

Figure 17. BGP Header [8 pp 19]

The marker usually contains all 1’s and is used to check synchronization between the sender and receiver. If an unexpected value is received in the marker field, the receiver will send back an error message and close the connection. The length field indicates the length of the BGP message. The type field indicates the type of the BGP message. Four message types are defined in RFC 1771: open (1), update (2), notification (3), and keepalive (4). Later RFC’s define more message types. [8 pp 19]

The Open message is sent by both sides of a BGP session immediately after the TCP connection has been established. [8 pp 20] The format of the Open message is shown in Figure 18.

Version	My AS	Hold Time	Identifier	Parlen	Optional Parameters
1 Byte	2 Bytes	2 Bytes	4 Bytes	1 Byte	0 - 255 Bytes

Figure 18. BGP Open message [8 pp 20]

The Version field indicates the version of BGP used, which is typically 4. The My AS field indicates the sender’s Autonomous System (AS) number. The Hold Time indicates the maximum idle time for the session. As the Open message is sent by both parties in a BGP session, the lower value for hold time from these messages is selected to be used for the session. The Identifier field contains an IP address assigned to the BGP speaker that is to be used for the BGP session. A router must use the same identifier for all BGP sessions. [8 pp 20] The Parlen field contains the length of the length of the optional parameters field, or a zero if there is no optional parameters. [8 pp 20]

Upon receiving and accepting a BGP Open message, a router will send back a keepalive message, and start sending the routes to be shared over BGP using update messages. In further operation, only periodic keepalive messages and incremental updates to the routing information will be sent over the BGP session. [8 pp 20] The BGP update message format is detailed in Figure 19.

UR Length	Withdrawn Routes	PA Length	Path attributes	NLRI
2 Bytes	Variable	2 Bytes	Variable	Variable

Figure 19. BGP Update message [8 pp 20]

The Unfeasible Routes (UR) Length field indicates the length of the Withdrawn Routes field. If the UR length field is zero, it means that there are no withdrawn routes included in this update message. The withdrawn routes then follow. Each withdrawn route consists of a prefix length followed by the actual prefix that is to be withdrawn. The Path Attributes (PA) Length field indicates the length of the Path Attributes field in a similar fashion as the UR Length field indicates the length of the Withdrawn Routes field. The Network Layer Reachability Information (NLRI) field contains network prefixes for routes to be added in the same format as the Withdrawn Routes field contains routes to be removed. The path attributes listed in the message apply to all of the prefixes listed in the NLRI field. [8 pp 20-21]

The path attributes all begin with the first byte containing the attribute flags, and the second byte indicating the attribute type. The attribute flags (optional / transitive / partial) make it possible to extend BGP with new attributes while making it possible for existing BGP implementations to know how to handle the attributes, without knowing what the attributes mean. [8 pp 21]

The following well-known path attributes are defined in RFC 1771 [51]:

- Origin (type code 1): defines the origin of the path information (IGP, EGP, or Incomplete).
- AS path (type code 2): contains the AS path segments used to reach the prefixes indicated by the NLRI.
- Next Hop (type code 3): defines the IP address of the border router that should be used as the next hop for the destinations listed in the NLRI field.

Furthermore, some optional, non-transitive, and discretionary attributes are defined – so that in total RFC 1771 defines 7 type codes. Further study of the additional type codes is omitted here, but the RFC gives more information and guidance on how they should be used. [51]

3.6.2 A Network Virtualization Overlay Solution Using EVPN

There are multiple RFC's and other documents outlining the use of BGP for exchanging information about MPLS-based VPLS and Ethernet VPN networks, such as RFC 4761, *VPLS using BGP for Auto-Discovery and Signalling* [52], and RFC 7432, *BGP MPLS-Based Ethernet VPN* [1], for instance. However, as has been outlined earlier, architectural approaches based on MPLS necessitate the use of MPLS-capable network hardware and software. Since we are interested in communicating over the 4G / 5G cellular network user plane data paths, the network isn't natively end-to-end MPLS capable, which would necessitate the use of yet another underlying tunnelling layer, if MPLS-based transport technologies were to be used.

The primary RFC document that outlines the use of BGP control plane for EVPN's built on top of IP-based layer 2 tunnelling protocols is RFC 8365, *A Network Virtualization Overlay Solution Using Ethernet VPN (EVPN)*. The RFC discusses the use of the functionality with several layer 2 tunnelling options: namely VXLAN, NVGRE and MPLS over GRE. It also notes that the specification is applicable to GENEVE, with some incremental work required. [2]

RFC 8365 builds upon multiple successive enhancements that have been incorporated into BGP since the version 4 specification was released as RFC 1771. An important addition is RFC 4760, *Multiprotocol Extensions for BGP-4*, which enables the use of BGP for relaying routing information about protocols other than IPv4. [53]

A related necessary addition is specified in RFC 3392, *Capabilities Advertisement with BGP-4*, which makes it possible for routers to advertise their supported capabilities as an optional parameter in the BGP Open message. [54] This capabilities advertisement functionality is then used in RFC 4760, so that routers are able to advertise their support for routing a particular Non-IPv4 protocol. [53]

RFC 4760 defines a BGP attribute called Multiprotocol Reachable NLRI, which has the type code 14. [53] The format of the Multiprotocol Reachable NLRI attribute is illustrated in Figure 20.

AFI	SAFI	Length of Next Hop Network Address	Network Address of Next Hop	Reserved	NLRI
2 Bytes	1 Byte	1 Byte	Variable	1 Byte	Variable

Figure 20. Multiprotocol Reachable NLRI

The Address Family Identifier (AFI) and Subsequent Address Family Identifier (SAFI) fields are used to identify the set of protocols that the address in the Next Hop field must belong to and how it is encoded, and also the format of the contents of the Network Layer Reachability Information (NLRI) field that follows. The Length of the Next Hop Network address field expresses the length of the subsequent Network Address of Next Hop field, which, again, is interpreted based on the contents of the AFI and SAFI.

RFC 7432 specifies that the AFI/SAFI pair utilized for EVPN is AFI of 25 (L2VPN) and SAFI of 70 (EVPN). [1] This same AFI/SAFI pair is utilized in IP-based overlay EVPN's as well. The format of the EVPN NLRI is detailed in Figure 21.

Route Type	Length	Route Type Specific
1 Byte	1 Byte	Variable

Figure 21. EVPN NLRI Format [1]

There are four route types defined in RFC 7432 [1]:

- Type 1: Ethernet Auto-Discovery (A-D) route
- Type 2: MAC/IP Advertisement route
- Type 3: Inclusive Multicast Ethernet Tag route
- Type 4: Ethernet Segment route

Additionally, route type 5, *IP Prefix Route*, has been specified in RFC 9136 [55], and routes of Type 6, *Selective Multicast Ethernet Tag Route*, Type 7, *Multicast Membership Report Synch Route*, and Type 8, *Multicast Leave Synch Route*, have been specified in RFC 9251. [56]

Since the desired outcome for this thesis is merely to set up single-homed layer 2 tunnels and bridge traffic through them, we only need to really be concerned with two of the route types. Those are the Type 2, MAC/IP Advertisement route, and Type 3, Inclusive Multicast Ethernet Tag (IMET) route.

The Type 3 IMET route is typically sent first. The contents of the Type 3 NLRI are outlined in Figure 22.

RD	Ethernet Tag ID	IP Address Length	Originating Router's IP Address
8 Bytes	4 Bytes	1 Byte	4 (or 16) Bytes

Figure 22. IMET Route Format [1]

The first field in the IMET route is the Route Distinguisher (RD). Since it's possible for multiple virtual networks to utilize the same addresses, the RD is used to distinguish between them. RFC 4364 defines three types for the RD, of which RFC 7432 recommends Type 1 to be used. This means that the RD should be an IP address (typically, a loopback address on the router), followed by an unique number. The Ethernet Tag ID that follows is either a 12-bit or 24-bit identifier that is used to identify a particular Ethernet broadcast domain in an EVPN instance. The following IP Address Length field specifies whether the following address is an IPv4 or IPv6 address, and the Originating Router's IP Address then follows. [1]

An IMET route must also carry one or more Route Target (RT) attributes. According to RFC 8365, both the RD and the RT can be automatically derived from the VNI, or they can be set manually. The RT is used to carry, among other things, the information about which tunnelling protocol is to be used. [2]

In simple terms, the Type 3 route conveys the information that a router with a certain IP address wants to participate in an EVPN for the given Ethernet Tag ID. The type of tunnelling protocol to be used is told as part of the Route Target. Based on Type 3 routes, it's then possible for participating routers to flood Broadcast, Unknown, and Multicast (BUM) traffic to the participating routers, using the desired layer 2 tunnelling protocol as encapsulation.

To exchange information about actual Ethernet hosts that are reachable via the routers participating in the EVPN, routes of type 2, or MAC/IP Advertisement routes, must be used. The contents of the type 2 route are shown in Figure 23.

RD	ESI	Ethernet Tag ID	MAC Address Length	MAC Address	IP Address Length	IP Address	MPLS Label1	MPLS Label2
8 Bytes	8 Bytes	4 Bytes	1 Byte	6 Byte	1 Byte	0, 4 or 16 Bytes	3 Bytes	0 or 3 Bytes

Figure 23. MAC/IP Advertisement Route

The RD in the MAC/IP Advertisement route serves the same function as in the IMET route. As it's possible in the case of multi-homed EVPN that the same MAC address is reachable via multiple paths, the ESI is used to distinguish between ethernet segments. The Ethernet Tag ID, again, serves a similar function as in the Type 3 route. The MAC Address Length field is always set to 48 bits and is followed by the 6-byte Ethernet MAC Address. An IP Address Length field, followed by an IP Address field then follows. By default the IP Address Length is set to 0, and no IP Address is included in the MAC/IP Advertisement route. [1]

Finally, two fields designated as MPLS Label1 and MPLS Label2 follow. In RFC 8365, the MPLS Label1 field is used to convey the Virtual Network Identifier (VNI), and thus it is called the VNI field. [2]

So, while the Type 3 route was utilized to advertise the availability of a given EVPN network at an IP tunnelling endpoint for BUM traffic, Type 2 routes are used to advertise that a specific MAC address is reachable at a tunnelling endpoint for sending unicast traffic to.

