A Study into Scalable Transport Networks for IoT Deployment

Prepared by:

Yandisa Sizamo

SZMYAN001

Department of Electrical Engineering

University of Cape Town

Prepared for:

Dr Daniel Ramotsoela

Department of Electrical Engineering

University of Cape Town

July 2021

Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for the Master of Engineering in Telecommunications

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.

2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in this thesis from the work(s) of other people, has been attributed and has been cited and referenced.

3. This thesis is my own work.

4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof

Name and Surname: Yandisa Sizamo Date: 12 July 2021

Signature… Signed by candidate ..

Acknowledgements

I would like to express my earnestness appreciation to the following contributors of this work:

- I would like to express my heartfelt gratitude to my supervisor Dr Daniel Ramotsoela of the Department of Electrical Engineering at the University of Cape Town for his support throughout the development of this work. His professionalism and guidance when I went wrong and allowing me to independently learn and grow academically while giving his input.

- A special thanks to Andre Kannemeyer for the technical guidance and supervision in the development of the Proof of Concept (PoC). His offering of laboratory facilities, assisting and guiding me towards the goal and unwavering dedication in the success of this work.

- A heat felt gratitude to my family and colleagues who have provided unstinting assistance and support when I was at a point of giving up.

- Most of all, I give all the glory to God for making all this possible.

Abstract

The growth of the internet towards the Internet of Things (IoT) has impacted the way we live. Intelligent (smart) devices which can act autonomously has resulted in new applications for example industrial automation, smart healthcare systems, autonomous transportation to name just a few. These applications have dramatically improved the way we live as citizens. While the internet is continuing to grow at an unprecedented rate, this has also been coupled with the growing demands for new services e.g. machine-to machine (M2M) communications, smart metering etc.

Transmission Control Protocol/Internet Protocol (TCP/IP) architecture was developed decades ago and was not prepared nor designed to meet these exponential demands. This has led to the complexity of the internet coupled with its inflexible and a rigid state. The challenges of reliability, scalability, interoperability, inflexibility and vendor lock-in amongst the many challenges still remain a concern over the existing (traditional) networks.

In this study, an evolutionary approach into implementing a “Scalable IoT Data Transmission Network” (S-IoT-N) is proposed while leveraging on existing transport networks. Most Importantly, the proposed evolutionary approach attempts to address the above challenges by using open (existing) standards and by leveraging on the (traditional/existing) transport networks.

The Proof-of-Concept (PoC) of the proposed S-IoT-N is attempted on a physical network testbed and is demonstrated along with basic network connectivity services over it. Finally, the results are validated by an experimental performance evaluation of the PoC physical network testbed along with the recommendations for improvement and future work.

Table of Contents

List of Figures

Abbreviations

API	Application Interface
BEB	Backbone Edge Bridge
BGP	Border Gateway Protocol
B-MAC	Back-Bone-MAC
CLNS	Connectionless Network Service
CMHC	Client-MAC-Hierarchy
FEC	Forward Equivalence Class
FIB	Forwarding Information Base
IGP	Interior Gateway Protocol
IoT	Internet of Things
IP/MPLS	Internet Protocol/Multi-Protocol Label Switching
IS-IS	Intermediate System to Intermediate System
LER	Label Edge Router
LLDP	Link Layer Discovery Protocol
LP-WAN	Low Power Wide Area Network
LSDB	Link State Database
LSR	Label Switch Router
MIB	Management Information Base
MPLS-TE	Multi-Protocol Label Switching-Traffic Engineering
NB-IoT	Narrow Band IoT
NNI	Network-to-Network Interfaces
OSPF	Open Shortest Path First
PoC	Proof of Concept
RSVP	Resource Reservation Protocol
SDN	Software Defined Networking
SD-WAN	Software Defined Wide Area Network
S-IoT-N	Scalable Transport Network for IoT Deployment
TCP/IP	Transmission Control Protocol/Internet Protocol
TLV	Type Length Values
UNI	User-to-Network Interface
Wi-Fi	Wireless-Fidelity

Chapter 1: Introduction

1.1 Research Motivation

Today, IoT devices occupy almost every facet of our lives, (smart-homes, smart cities, automated industrial environments) and so on. Smart sensors connected to the Internet are deeply ingrained in our daily lives, monitoring of the environment for example, temperature sensors, video surveillance, motion detection, smart-metering etc. In some cases, the information collected is used to make critical decisions for example shutting down power to a pump or elevator through actuators and so on. The implication of this is that today's environment is disseminated by silent objects which are tightly linked and silently control the way we live.