Thus, in conclusion, by the mechanisms outlined in RFC 8365, RFC 7432, and earlier extensions to BGP, it's possible to use BGP for dynamic establishment of EVPN networks based on layer 2 tunnelling protocols, such as VXLAN.

Support for BGP-based EVPN has been implemented in the Free Range Routing (FRR) open-source software package. [57] Furthermore, the GoBGP package, programmed in the Go programming language, also has experimental support. [58]

4 Hardware Acceleration Support for Layer 2 Tunnelling

In this chapter, we look into the role of hardware acceleration in the implementation of layer 2 tunnelling protocols. This chapter aims to explore the various implementation approaches for tunnelling protocols, and to highlight how hardware acceleration can improve the scalability of these solutions.

4.1 Tunnelling Protocol Implementation Approaches

In the previous chapter we outlined a myriad of existing layer 2 tunnelling protocols that operate over IP networks – and even some that natively don’t do so (i.e., MPLS). As was noted, all of these protocols have been implemented as open-source software, alongside the BGP control-plane functionality required for dynamic EVPN establishment. However, implementing tunnelling protocols purely as software functionality inside the OS Kernel has its limitations, particularly in terms of processing overhead, which can significantly limit throughput and packet rate.

Packet processing in typical operating systems involves a multi-stage pipeline that spans the network interface controller (NIC), kernel space, and user space. When a packet is received, it triggers a hardware interrupt, followed by a software interrupt request (softirq) in the kernel, leading to the packet being processed through various protocol layers. This pipeline can become a bottleneck in high-throughput environments. When layer 2 tunnelling protocols are used, the added complexity of packet encapsulation and decapsulation further contributes to processing overhead. The packet first undergoes encapsulation at the sender's end, adding an additional layer to the packet, and is later decapsulated at the receiver's end. This additional processing step, while necessary for network virtualization, introduces more CPU load and can reduce throughput, especially in high-speed network environments, or networks with a significant number of tunnel endpoints. [59]

Over the years, various approaches for mitigating this processing overhead have been implemented. One of them is to by-pass the normal packet processing path in the OS Kernel, for example by using functionality such as Linux eXpress Data Path (XDP), or the Data Plane Development Kit (DPDK). Open vSwitch has support for DPDK, which can significantly enhance the throughput of Open vSwitch, and has support for performance improvements via the use of DPDK for VXLAN tunnelling as well. [60] There are also programmable so called SmartNICs that can further accelerate packet processing when DPDK or XDP is used, such as the Mellanox ConnectX series of NICs. [61]

In the case when the end goal of layer 2 tunnelling is to transfer packets to and from a hardware switched layer 2 network, it's also possible to implement the tunnelling encapsulation and de-encapsulation, and IP transmission of the tunnelled packets, entirely in the hardware logic, so that the packets never have to leave the "forwarding plane" of the switch – i.e. they are never processed by a general purpose CPU and operating system. The inner workings of the packet processing pipelines are confidential information of the network silicon vendors, but some details of how a possible implementation might work can be found in various research papers – such as *Open vSwitch Vxlan Performance Acceleration in Cloud Computing Data Center* by Yan and Wang, where an FPGA-based accelerator for Open vSwitch VXLAN data path is proposed. [62]

4.2 Protocol Support in Switch ASICs

As outlined in the previous section, hardware acceleration exists for various tunnelling protocols. In this section, we will take a look at the hardware acceleration capabilities in programmable Ethernet Switch Application Specific Integrated Circuits (ASICs) with 3.2 Tbps of bandwidth. The 3.2 Tbps family of ASIC has been selected because it's available from most high-performance switch chip vendors and strikes a reasonable balance between cost and performance. This class of switch chips is typically utilized in Top-of-Rack, aggregation, and spine switches in modern datacenters.

The major companies manufacturing high-speed Ethernet switching chips are Broadcom, Nvidia Mellanox, and Marvell. [63] Furthermore, e.g. Cisco is making their own switching silicon.

Looking at the offering from these companies, the following switch chips used in 3.2 Tbps switches were selected for comparison:

- Broadcom Trident3-X7 [64]
- Nvidia Mellanox Spectrum 2 [65]
- Marvell Prestera 98CX8520 [66]
- Cisco Silicon One Q202 [67]

Further looking at the datasheets for the selected switch chips, support for the following layer 2 tunnelling protocols is indicated:

Chip	**Protocol Support Indicated**
Trident3-X7	GENEVE, VXLAN(-GPE), MPLS
Spectrum 2	GENEVE, VXLAN(-GPE), NVGRE, MPLS
Prestera 98CX	GENEVE, VXLAN(-GPE), NVGRE, MPLS
Cisco Q202	VXLAN, GRE, MPLS

Table 1. L2 tunnelling protocol support of switch chips [64], [65], [66], [67]

Thus, it can be observed, that VXLAN is supported in every chip selected, with GENEVE and NVGRE also having wide-spread support. However, in addition to the switch chip having support, the operating system of the switch must also have support for configuring the desired tunnelling and control plane functionality. But since the CPUs on switches are generally reserved for management functionality and control plane traffic only, having support on the ASIC is generally a pre-requisite for wire speed tunnelling on modern switches and routers.

5 PoC Proposal and Implementation

This chapter presents an overview of the Proof-of-Concept proposal and its subsequent implementation. The goal of this final chapter is to tie together the theoretical concepts discussed earlier into a concrete example solution in a 4G / 5G private wireless network environment.

5.1 The Existing 4G / 5G Test Network

An existing laboratory test network was used as the basis for the tests. The high-level network architecture is outlined in Figure 24.

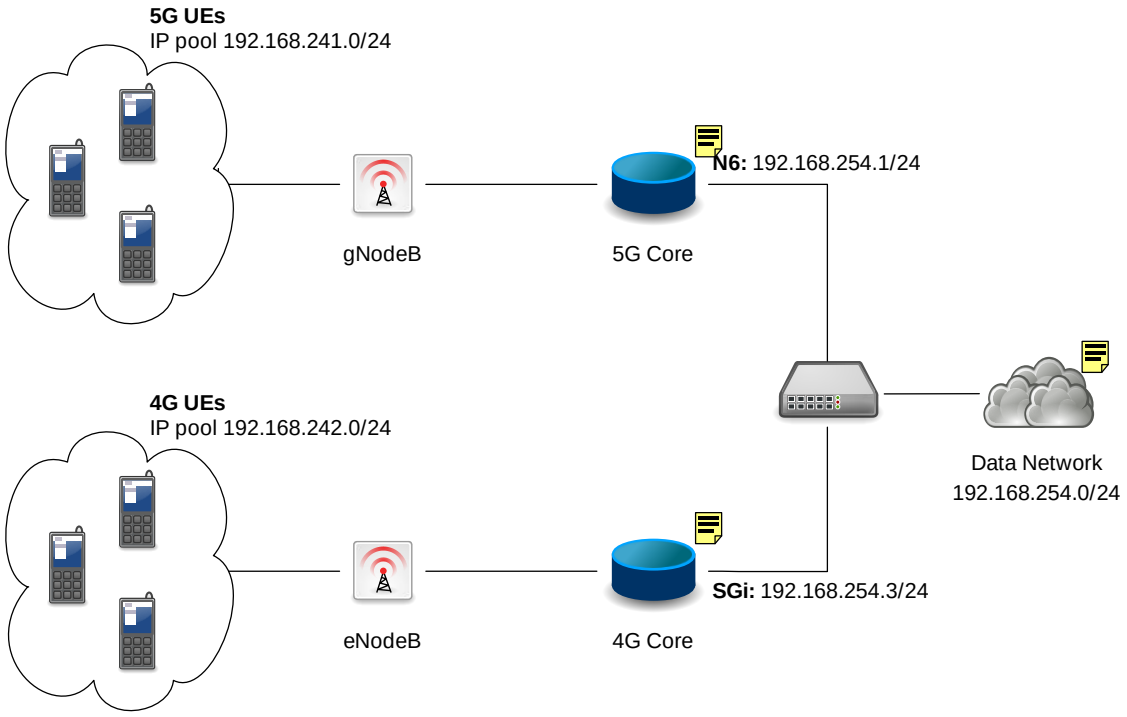

Figure 24. Existing Test Network

The existing test network has multiple eNodeB's and gNodeB's on various frequencies, and both 4G and 5G cellular network cores. The 5G core is assigning UE IP addresses from the pool 192.168.241.0/24 and the N6 IP address is 192.168.254.1/24. The 4G core is assigning UE IP addresses from the pool 192.168.242.0/24 and the SGi IP address is 192.168.254.3/24. Both cores are connected through routers and switches to the data network, which has

multiple servers and other equipment residing in the 192.168.254.0/24 address space, with routing to other networks as needed.

5.2 Criteria and Initial High-Level Design for the PoC

The PoC must be able to bridge Ethernet segments behind both 4G and 5G mobile routers with segments behind a gateway residing in the data network. Static UE IP address allocation on the mobile routers must not be used, and the VXLAN gateway in the Data Network must not require additional configuration to be added per individual endpoint.

Based on the review of the available protocols for tunnelling and dynamic EVPN control-plane in chapter 3, and on the review of available hardware functionality in chapter 4, a high-level design for the PoC was drafted. The high-level design is illustrated in figure 25.

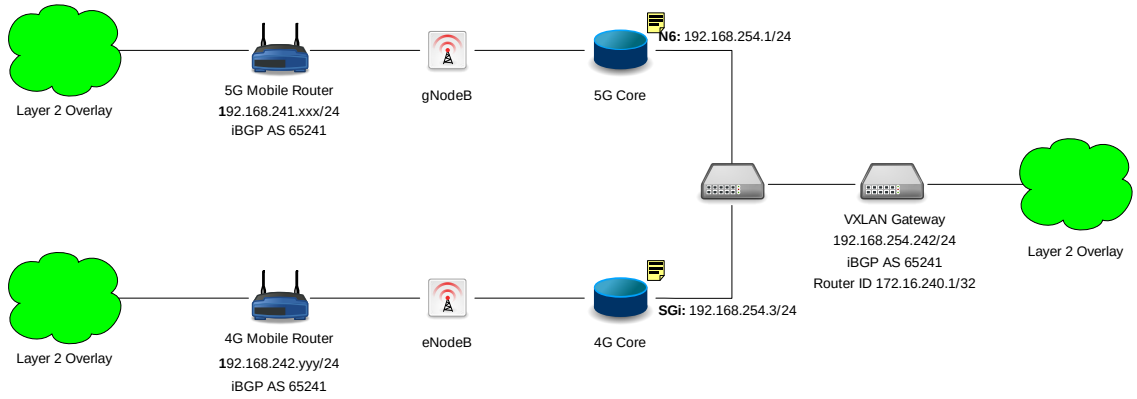

Figure 25. PoC High-Level Design

For the PoC, VXLAN with BGP EVPN control plane was selected as the technology, as this has the widest support in hardware and software.

The Layer 2 overlay network spans the LAN side of the 4G and 5G mobile routers, and also the LAN side of the VXLAN gateway in the data network.

The mobile routers get dynamically assigned IP addresses from the APN UE pools of the 4G and 5G core networks, and they use BGP with the AS number of

65241. On the mobile routers, the dynamically assigned UE IP is the router ID. While this approach has some drawbacks, it simplifies configuration – as using a separate loopback address as the router ID would require the loopback address to be reachable via the cellular network. While this is possible, as the mobile cores support so-called Framed Route functionality, additional configuration and complexity would still be required.

The VXLAN gateway has the IP address 192.168.254.242/24 in the data network, and it also has a separate loopback address of 172.16.240.1/32 that is used as the Router ID. This means that routing needs to be configured so that:

- The 4G and 5G cores know to reach 172.16.240.1/32 via 192.168.254.242
- The VXLAN gateway knows that 192.168.241.0/24 is reachable via 192.168.254.1, and 192.168.242.0/24 is reachable via 192.168.254.3

The aforementioned routes are configured statically in the network, reducing the complexity for this PoC. They could also be provided via other means, for example, through separate BGP “underlay” routing, or by another dynamic routing protocol, if desired.

The AS number on the VXLAN gateway is also 65241. This means that in this PoC, so-called Interior BGP (IBGP) routing is used. This makes it easier to manage the routing policies and configuration in the scope of this PoC, compared to the alternative of having multiple AS’s, in a configuration which is called Exterior BGP, or EBGP.

5.3 Hardware and Software Selection and Configuration

In this section, the equipment and software selected for the PoC is described in more detail, alongside the necessary software and configuration used to achieve the desired functionality.

As there appears to be no generally available mobile router that supports VXLAN EVPN with BGP control plane, custom Proof-of-Concept routers were crafted based on off-the-shelf development boards, cellular modems, and the OpenWRT open-source operating system. Both the 4G and 5G and routers used in the PoC follow typical architecture as observed in commercial cellular routers, while support for mainline OpenWRT brings additional software flexibility that's mandatory for the purposes of this PoC.

5.3.1 4G Mobile Router Hardware

The 4G mobile router is based on a Sinovoip Banana Pi BPI-R3 development board. This is a high-performance router development board based on the MediaTek MT7986 (Filogic 830) System-on-Chip, which has four ARM Cortex A53 64-bit processor cores. The board has 2GB of Random-Access Memory (RAM), 8GB of built-in embedded Multimedia Card (eMMC) FLASH and a slot for a micro Secure Digital (microSD) card, two 2.5GbE Small Form-factor Pluggable (SFP) cages, and five Gigabit Ethernet Ports. The board has on-board 2.4GHz and 5GHz Wi-Fi 6, and a Micro PCI Express (mPCIe) slot for a cellular modem, with a single micro-SIM slot. [68]

A Telit LM940 LTE modem was selected to be used to provide cellular connectivity for this board. The LM940 is an LTE Category 11 cellular modem based on the Qualcomm Snapdragon X12 LTE Modem, built in a full-size, single-sided mPCIe card form factor. [69] The 4G router hardware is photographed in Figure 26.

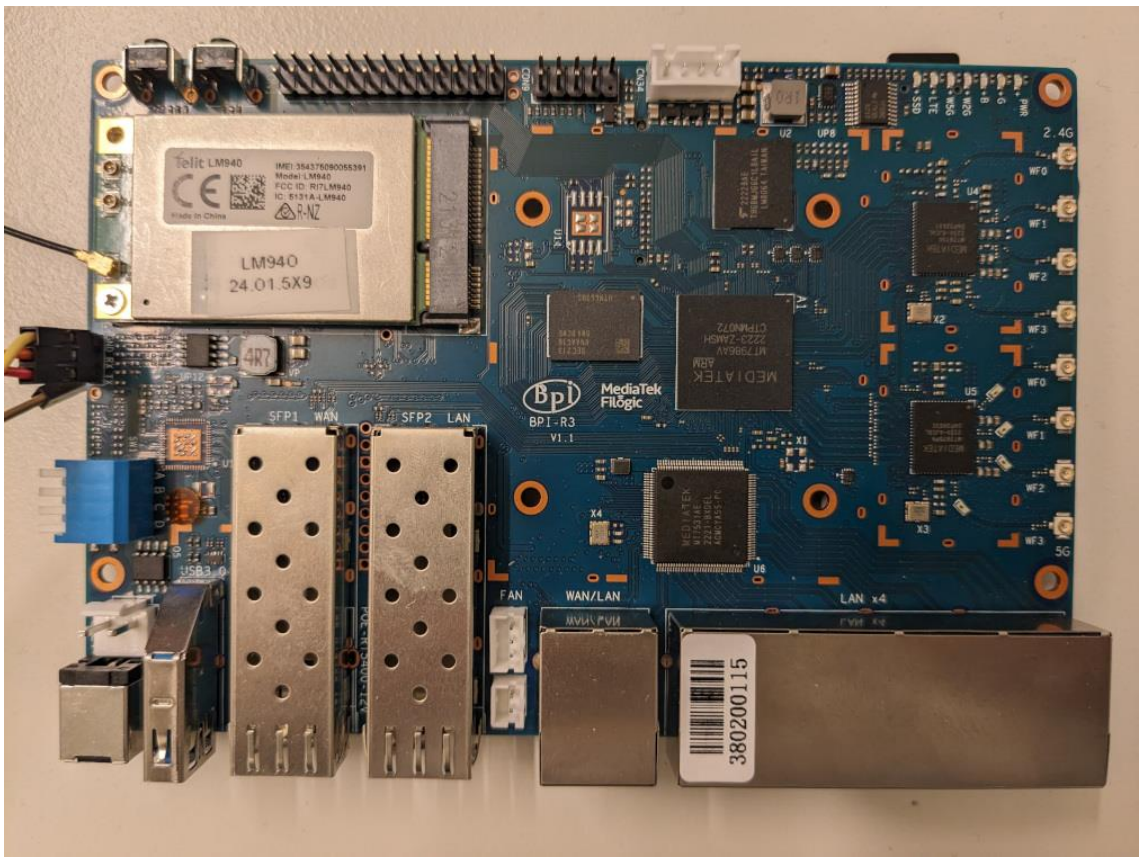

Figure 26. 4G Router Hardware

5.3.2 5G Mobile Router Hardware

Different hardware was utilized for the 5G Mobile Router. This was due to various reasons, such as there being only one of the BPI-R3 boards available, the limitations on the mPCIe interface on that board, but also to test the generalizability of the proposed solution.

The development platform selected for the 5G Mobile Router was the Globalscale MochaBIN. This board has a Marvell Armada 88F7040 System-on-Chip, which has four ARM Cortex A72 cores. The board has 4 GiB of DDR4 RAM and 16 GB of built-in eMMC storage. It has a 10-gigabit SFP+ cage and a gigabit SFP cage, alongside 5 Gigabit RJ45 Ethernet ports, one of which has Power-over-Ethernet (PoE) in functionality. The board has no on-board WiFi, but a separate WiFi card can be inserted into the mPCIe slot. The board has a m.2 B-key slot for inserting a cellular modem, and two SIM slots – one of which is connected to the m.2 slot, and the other one to the mPCIe slot. [70]

A Telit FN990A28 modem module was used to provide cellular connectivity for this router. The FN990 is a 5G sub-6GHz m.2 module based on the Qualcomm Snapdragon SDX62 modem chipset. [71] A picture of the 5G router as used for the PoC is in Figure 27.