Many aspects of our daily lives have been revolutionized by these disseminated objects and they have indeed impacted our social, economic, health and education. At the same time, the Internet continues to grow at an unprecedented rate with a continuously growing number of smart devices and its users. This has resulted in an explosion of new service demands for example, smart metering, enhanced industrial automation, improved security, Artificial Intelligence (AI), and Machine-to-Machine (M2M) communications.

Recent research [1] predicts that the number of connected devices will reach 100 billion by 2025. Technologies such as Narrow Band IoT (NB-IoT) [2] or Long Range (LoRa) [3] are widely used in long-distance transmission for IoT. These typical IoT applications include remote meter reading, smart parking systems, smart water metering, environmental monitoring and so on. These applications are characterized by low data bandwidth, high latency, and large coverage range.

This continued growth of new services in the Internet along with the continuing traffic growth is due to the availability of new information, innovative ways of doing things and the adoption of open standards for example SDN based Open-Flow in telecommunications networks.

Transmission Control Protocol/Internet Protocol (TCP/IP) architecture was developed decades ago and was not prepared nor designed to meet these explosive demands. This has led to the complexity of the Internet coupled with its inflexible and a rigid state. The challenges of reliability, scalability, interoperability, inflexibility and vendor lock-in amongst the many challenges still remain a concern over the existing (traditional) networks. These characteristics of the Internet have created high barriers to entry and a slowing down adoption process of new network architectures and technologies and subsequently services required to enable innovation in the Internet.

These and many other challenges are deeply ingrained characteristics of the Internet. Software Defined Networking (SDN) [4] on the other hand enables network simplicity by using open standards with the aim of introducing flexibility and vendor neutrality. SDN works hand in hand with Network Function Virtualization (NFV) [5] of which they both have introduced disruptive innovation in the networking space which is posing threats and challenges in the adaptation of traditional networks. The complementary functioning of SDN/NFV results in a flexible, simpler and open networking standards and subsequently cheaper and more innovative ways of running telecommunications networks while at the same time enabling longer-term benefits.

In this study, an evolutionary approach into implementing a “Scalable IoT Data Transmission Network” (S-IoT-N) is proposed while leveraging on existing transport networks. Most Importantly, the proposed evolutionary approach attempts to address the challenges of reliability, scalability, interoperability, inflexibility and vendor lock-in by using open (existing) standards and by leveraging on the (traditional/existing) transport networks.

The proposed “S-IoT-N” leverages on the existing and matured transport network technologies in particular the Internet Protocol/Multiprotocol Label Switching (IP/MPLS) by integrating the ongoing innovations in an effort to further entrench the use of open standards. In essence, the S-IoT-N is based on innovative network transmission techniques while at the same time addressing the challenges of scalability, reliability, interoperability and vendor lock-in.

1.2 Problem Description

During the initial research phase, it was found that implementing these innovations will greatly simplify the integration of new networking solutions and could potentially solve the major issues associated with rigidity, complexity, inflexibility and vendor lock-in characteristics associated with these traditional transmission networks. This can be achieved by the introduction of new and relevant solutions in the networking space together with innovative approaches in the provisioning and management of these transport networks. Conclusively, the realization of the implementation of scalable transmission networks still remains as a challenge due to the concerns in overcoming the complications associated with scalability, reliability and interoperability in TCP/IP and IP/MPLS technologies.

Most importantly, in order to realize a scalable transport network for IoT deployment, the following problems must be solved.

1. Firstly, an approach into implementing an intergratable and a scalable transport network for IoT deployment using the existing transport infrastructures like IP/MPLS, TCP/IP or SDN as key enabling technologies must be identified.