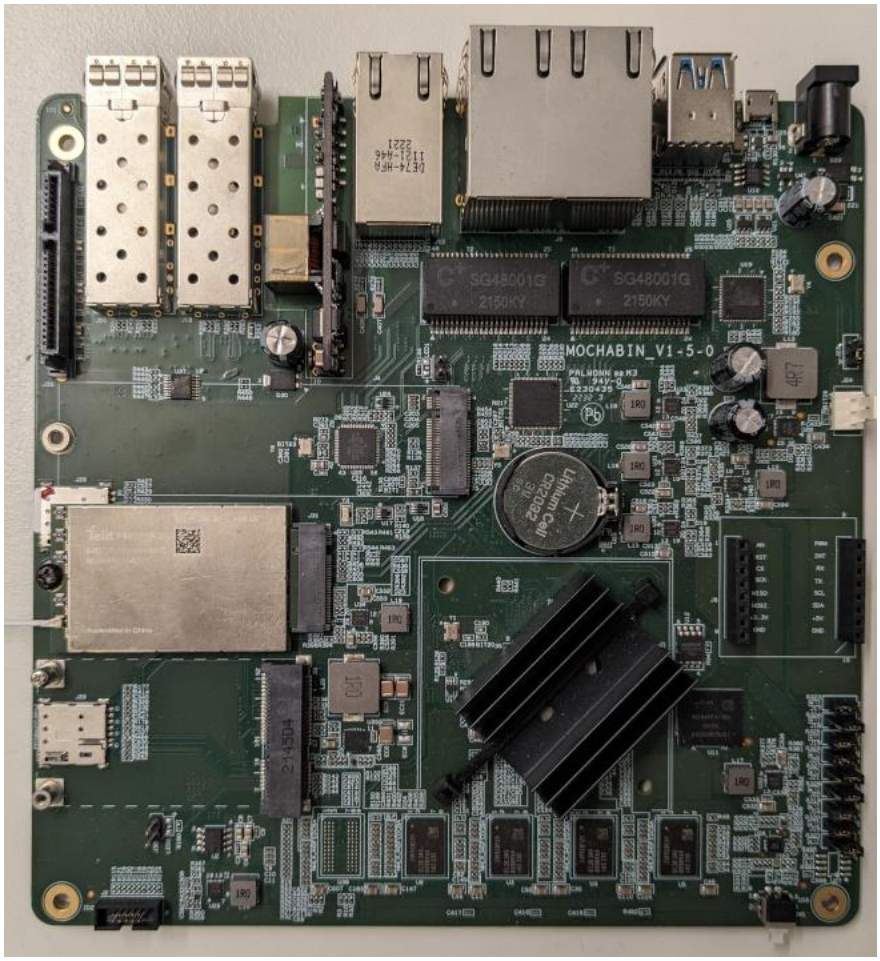

Figure 27. 5G Router Hardware

5.3.3 Mobile Router Software

As mentioned, OpenWRT was used as the base software platform for the mobile routers. OpenWRT is an open-source project that provides a Linux operating system for embedded devices – mainly various kinds of routers and wireless access points. It also provides a Buildroot-based framework for customizing and building firmware images for embedded devices, alongside a large package collection. At the time of writing, the current stable release of OpenWRT is 23.05.2 [72]

For the PoC, the Free Range Routing (FRR) packages and support for VXLAN were added to the OpenWRT base image, alongside ModemManager, which is a software package for controlling cellular modems. Some necessary troubleshooting tools were also included, such as the picocom terminal emulator application, and the tcpdump traffic capture package. Instead of the default *ip* tool, which is a reduced version from the busybox suite of utilities, the full iproute2 package was also added.

The FRR BGP daemon in OpenWRT is built in such a way that it requires TCP MD5 signature functionality being enabled in the Linux Kernel. However, the default OpenWRT kernel doesn't enable this option. So, for both router platforms, the Kernel, along with the whole OpenWRT base system image, was re-built with the CONFIG_TCP_MD5SIG flag enabled. [73] The resulting custom base-system images were then used in combination with the OpenWRT image builder tool, to craft custom firmware images which included all the necessary packages and the custom-built kernel.

Both mobile routers were configured to act as a VXLAN VTEP and learn and advertise EVPN routes over BGP. Firstly, the interfaces were configured as shown in Listing 1.

```
config interface 'wwan'
        option device '/…'
        option proto 'modemmanager'
        option apn 'internet'
        option allowedauth 'none'
        option iptype 'ipv4'

config interface 'vxlan100'
        option proto 'vxlan'
        option port '4789'
        option vid '100'
        option learning '0'
        option tunlink 'wwan'

config interface 'vni100'
        option type 'bridge'
        list ports 'lan1'
        list ports 'vxlan100'
```

Listing 1. Mobile router network interface configuration (/etc/config/network)

Listing 1 contains the relevant portion of the OpenWRT network configuration file. On the mobile routers, the *wwan* interface is the cellular modem. The device path points to the USB device in the /sys filesystem – but as it's different on both devices, it is not included here. The modem is managed by the ModemManager software package, and connects to the *internet* APN using IPv4 as the session type.

In the *vxlan100* configuration, the interface is set to use the VXLAN protocol with UDP port 4789. It uses a VNI of 100. The automatic MAC address learning is disabled (learning '0'), as the MAC learning is controlled by BGP EVPN. The VXLAN tunnel is linked to the cellular modem connection established by the wwan interface.

For *vni100*, the configuration establishes a network bridge, combining physical *lan1* port and the virtual *vxlan100* interface. This setup allows direct communication between the physical local network and the virtual VXLAN segment.

In the main configuration file for FRR, only some debug options for BGP and a static for 172.16.240.1/32 on the *wwan0* link is configured, as indicated by Listing 2. As FRR needs to be able to resolve paths towards routes it receives over BGP, a static route also needs to be added towards the UE pool of the other network than the one the UE is connected to. This is not an optimal solution, but a better one was not discovered in the timeframe of the PoC.

```
!
debug bgp neighbor-events
debug bgp update-groups
debug bgp updates in
debug bgp updates out
debug bgp zebra
!
ip route 172.16.240.1/32 wwan0
ip route 192.168.241.0/24 wwan0
!
```

Listing 2. Relevant portion of the main FRR configuration file

Furthermore, as shown in Listing 3, BGPd needs to be specifically enabled in the file */etc/frr/daemons*.

```
bgpd=yes
```

Listing 3. Enabling BGPd in /etc/frr/daemons

The actual BGP configuration happens dynamically when the wwan0 interface comes up, as that's the point in time at which it's possible to know the UE IP address, which is used as the Router ID for the mobile routers. The script that runs when wwan0 comes up is placed in */etc/hotplug.d/iface/30-wwan-router-id*, and it is shown in Listing 4.

```
#!/bin/sh

if [ "$DEVICE" = "wwan0" ]; then
        wwan_ip=$(ip -4 a show dev wwan0| \
          awk '/inet/ { split($2, a, "/"); print a[1]; }')

        tmpfile=$(mktemp)

        sed "s:{{IP}}:$wwan_ip:g" \
          /etc/frr/bgp-template.conf > $tmpfile

        vtysh < $tmpfile > /dev/null

        rm $tmpfile
fi
```

Listing 4. Interface hot plug script

This script is run for all interfaces, with the interface name being placed into the *$DEVICE* environment variable. This script only activates if the device is *wwan0*. Upon activation, the IP address of the interface is retrieved using the *ip* and *awk* tools, and placed into the *$wwan_ip* variable. A temporary file is created by *mktemp*. The configuration command template file in */etc/frr/bgp-template.conf* is then read by the *sed* tool, with the template variable {{IP}} being replaced by the IP address in *$wwan_ip.* The output of *sed* is directed to the previously created temporary file. The contents of the temporary file are then fed into the *vtysh* tool, which is the command line for configuring FRR. The contents of the configuration template file */etc/frr/bgp-template.conf* are displayed in Listing 5.

```
configure terminal
router bgp 65241
 bgp router-id {{IP}}
 no bgp default ipv4-unicast
 neighbor BGP-EVPN peer-group
 neighbor BGP-EVPN remote-as 65241
 neighbor 172.16.240.1 peer-group BGP-EVPN
 address-family l2vpn evpn
  neighbor BGP-EVPN activate
  advertise-all-vni
  vni 100
    rd {{IP}}:100
  exit-vni
 exit-address-family
exit
```

Listing 5. Contents of /etc/frr/bgp-template.conf

In this file, a BGP routing instance with the AS number 65241 is configured. The router ID is dynamically set to the wwan0 device IP, as explained previously. IPv4 unicast routing is disabled entirely.

A peer group for EVPN is created with *neighbor BGP-EVPN peer-group*, and it's specified that the peers in this group will use the same AS 65241, indicated by *neighbor BGP-EVPN remote-as 65241*. The VXLAN-EVPN gateways Router ID, 172.16.240.1, is then added to this BGP-EVPN peer group.

The configuration moves into the EVPN context with *address-family l2vpn evpn*. Here, the BGP-EVPN peers are activated for EVPN route exchange with *neighbor BGP-EVPN activate*, and the router is instructed to advertise all VXLAN Network Identifiers (VNIs) associated with the EVPN using *advertise-all-vni*. This setup ensures that all VNIs are known to the BGP peers. Additionally, the Route Distinguisher for VNI 100 is set to be the UE IP, followed by the VNI ID 100.

5.3.4 VXLAN-EVPN Gateway

A Nokia 7220 IXR-D2 Interconnect Router was selected to be used as the data network side gateway. The 7220 IXR-D series of routers are high performance routers, designed to be used in data centre connectivity fabric as leaf and spine routers. The 7220 IXR-D2 is a one rack unit router with 48x 25-gigabit SFP28 ports, and 8x 100-gigabit QSFP28 ports. It has a system throughput of 4.0 Tb/s (Half Duplex). The 7220 IXR-D series devices all run the Nokia Service Router Linux (SR Linux) Network Operating System (NOS). [74]

The 7220 IXR-D2 is used to interface towards the DN-side ethernet segment on layer 2, and towards the N6 and SGi interfaces of the mobile cores on the IP layer. It also acts as a central “hub”, or a so-called Route Reflector, for the BGP routing. The 7220 IXR-D2 was configured to accept BGP EVPN peering from the UE address pools, and switch the VXLAN tunnels with the Ethernet segment VLAN. The relevant configuration sections are described next.

```
interface ethernet-1/1 {
    admin-state enable
    vlan-tagging true
    ethernet {
        port-speed 10G
    }
    subinterface 70 {
        admin-state enable
        ipv4 {
            admin-state enable
            address 192.168.254.242/24 {
            }
        }
        vlan {
            encap {
                single-tagged {
                    vlan-id 70
                }
            }
        }
    }
    subinterface 108 {
        type bridged
        admin-state enable
        vlan {
            encap {
                single-tagged {
                    vlan-id 108
                }
            }
        }
    }
}
```

Listing 6. ethernet-1/1 interface configuration

In Listing 6, the first SFP28 port interface, ethernet-1/1, is configured with two VLANs, with VLAN ID's 70 and 108. This is achieved by enabling vlan-tagging with *vlan-tagging true*, having distinct sub-interfaces on the physical interface identified by the VLAN ID (as is a good convention), and the *vlan encap single-tagged vlan-id* configuration section which identifies the VLAN tag that should be used for this interface. Subinterface 70 is configured with the IP address 192.168.254.242/24. As subinterface 108 is only used for bridging, it doesn't need to have an IP address associated with it.

```
interface system0 {
    admin-state enable
    subinterface 0 {
        ipv4 {
            admin-state enable
            address 172.16.240.1/32 {
            }
        }
    }
}
```

Listing 7. system0 interface configuration

In Listing 7, the system0 interface is configured with a subinterface 0, that has the IP address of 172.16.240.1/32. This is the way loopback interfaces are configured in SR Linux, and this configuration enables the use of this loopback address as the Router ID for BGP.

```
tunnel-interface vxlan1 {
    vxlan-interface 1 {
        type bridged
        ingress {
            vni 100
        }
    }
}
```

Listing 8. vxlan1.1 tunnel interface configuration

In listing 8, the vxlan1.1 tunnel interface is configured to use VNI 100.

```
network-instance layer2-mac-vrf {
    type mac-vrf
    admin-state enable
    interface ethernet-1/1.108 {
    }
    vxlan-interface vxlan1.1 {
    }
    protocols {
        bgp-evpn {
            bgp-instance 1 {
                admin-state enable
                vxlan-interface vxlan1.1
                evi 100
            }
        }
        bgp-vpn {
            bgp-instance 1 {
                route-target {
                    export-rt target:65241:100
                    import-rt target:65241:100
                }
            }
        }
    }
}
```

Listing 9. Configuration for layer2-mac-vrf

In Listing 9, a network instance called layer2-mac-vrf, of type *mac-vrf*, is configured. The mac-vrf type of Virtual Routing and Forwarding (VRF) operates on layer 2, and can be thought of essentially as a virtual layer 2 switch instance.

Two interfaces are added to this mac-vrf: subinterface 108 of ethernet-1/1, as configured earlier, and the VXLAN interface vxlan1.1. BGP EVPN is configured in this network instance, using the VXLAN interface vxlan1.1 and an EVPN Virtual Identifier (EVI) of 100.

The route targets are configured so that routes from EVI 100 of AS 65241 are imported into the BGP-VPN instance, and conversely, routes from this instance are exported into the same EVI / AS.

```
network-instance lbo-ip-vrf {
    interface ethernet-1/1.70 {}
    interface system0.0 {}
    protocols {
        bgp {
            admin-state enable
            autonomous-system 65241
            router-id 172.16.240.1
            dynamic-neighbors {
                accept {
                    match 192.168.241.0/24 {
                        peer-group BGP-EVPN
                    }
                    match 192.168.242.0/24 {
                        peer-group BGP-EVPN
                    }
                }
            }
            group BGP-EVPN {
                admin-state enable
                export-policy all
                import-policy all
                peer-as 65241
                afi-safi evpn {
                    admin-state enable
                }
                afi-safi ipv4-unicast {
                    admin-state disable
                }
                route-reflector {
                    client true
                    cluster-id 172.16.240.1
                }
                timers {
                    minimum-advertisement-interval 1
                }
            }
        }
    }
    static-routes {
        route 192.168.241.0/24 { next-hop-group sa_upf }
        route 192.168.242.0/24 { next-hop-group nsa_pgw }
    }
    next-hop-groups {
        group nsa_pgw {
            nexthop 0 {
                ip-address 192.168.254.3
            }
        }
        group sa_upf {
            nexthop 0 {
                ip-address 192.168.254.1
            }
        }
    }
}
```

Listing 10. lbo-ip-vrf configuration

In Listing 10, an IP VRF called lbo-ip-vrf is configured. An IP-type VRF can be thought of as a virtual IP routing instance. This VRF has two interfaces: the VLAN 70 sub-interface on ethernet-1/1, and the loopback interface system0.0.

BGP is configured on this VRF, using AS number 65421, and router ID of 172.16.240.1. The *dynamic-neighbors* configuration section makes it possible to accept peering with hosts in the entire subnets of 192.168.241.0/24 and 192.168.242.0/24, instead of having to manually configure every peer address. Peers from these subnets are accepted and placed into the peer-group *BGP-EVPN*.

In the BGP-EVPN peer group, all routes are accepted for import and export from the evpn AFI/SAFI type. This router acts as a route reflector, meaning it distributes routes learned from peers to other peers. This makes it possible for client devices to learn routing information about each other, without establishing BGP peering directly between themselves. The minimum advertisement interval is set to 1, meaning that new routing information will be distributed once per second, if necessary.

The *static-routes* and *next-hop-groups* configuration section define the static routes for reaching 192.168.241.0/24 (via 192.168.254.1) and 192.168.242.0/24 (via 192.168.254.3).

Finally, the *all* policy used for import and export must be defined. It's defined with just a default action of accept, so that all routes are accepted. The routing policy is shown in Listing 11.

```
routing-policy {
    policy all {
        default-action {
            policy-result accept
        }
    }
}
```

Listing 11. Routing policy

5.3.5 Verifying the PoC Functionality

The devices were configured and connected to the network. Laptops were connected to each Layer 2 segment as follows:

- In the DN:
    - MAC address 54:E1:AD:B5:F5:22
    - IPv6 link-local address fe80::f5b9:4c69:6786:89f/64
- Behind the 4G router:
    - MAC address E8:6A:64:8F:FF:AC
    - IPv6 link-local address fe80::ab83:60c8:d9c2:dc63/64
- Behind the 5G router:
    - MAC address 3C:52:82:EA:57:51
    - IPv6 link-local address fe80::262d:a034:98ee:634f/64

IPv6 link-local addresses were used, as since the network is otherwise IPv4 only, and even IPv6 routers can’t route between link-local addresses, there is no other way to transmit IPv6 link-local traffic over this network than successfully working Layer 2 tunnelling. It was successfully verified that each laptop is able to ping each other using the IPv6 link-local addresses, and also it was possible to send TCP and UDP traffic using the iperf3 tool over the network.

On the 7220 IXR-D2 router, it could be seen that the BGP peering was successful by running the *show network-instance lbo-ip-vrf protocols bgp neighbor* command in the running mode, as shown in Listing 12. Some of the columns are omitted from this and the following outputs, as they are wide to fit on the page otherwise.

```
A:srlinux# show network-instance lbo-ip-vrf protocols bgp neighbor
-------------------------------------------------------------------------
BGP neighbor summary for network-instance "lbo-ip-vrf"
Flags: S static, D dynamic, L discovered by LLDP, B BFD enabled, - disabled
-------------------------------------------------------------------------
-------------------------------------------------------------------------
+------------+-----------------+----------+-------+---------+-------------+
| Net-Inst   |       Peer      |   Group  | Flags | Peer-AS |    State    |
+============+=================+==========+=======+=========+=============+
| lbo-ip-vrf | 192.168.241.142 | BGP-EVPN | D     | 65241   | established |
| lbo-ip-vrf | 192.168.242.177 | BGP-EVPN | D     | 65241   | established |
+------------+-----------------+----------+-------+---------+-------------+
-------------------------------------------------------------------------
Summary:
0 configured neighbors, 0 configured sessions are established,0 disabled
peers
2 dynamic peers
--{ + running }--[  ]--
```

Listing 12. BGP Neighbors on 7220 IXR-D2

It was then verified that the IMET routes are learned from the mobile router peers, by the *show network-instance lbo-ip-vrf protocols bgp routes evpn route-type 3 summary* command, as shown in Listing 13.

```
# show network-instance lbo-ip-vrf protocols bgp routes evpn route-type 3
summary
-------------------------------------------------------------------------
Show report for the BGP route table of network-instance "lbo-ip-vrf"
-------------------------------------------------------------------------
Status codes: u=used, *=valid, >=best, x=stale
Origin codes: i=IGP, e=EGP, ?=incomplete
-------------------------------------------------------------------------
BGP Router ID: 172.16.240.1      AS: 65241      Local AS: 65241
-------------------------------------------------------------------------
Type 3 Inclusive Multicast Ethernet Tag Routes
+--------+---------------------+--------+-----------------+---------------
| Status | Route-distinguisher | Tag-ID | Originator-IP   | neighbor
+========+=====================+========+=================+===============
| u*>    | 192.168.241.142:100 | 0      | 192.168.241.142 | 192.168.241.142
| u*>    | 192.168.242.177:100 | 0      | 192.168.242.177 | 192.168.242.177
+--------+---------------------+--------+-----------------+---------------
-------------------------------------------------------------------------
2 Inclusive Multicast Ethernet Tag routes 2 used, 2 valid
-------------------------------------------------------------------------
--{ + running }--[  ]--
```