2. Secondly, how to successfully address the concerns of scalability, interoperability, reliability and flexibility in the proposed S-IoT-N architecture.

The above problems must be solved in order to realize a scalable transport network that will enable easier, simplified and quicker deployment for IoT services.

1.3 Research Objectives

The objective of this study is to realize a scalable transport network able to accommodate the needs of today's growing IoT needs or applications which will solve the current issues associated with vendor lock-in, complexity, inflexibility and lack of scalability while at the same time enabling innovation in the provisioning of IoT services. From this main objective, the following can be derived:

1. Identification of related and current research work in the field of networking relating to TCP/IP, IP/MPLS, SDN, IS-IS and Access Network Technologies.
2. To identify the current (existing) and relevant transport networks and their architectures. Identify their benefits, challenges and concerns which will enable the study in determining areas of improvement and future work.
3. The proposal of an approach and a suitable implementation strategy that will result in the realization of the main research objective of this study.

4. Address the known issues and concerns in the traditional network technologies.
5. Demonstrate a Proof of Concept (PoC) implementation on a physical network testbed along with simplified provisioning of basic services over it.
6. Validate the performance of the proposed S-IoT-N testbed.

1.4 Research Questions

From the above research objectives, the more fundamental questions to be foregrounded are:

1. What would be the correct approach together with its strategy that would result in the effective implementation of the proposed solution which will holistically address the challenges and stated concerns while utilizing the existing network infrastructures?

From the above main question, the following sub-questions can be derived:

2. What are the relevant technologies and architectures that can support and guide in the development of the solution that will realize the study objective?
3. What is the current research work relating to the implementation of transport networks that will guide this study in realizing its objectives?
4. What are the current challenges together with their associated benefits which relate to the existing network architectures and technologies which are in accordance with the research objectives of this study?

1.5 Research Scope

This study focuses primarily on solving challenges on a single domain network architecture and does not consider any underlying physical transport infrastructure e.g. fibre/microwave or any other variables of it. The solution, however, should be scalable to multi domain networks. Most importantly, the proposed S-IoT-N architecture should be able to operate on any type of network infrastructure due to its flexible overlay technology which does not consider any underlying infrastructure.

1.6 Related Work

As far as the problem description of this study is concerned, the related work and research must focus on existing or already attempted solutions that will address the objective of implementing the S-IoT-N model on existing network architectures like IP/MPLS, TCP/IP or SDN and its variants (e.g. SD-WAN). Accordingly, the related work is discussed in the next paragraphs and detailed in Chapter-2.

A study by Casado *et al.* [6] revealed major setbacks in OpenFlow based SDN architecture which were related to the complexities when implementing network address mapping and the related inflexibility at the network core and edge. They then proposed a hybrid approach that would apply the insights of the underlying existing transport technologies (TCP/IP/MPLS) and SDN. SDN proposes open source standards which introduces simplified hardware (white boxes) and open source software along with a demarcation between the network core and its edge. The hybrid approach by Casado proposes the idea of “network fabrics” which involves implementing the network core as a simple network fabric while pushing the complexities towards the network edge for example the processing of IP-packets. According to this study, this approach would result in the implementation of a flexible network core and the independent evolution of both network core and edge. Figure1-1 shows a simplified diagram depicting the idea of a network fabric they proposed.