Listing 13. EVPN type 3 routes received by the 7220 IXR-D2

Next, it was observed that the type 2 routes for the laptops behind the mobile routers were learned as well, as shown by Listing 14.

```
# show network-instance lbo-ip-vrf protocols bgp routes evpn route-type 2
summary
-------------------------------------------------------------------------
Show report for the BGP route table of network-instance "lbo-ip-vrf"
-------------------------------------------------------------------------
Status codes: u=used, *=valid, >=best, x=stale
Origin codes: i=IGP, e=EGP, ?=incomplete
-------------------------------------------------------------------------
BGP Router ID: 172.16.240.1      AS: 65241      Local AS: 65241
-------------------------------------------------------------------------
Type 2 MAC-IP Advertisement Routes
+--------+-----------------------+------------+-------------------+
| Status |  Route-distinguisher  |   Tag-ID   |    MAC-address    |
+========+=======================+============+===================+
| u*>    | 192.168.241.142:100   | 0          | 3C:52:82:EA:57:51 |
| u*>    | 192.168.242.177:100   | 0          | E8:6A:64:8F:FF:AC |
+--------+-----------------------+------------+-------------------+
-------------------------------------------------------------------
2 MAC-IP Advertisement routes 2 used, 2 valid
-------------------------------------------------------------------------
--{ + running }--[  ]--
```

Listing 14. EVPN type 2 routes received by the 7220 IXR-D2

There are multiple other commands that can be used to validate and verify various aspects of the EVPN configuration in SR Linux. Figure 14 shows one of the perhaps more interesting ones, *show network-instance layer2-mac-vrf bridge-table mac-table all*, which can be used to observe the MAC table as learned by the MAC-VRF instance. From this it can be seen that the MAC address 54:E1:AD:B5:F5:22 has been learnt locally by the bridge, while the other ones have been received over BGP from the mobile routers.

```
A:srlinux# show network-instance layer2-mac-vrf bridge-table mac-table all
------------------------------------------------------------------------------
Mac-table of network instance layer2-mac-vrf
------------------------------------------------------------------------------
+-------------------+-------------------------------------------------------+
|      Address      |                     Destination                       |
+===================+=======================================================+
| 3C:52:82:EA:57:51 | vxlan-interface:vxlan1.1 vtep:192.168.241.142 vni:100
| 54:E1:AD:B5:F5:22 | ethernet-1/1.108
| E8:6A:64:8F:FF:AC | vxlan-interface:vxlan1.1 vtep:192.168.242.177 vni:100
+-------------------+-------------------------------------------------------+
Total Irb Macs                 :    0 Total    0 Active
Total Static Macs              :    0 Total    0 Active
Total Duplicate Macs           :    0 Total    0 Active
Total Learnt Macs              :    1 Total    1 Active
Total Evpn Macs                :    2 Total    2 Active
Total Evpn static Macs         :    0 Total    0 Active
Total Irb anycast Macs         :    0 Total    0 Active
Total Proxy Antispoof Macs     :    0 Total    0 Active
Total Reserved Macs            :    0 Total    0 Active
Total Eth-cfm Macs             :    0 Total    0 Active
--{ + running }--[  ]--
```

Listing 15. MAC table of the 7220 IXR-D2’s MAC-VRF

On the mobile routers, similar information can be retrieved from FRR by using the *vtysh* tool. The beginning of the peering information from the 4G router is shown in Listing 16.

```
OpenWrt# show bgp neighbors
BGP neighbor is 172.16.240.1, remote AS 65241, local AS 65241, internal link
  Local Role: undefined
  Remote Role: undefined
 Member of peer-group BGP-EVPN for session parameters
  BGP version 4, remote router ID 172.16.240.1, local router ID
192.168.242.181
  BGP state = Established, up for 00:00:14
  Last read 00:00:10, Last write 00:00:13
  Hold time is 45 seconds, keepalive interval is 15 seconds
  Configured hold time is 180 seconds, keepalive interval is 60 seconds
  Configured conditional advertisements interval is 60 seconds
  Neighbor capabilities:
    4 Byte AS: advertised and received
    Extended Message: advertised
    AddPath:
      L2VPN EVPN: RX advertised
    Long-lived Graceful Restart: advertised
    Route refresh: advertised and received(new)
    Enhanced Route Refresh: advertised
    Address Family L2VPN EVPN: advertised and received
<...>
```

Listing 16. BGP neighbor information on the 4G router

As the IXR acts as a route reflector, the 4G and 5G routers only peer with the 7220 IXR-D2 but are still able to receive all routing information from each other via the 7220 IXR-D2. Listing 17 shows that the 4G router can receive type 3 routes from the 7220 IXR-D2 and 5G router and listing 18 shows the corresponding type 2 routes being received.

```
OpenWrt# show bgp l2vpn evpn route type 3
BGP table version is 6, local router ID is 192.168.242.181
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal
Origin codes: i - IGP, e - EGP, ? - incomplete
EVPN type-1 prefix: [1]:[EthTag]:[ESI]:[IPlen]:[VTEP-IP]:[Frag-id]
EVPN type-2 prefix: [2]:[EthTag]:[MAClen]:[MAC]:[IPlen]:[IP]
EVPN type-3 prefix: [3]:[EthTag]:[IPlen]:[OrigIP]
EVPN type-4 prefix: [4]:[ESI]:[IPlen]:[OrigIP]
EVPN type-5 prefix: [5]:[EthTag]:[IPlen]:[IP]

   Network          Next Hop            Metric LocPrf Weight Path
                    Extended Community
Route Distinguisher: 172.16.240.1:100
 *>i[3]:[0]:[32]:[172.16.240.1]
                    172.16.240.1                 100      0 i
                    RT:65241:100 ET:8
Route Distinguisher: 192.168.241.142:100
 *>i[3]:[0]:[32]:[192.168.241.142]
                    192.168.241.142              100      0 i
                    RT:65241:100 ET:8
Route Distinguisher: 192.168.242.181:100
 *> [3]:[0]:[32]:[192.168.242.181]
                    192.168.242.181                   32768 i
                    ET:8 RT:65241:100

Displayed 3 prefixes (3 paths) (of requested type)
```

Listing 17. EVPN type 3 routes received by the 4G router

```
OpenWrt# show bgp l2vpn evpn route type 2
BGP table version is 6, local router ID is 192.168.242.181
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal
Origin codes: i - IGP, e - EGP, ? - incomplete
EVPN type-1 prefix: [1]:[EthTag]:[ESI]:[IPlen]:[VTEP-IP]:[Frag-id]
EVPN type-2 prefix: [2]:[EthTag]:[MAClen]:[MAC]:[IPlen]:[IP]
EVPN type-3 prefix: [3]:[EthTag]:[IPlen]:[OrigIP]
EVPN type-4 prefix: [4]:[ESI]:[IPlen]:[OrigIP]
EVPN type-5 prefix: [5]:[EthTag]:[IPlen]:[IP]

   Network          Next Hop            Metric LocPrf Weight Path
                    Extended Community
Route Distinguisher: 172.16.240.1:100
 *>i[2]:[0]:[48]:[54:e1:ad:b5:f5:22]
                    172.16.240.1                 100      0 i
                    RT:65241:100 ET:8
Route Distinguisher: 192.168.241.142:100
 *>i[2]:[0]:[48]:[3c:52:82:ea:57:51]
                    192.168.241.142              100      0 i
                    RT:65241:100 ET:8
Route Distinguisher: 192.168.242.181:100
 *> [2]:[0]:[48]:[e8:6a:64:8f:ff:ac]
                    192.168.242.181                   32768 i
                    ET:8 RT:65241:100

Displayed 3 prefixes (3 paths) (of requested type)
```

Listing 18. EVPN type 2 routes received by the 4G router

It's also possible to list all the MACs learned in VNI 100, by using the *show evpn mac vni 100* command. This is illustrated in Listing 19.

```
OpenWrt# show evpn mac vni 100
Number of MACs (local and remote) known for this VNI: 3
Flags: N=sync-neighs, I=local-inactive, P=peer-active, X=peer-proxy
MAC               Type   Flags Intf/Remote ES/VTEP            VLAN  Seq #'s
54:e1:ad:b5:f5:22 remote       172.16.240.1                          0/0
e8:6a:64:8f:ff:ac local        lan1                                  0/0
3c:52:82:ea:57:51 remote       192.168.241.142                       0/0
```

Listing 19. MAC table of VNI 100

Besides FRR, the forwarding database of the vxlan100 pseudodevice can be investigated as well. Listing 20 shows the forwarding database, and also illustrates how the VXLAN destinations are learned dynamically based on the information received by FRR over BGP.

```
root@OpenWrt:/# bridge fdb show dev vxlan100
3c:52:82:ea:57:51 vlan 1 extern_learn offload master br-vni100
3c:52:82:ea:57:51 extern_learn offload master br-vni100
54:e1:ad:b5:f5:22 vlan 1 extern_learn offload master br-vni100
54:e1:ad:b5:f5:22 extern_learn offload master br-vni100
c2:e9:e9:ba:1b:cf vlan 1 offload master br-vni100 permanent
c2:e9:e9:ba:1b:cf offload master br-vni100 permanent
00:00:00:00:00:00 dst 172.16.240.1 self permanent
00:00:00:00:00:00 dst 192.168.241.142 self permanent
3c:52:82:ea:57:51 dst 192.168.241.142 self extern_learn
54:e1:ad:b5:f5:22 dst 172.16.240.1 self extern_learn
```

Listing 20. Forwarding table of the vxlan100 device.

To summarize, it’s been verified that the PoC works as intended. The VXLAN endpoints are able to relay Layer 2 traffic to each other based on the information learned over BGP.

6 Conclusions

This thesis successfully explored the integration of private 4G and 5G networks with BGP EVPN. The Proof of Concept (PoC) developed in this study demonstrated the practical feasibility of employing VXLAN for tunnelling and dynamic EVPN control-plane management.

The study shows that BGP-EVPN with VXLAN can be used to enhance network flexibility and scalability in Layer 2 tunnelling solutions in private cellular networks. Being able to utilize dynamic IP address allocation capability of the mobile routers, using these as router IDs in BGP configurations, showcases the potential of this technology in environments where static IP allocation is impractical. Using off-the-shelf development boards and open-source software for the mobile routers indicates the potential for cost-effective and flexible solutions in productizing this approach for mobile routers.

Looking ahead, it is necessary to perform further verification and stability testing under varying network conditions to ensure the solution's reliability and robustness. Optimizing BGP parameters should be done as well, to ensure network performance and stability. Advanced features like ECMP and multi-homing could be explored to provide redundancy and efficiency. Moreover, there is considerable scope for more seamless integration of BGP with mobile routers than what has been done in the scope of this Proof-of-Concept.

Even as the 5G technological landscape evolves and features like Ethernet PDU get gradually implemented, older 5G and 4G networks and equipment still remain in use, and the diverse needs of enterprises and industries will ensure that EVPN solutions still stay relevant in this scope as well. The adaptability and ongoing development of EVPN technologies ensure their continued significance in the networking ecosystem, complementing emerging technologies and serving as a crucial component in modern industrial and enterprise networking solutions.

References

1. Drake J, Henderickx W, Sajassi A, Aggarwal R, Bitar NN, Isaac A, et al. BGP MPLS-Based Ethernet VPN [Internet]. Internet Engineering Task Force; 2015 Feb [cited 2023 Apr 23]. Report No.: RFC 7432. Available from: https://datatracker.ietf.org/doc/rfc7432

2. Sajassi A, Drake J, Bitar N, Shekhar R, Uttaro J, Henderickx W. A Network Virtualization Overlay Solution Using Ethernet VPN (EVPN) [Internet]. Internet Engineering Task Force; 2018 Mar [cited 2023 Apr 23]. Report No.: RFC 8365. Available from: https://datatracker.ietf.org/doc/rfc8365

3. Creswell JW, Creswell JD. Research Design: Qualitative, Quantitative, and Mixed Methods Approaches. Sixth edition. Thousand Oaks, California: SAGE Publications, Inc; 2022. 320 p.

4. Fall K, Stevens W. TCP/IP Illustrated: The Protocols, Volume 1. 2nd edition. Upper Saddle River, NJ: Addison-Wesley Professional; 2011. 1056 p.

5. Peterson LL, Davie BS. Computer Networks: A Systems Approach. 5th edition. Amsterdam ; Boston: Morgan Kaufmann; 2011. 920 p.

6. Huynh M, Mohapatra P. Metropolitan Ethernet Network: A move from LAN to MAN. Comput Netw. 2007 Dec 5;51(17):4867–94.

7. Honda O, Ohsaki H, Imase M, Ishizuka M, Murayama J. Understanding TCP over TCP: effects of TCP tunneling on end-to-end throughput and latency. In: Atiquzzaman M, Balandin SI, editors. Boston, MA; 2005 [cited 2023 Aug 3]. p. 60110H. Available from: http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.630496

8. Beijnum I van. BGP. 1st ed. Beijing ; Sebastopol, CA: O'Reilly; 2002. 272 p.

9. Holma H, Toskala A, editors. LTE for UMTS: Evolution to LTE-Advanced. Second Edition. Chichester, West Sussex, United Kingdom: Wiley; 2011. 543 p.

10. Schmitt P, Landais B, Yong Yang F. Control and User Plane Separation of EPC nodes (CUPS) [Internet]. [cited 2023 Sep 26]. Available from: https://www.3gpp.org/news-events/3gpp-news/cups

11. Holma H, Toskala A, Nakamura T, editors. 5G Technology: 3GPP New Radio [Internet]. 1st ed. Wiley; 2020 [cited 2023 Oct 5]. Available from: https://onlinelibrary.wiley.com/doi/book/10.1002/9781119236306

12. Nath SV, Simlai A, Kara O. Mastering 5G Network Design, Implementation, and Operations. Livery Place 35 Livery Street Birmingham B3 2PB, UK.: Packt Publishing Ltd.; 2023. 434 p.

13. Hanks JDR, Reynolds H, Roy D. Juniper MX Series: A Comprehensive Guide to Trio Technologies on the MX. 2nd edition. O'Reilly Media; 2016. 1142 p.

14. Li T, Farinacci D, Hanks SP, Meyer D, Traina PS. Generic Routing Encapsulation (GRE) [Internet]. Internet Engineering Task Force; 2000 Mar [cited 2023 Aug 1]. Report No.: RFC 2784. Available from: https://datatracker.ietf.org/doc/rfc2784

15. Farinacci D, Hanks SP, Li T, Traina PS. Generic Routing Encapsulation (GRE) [Internet]. Internet Engineering Task Force; 1994 Oct [cited 2023 Aug 1]. Report No.: RFC 1701. Available from: https://datatracker.ietf.org/doc/rfc1701

16. Farinacci D, Hanks SP, Li T, Traina PS. Generic Routing Encapsulation over IPv4 networks [Internet]. Internet Engineering Task Force; 1994 Oct [cited 2023 Aug 1]. Report No.: RFC 1702. Available from: https://datatracker.ietf.org/doc/rfc1702

17. Dommety G. Key and Sequence Number Extensions to GRE [Internet]. Internet Engineering Task Force; 2000 Sep [cited 2023 Aug 1]. Report No.: RFC 2890. Available from: https://datatracker.ietf.org/doc/rfc2890

18. Pignataro C, Bonica R, Krishnan S. IPv6 Support for Generic Routing Encapsulation (GRE) [Internet]. Internet Engineering Task Force; 2015 Oct [cited 2023 Aug 1]. Report No.: RFC 7676. Available from: https://datatracker.ietf.org/doc/rfc7676

19. IEEE 802 Numbers [Internet]. [cited 2023 Aug 1]. Available from: https://www.iana.org/assignments/ieee-802-numbers/ieee-802-numbers.xhtml

20. Layer-2 GRE Tunnels [Internet]. [cited 2023 Aug 1]. Available from: https://www.arubanetworks.com/techdocs/ArubaOS_8.11.0_Web_Help/Content/arubaos-solutions/nwk-params/l2-gre-tunn.htm

21. EdgeRouter - EoGRE Layer 2 Tunnel [Internet]. Ubiquiti Support and Help Center. [cited 2023 Aug 1]. Available from: https://help.ui.com/hc/en-us/articles/204961754-EdgeRouter-EoGRE-Layer-2-Tunnel

22. Universal TUN/TAP device driver — The Linux Kernel documentation [Internet]. [cited 2023 Aug 2]. Available from: https://www.kernel.org/doc/html/latest/networking/tuntap.html

23. tap [Internet]. [cited 2023 Aug 2]. Available from: https://man.freebsd.org/cgi/man.cgi?query=tap

24. tap(4) - NetBSD Manual Pages [Internet]. [cited 2023 Aug 2]. Available from: https://man.netbsd.org/tap.4

25. An introduction to Linux virtual interfaces: Tunnels [Internet]. Red Hat Developer. 2019 [cited 2023 Aug 2]. Available from: https://developers.redhat.com/blog/2019/05/17/an-introduction-to-linux-virtual-interfaces-tunnels

26. gre(4) - OpenBSD manual pages [Internet]. [cited 2023 Aug 2]. Available from: https://man.openbsd.org/egre.4

27. What Is Open vSwitch? — Open vSwitch 3.2.90 documentation [Internet]. [cited 2023 Aug 28]. Available from: https://docs.openvswitch.org/en/latest/intro/what-is-ovs/

28. Garg P, Wang YS. NVGRE: Network Virtualization Using Generic Routing Encapsulation [Internet]. Internet Engineering Task Force; 2015 Sep [cited 2023 Aug 3]. Report No.: RFC 7637. Available from: https://datatracker.ietf.org/doc/rfc7637

29. Holdrege M, Srisuresh P. IP Network Address Translator (NAT) Terminology and Considerations [Internet]. Internet Engineering Task Force; 1999 Aug [cited 2023 Aug 22]. Report No.: RFC 2663. Available from: https://datatracker.ietf.org/doc/rfc2663

30. Ghein LD. MPLS Fundamentals: Ccie No. 1897. 1st edition. Indianapolis, IN: Cisco Systems; 2006. 626 p.

31. Viswanathan A, Rosen EC, Callon R. Multiprotocol Label Switching Architecture [Internet]. Internet Engineering Task Force; 2001 Jan [cited 2023 Aug 8]. Report No.: RFC 3031. Available from: https://datatracker.ietf.org/doc/rfc3031

32. Tappan D, Rekhter Y, Conta A, Fedorkow G, Rosen EC, Farinacci D, et al. MPLS Label Stack Encoding [Internet]. Internet Engineering Task Force; 2001 Jan [cited 2023 Aug 8]. Report No.: RFC 3032. Available from: https://datatracker.ietf.org/doc/rfc3032

33. Martini L, El-Aawar N, Rosen EC, Heron G. Encapsulation Methods for Transport of Ethernet over MPLS Networks [Internet]. Internet Engineering Task Force; 2006 Apr [cited 2023 Aug 8]. Report No.: RFC 4448. Available from: https://datatracker.ietf.org/doc/rfc4448

34. Townsley M, Wainner S, Seely T, Young J, Pignataro C. Encapsulation of MPLS over Layer 2 Tunneling Protocol Version 3 [Internet]. Internet Engineering Task Force; 2007 Mar [cited 2023 Aug 8]. Report No.: RFC 4817. Available from: https://datatracker.ietf.org/doc/rfc4817

35. Xu X, Sheth N, Yong L, Callon R, Black DL. Encapsulating MPLS in UDP [Internet]. Internet Engineering Task Force; 2015 Apr [cited 2023 Aug 8]. Report No.: RFC 7510. Available from: https://datatracker.ietf.org/doc/rfc7510