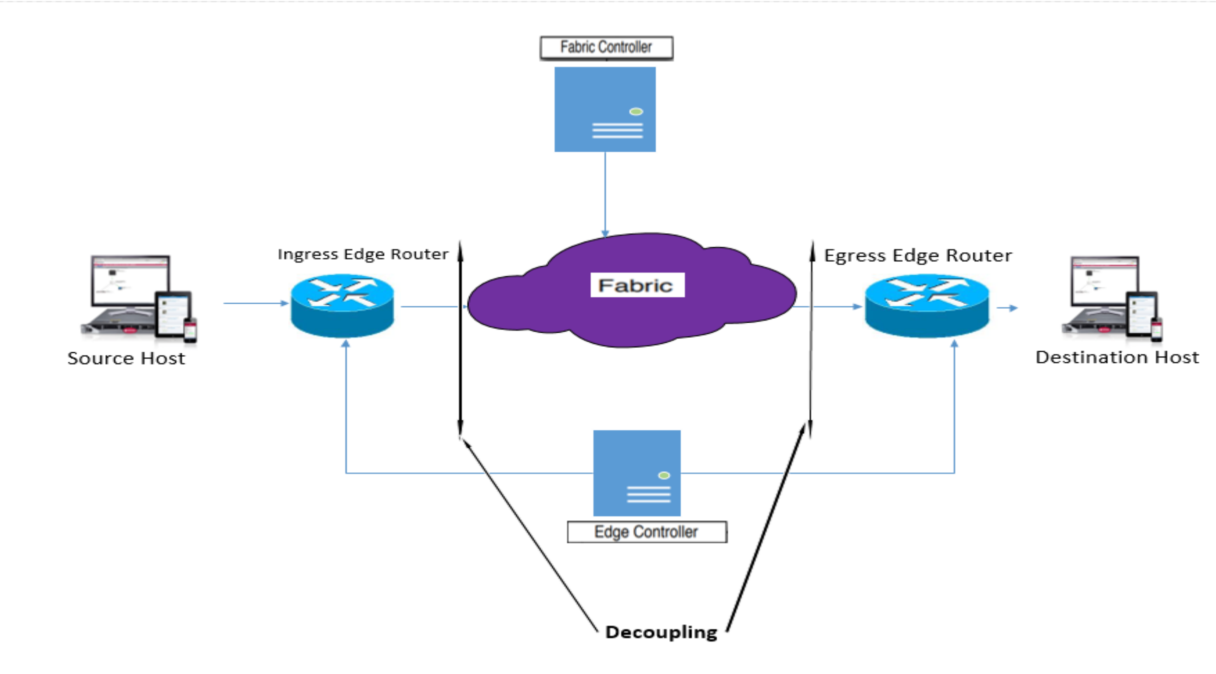


Figure 1-1 The proposed network fabric design by Casado *et al.* [6].

Also, figure 1-1, shows a decoupling of both the network core and the edge which results in their independent evolution.

In another related study by Hampel *et al.* [7] they proposed the approach for extending the SDN concept to the networks domain in order to overcome the short challenges of over dependence on vendor solutions which are specific, rigid, complex and inflexible and are purposefully designed for their specific equipment.. Their approach motivated for replacement network edge gateways with open source SDN nodes (white boxes) in a phased approach and equipping the white boxes with capabilities of IP fundamental operations (vertical forwarding). However, their approach did not facilitate any independent network architecture evolution as it was based only on tunnelling solutions which were based on TCP/IP.

Mogul *et al*. [8] proposed DevoFlow which is a modification to the OpenFlow model aimed at overcoming the reliability and scalability challenges over OpenFlow based SDN solutions. DevoFlow is aimed at reducing the network overhead in at the south-bound API (switch-controller) communication during the flow setup and while gathering statistics. DevoFlow also involves the separation of centralized visibility from centralized control while delegating those decisions to the data plane which results in the overall network efficiency and reliability.

Gouveia *et al.* [9] complemented the above studies by proposing a framework which would implement OpenFlow based SDN solutions which would efficiently and effectively manage and provision network connectivity services. They validated their study by a network testbed implementation which included an experimental performance evaluation. The study recommendations were that the framework needed to be further optimized and adapted accordingly so as to be effective in the existing network architectures.

However, their study did not consider other alternative implementation approaches and strategies, but it showed that the OpenFlow based SDN solution can improve and optimize most of the existing (traditional) and complicated control plane implementations such as MPLS Traffic Engineering and MPLS-based VPNs [10]. Fundamentally, none of these studies attempted to address the challenges and concerns in terms of flexibility, scalability and reliability.

Thus, it can be concluded that the above related work mainly focused on addressing the concerns over recent transport technologies like SDN and not the other underlying traditional transmission infrastructures (TCP/IP/MPLS). Most importantly, the studies do not present implementation approaches of the proposed solutions which in our case is essential for realizing the (S-IoT-N).