36. Rosen EC, Worster T, Rekhter Y. Encapsulating MPLS in IP or Generic Routing Encapsulation (GRE) [Internet]. Internet Engineering Task Force; 2005 Mar [cited 2023 Aug 8]. Report No.: RFC 4023. Available from: https://datatracker.ietf.org/doc/rfc4023

37. G_Nault. How to configure MPLS with tc in the Linux kernel [Internet]. Enable Sysadmin. Red Hat, Inc.; 2021 [cited 2023 Aug 15]. Available from: https://www.redhat.com/sysadmin/mpls-tc-linux-kernel

38. Townsley M, Goyret I, Lau J. Layer Two Tunneling Protocol - Version 3 (L2TPv3) [Internet]. Internet Engineering Task Force; 2005 Mar [cited 2023 Aug 15]. Report No.: RFC 3931. Available from: https://datatracker.ietf.org/doc/rfc3931

39. Valencia AJ, Zorn G, Palter W, Pall GS, Townsley M, Rubens A. Layer Two Tunneling Protocol “L2TP” [Internet]. Internet Engineering Task Force; 1999 Aug [cited 2023 Aug 15]. Report No.: RFC 2661. Available from: https://datatracker.ietf.org/doc/rfc2661

40. L2TP — The Linux Kernel documentation [Internet]. [cited 2023 Aug 16]. Available from: https://www.kernel.org/doc/html/latest/networking/l2tp.html

41. Katz A. NVIDIA to Acquire Networking Software Trailblazer Cumulus [Internet]. NVIDIA Blog. 2020 [cited 2023 Aug 16]. Available from: https://blogs.nvidia.com/blog/2020/05/04/nvidia-acquires-cumulus/

42. Mahalingam M, Dutt D, Duda K, Agarwal P, Kreeger L, Sridhar T, et al. Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks [Internet]. Internet Engineering Task Force; 2014 Aug [cited 2023 Aug 16]. Report No.: RFC 7348. Available from: https://datatracker.ietf.org/doc/rfc7348

43. Maino F, Kreeger L, Elzur U. Generic Protocol Extension for VXLAN (VXLAN-GPE) [Internet]. Internet Engineering Task Force; 2021 Sep [cited 2023 Dec 4]. Report No.: draft-ietf-nvo3-vxlan-gpe-12. Available from: https://datatracker.ietf.org/doc/draft-ietf-nvo3-vxlan-gpe-12

44. Bernat V. VXLAN & Linux [Internet]. 2017 [cited 2023 Aug 22]. Available from: https://vincent.bernat.ch/en/blog/2017-vxlan-linux

45. Virtual eXtensible Local Area Networking documentation — The Linux Kernel documentation [Internet]. [cited 2023 Aug 22]. Available from: https://www.kernel.org/doc/html/latest/networking/vxlan.html

46. vxlan(4) [Internet]. [cited 2023 Aug 22]. Available from: https://man.freebsd.org/cgi/man.cgi?query=vxlan&sektion=4&manpath=OpenBSD+7.2

47. vxlan(4) - OpenBSD manual pages [Internet]. [cited 2023 Aug 22]. Available from: https://man.openbsd.org/vxlan.4

48. Gross J, Ganga I, Sridhar T. Geneve: Generic Network Virtualization Encapsulation [Internet]. Internet Engineering Task Force; 2020 Nov [cited 2023 Aug 24]. Report No.: RFC 8926. Available from: https://datatracker.ietf.org/doc/rfc8926

49. Linux Kernel Driver DataBase: CONFIG_GENEVE: Generic Network Virtualization Encapsulation [Internet]. [cited 2023 Aug 28]. Available from: https://cateee.net/lkddb/web-lkddb/GENEVE.html

50. Lasserre M, Kompella V. Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling [Internet]. Internet Engineering Task Force; 2007 Jan [cited 2023 Sep 5]. Report No.: RFC 4762. Available from: https://datatracker.ietf.org/doc/rfc4762

51. A Border Gateway Protocol 4 (BGP-4) [Internet]. Internet Engineering Task Force; 1995 Mar [cited 2023 Nov 26]. Report No.: RFC 1771. Available from: https://datatracker.ietf.org/doc/rfc1771

52. Rekhter Y, Kompella K. Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling [Internet]. Internet Engineering Task Force; 2007 Jan [cited 2023 Nov 27]. Report No.: RFC 4761. Available from: https://datatracker.ietf.org/doc/rfc4761

53. Chandra R, Bates TJ, Rekhter Y, Katz D. Multiprotocol Extensions for BGP-4 [Internet]. Internet Engineering Task Force; 2007 Jan [cited 2023 Nov 27]. Report No.: RFC 4760. Available from: https://datatracker.ietf.org/doc/rfc4760

54. Scudder J, Chandra R. Capabilities Advertisement with BGP-4 [Internet]. Internet Engineering Task Force; 2002 Nov [cited 2023 Nov 27]. Report No.: RFC 3392. Available from: https://datatracker.ietf.org/doc/rfc3392

55. Rabadan J, Henderickx W, Drake J, Lin W, Sajassi A. IP Prefix Advertisement in Ethernet VPN (EVPN) [Internet]. Internet Engineering Task Force; 2021 Oct [cited 2023 Nov 27]. Report No.: RFC 9136. Available from: https://datatracker.ietf.org/doc/rfc9136

56. Sajassi A, Thoria S, Mishra MP, Drake J, Lin W. Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Proxies for Ethernet VPN (EVPN) [Internet]. Internet Engineering Task Force; 2022 Jun [cited 2023 Nov 27]. Report No.: RFC 9251. Available from: https://datatracker.ietf.org/doc/rfc9251

57. EVPN — FRR latest documentation [Internet]. [cited 2023 Nov 30]. Available from: https://docs.frrouting.org/en/latest/evpn.html

58. gobgp/docs/sources/evpn.md at master · osrg/gobgp [Internet]. GitHub. [cited 2023 Nov 30]. Available from: https://github.com/osrg/gobgp/blob/master/docs/sources/evpn.md

59. Lei J, Munikar M, Suo K, Lu H, Rao J. Parallelizing packet processing in container overlay networks. In: Proceedings of the Sixteenth European

Conference on Computer Systems [Internet]. New York, NY, USA: Association for Computing Machinery; 2021 [cited 2023 Dec 1]. p. 261–76. (EuroSys '21). Available from: https://dl.acm.org/doi/10.1145/3447786.3456241

60. Open vSwitch* with DPDK Overview [Internet]. Intel. [cited 2023 Dec 2]. Available from: https://www.intel.com/content/www/us/en/developer/articles/technical/open-vswitch-with-dpdk-overview.html

61. 39. NVIDIA MLX5 Ethernet Driver — Data Plane Development Kit 23.11.0 documentation [Internet]. [cited 2023 Dec 2]. Available from: https://doc.dpdk.org/guides/nics/mlx5.html

62. Yan Y, Wang H. Open vSwitch Vxlan performance acceleration in cloud computing data center. In: 2016 5th International Conference on Computer Science and Network Technology (ICCSNT) [Internet]. Changchun, China: IEEE; 2016 [cited 2023 Dec 2]. p. 567–71. Available from: http://ieeexplore.ieee.org/document/8070222/

63. Broadcom extends its network chips to 800 Gbps [Internet]. [cited 2023 Dec 4]. Available from: https://www.spglobal.com/marketintelligence/en/news-insights/research/broadcom-extends-its-network-chips-to-800-gbps

64. Trident3-X7 / BCM56870 Series [Internet]. [cited 2023 Dec 4]. Available from: https://www.broadcom.com/products/ethernet-connectivity/switching/strataxgs/bcm56870-series

65. Nvidia Mellanox Spectrum 2 Datasheet [Internet]. [cited 2023 Dec 4]. Available from: https://network.nvidia.com/files/doc-2020/pb-spectrum-2.pdf

66. Marvell Prestera CX Series Product Overview [Internet]. [cited 2023 Dec 4]. Available from: https://service.component-electronics.ru/public/upload/Prestera%2098CX85xx%20Series%20.pdf

67. Solutions - Cisco Silicon One Q202 and Q202L Processors Data Sheet - Cisco [Internet]. [cited 2023 Dec 4]. Available from: https://www.cisco.com/c/en/us/solutions/collateral/silicon-one/datasheet-c78-744311.html

68. Banana Pi BPI-R3 - Banana Pi Wiki [Internet]. [cited 2023 Dec 8]. Available from: https://wiki.banana-pi.org/Banana_Pi_BPI-R3

69. LM940 [Internet]. Telit. [cited 2023 Dec 8]. Available from: https://www.telit.com/devices/lm940-advanced-lte-data-card/

70. MOCHAbin [Internet]. Globalscaletechnologies.com. 2022 [cited 2023 Dec 8]. Available from: https://globalscaletechnologies.com/product/mochabin-copy/

71. FN990Axx [Internet]. Telit. [cited 2023 Dec 8]. Available from: https://www.telit.com/devices/fn990axx/

72. Brown R. Welcome to the OpenWrt Project [Internet]. OpenWrt Wiki. 2016 [cited 2023 Dec 8]. Available from: https://openwrt.org/start

73. Linux Kernel Driver DataBase: CONFIG_TCP_MD5SIG: TCP: MD5 Signature Option support (RFC2385) [Internet]. [cited 2023 Dec 8]. Available from: https://cateee.net/lkddb/web-lkddb/TCP_MD5SIG.html

74. Nokia 7220 IXR-D series Interconnect Routers for SR Linux [Internet]. 2023 [cited 2023 Dec 9]. Available from: https://web.archive.org/web/20230127013147/https://onestore.nokia.com/asset/i/207599