1.7 Thesis Structure

The organization of this study is tailored to answer the research questions in a sequential and a step-by-step manner. As such, it is accordingly structured as follows:

Chapter-2 introduces and discusses the relevant and existing network technologies and architectures along with their associated benefits, challenges and concerns. Chapter-3 proposes the evolution approach together with its strategy which is introduced and discussed with the goal of realizing the proposed S-IoT-N which is built over the existing transport networks TCP/IP/MPLS, or SDN. Chapter-4 discusses the PoC implementation of the proposed S-IoT-N architecture which is demonstrated in a physical testbed along with provisioning of basic network connectivity services over it. It also covers performance evaluation and validation. Chapter-5 concludes the study with the recommendations for future work.

Chapter 2: Relevant Architectures and Technologies

Unquestionably, IoT has had a very high impact and will continue to impact on several aspects of our everyday-life. Its capability of embedding intelligent behaviour in the "Things" resulting in "smart" as it is often referred to in IoT references such as smart home, smart city, smart car and so on. IoT integrates the increased computing power and memory capacity of embedded devices to implement intelligent behaviour on these devices in such a way that they can communicate with each other and independently make decisions such as shutting down the elevator, a pump, a power plant etc. In this chapter, we discuss the relevant technologies and architectures along with their benefits and challenges.

The study looks at the enabling technologies that will aid in the implementation of the proposed S-IoT-N Architecture. As discussed in Chapter-1, we take a look on existing work relating to the relevant network architectures. We investigate relevant architectures together with their benefits, challenges, and concerns relating to these network technologies and architectures which are in accordance with the main research question of this study. The chapter concludes by paving a way to Chapter-3 which proposes the implementation approach and its strategy to implement the proposed "S-IoT-N". The relevant topics for this study are structured as follows:

- Transmission Control/Internet Protocol (TCP/IP) (2-1)
- Multi-Protocol Label Switching (IP/MPLS) (2-2)
- Software Defined Networking (SDN) (Section 2-3)
- Wireless Access Networks (Section 2-4).
- Intermediate System to Intermediate System (IS-IS) Section 2-5.
- Link Layer Discovery Protocol (LLDP) Section 2-6

2.1 Transmission Control/Internet Protocol (TCP/IP)

The term TCP/IP is largely used to describe the Internet suite of protocols. Transmission Control Protocol and Internet Protocol (TCP/IP) work together to form the Internet. Internet protocols can be used to communicate across any set of interconnected networks. They are well suited for local-area network (LAN) and wide-area network (WAN) communications. The TCP/IP suite not only includes lower-layer specifications (such as TCP and IP), but also specifications for such common applications as SMTP (e-mail), Telnet (terminal emulation), and FTP (file transfer). Thanks to the standardization and co-operation of the internet users which has made the internet one of the most crucial resources of our time.

The roots of TCP/IP go back as early as the 1970s from ARPANET. It is a set of communications protocols which define the networking model for the Internet and service provider networks. IP's task is the delivery of packets from the source host to the destination host based on their addresses. In order to achieve this, IP uses methods and structures for putting addressing information (tags) within packets/datagrams, a process called encapsulation. Routers which perform switching and data forwarding and are responsible for the delivery of packets from source to destination. A typical packet forwarding process in a router is as follows:

- Upon receipt of a packet, the router analyses the address field of the packet. It then performs a routing look-up by consulting its routing table which can be either created statically or through a dynamic routing process. In the case of static routing, the set of paths towards destination prefixes is manually configured on every router in the network. In the case of a dynamic routing process, the routing table (Forwarding Information Base or FIB) is created by the Routing Information Base (RIB) which is created by the dynamic routing process algorithms for example Shortest Path First (SPF) [11]. This allows the router to identify the output interface where the IP-packet will be forwarded according to the contents of the Forwarding Information Base (FIB). The FIB of an IP router contains a map/topology of the network which assists the router to make forwarding decisions based on what is contained in the FIB.

Thus, IP-packet forwarding depends upon a router-to-router (hop-by-hop) transmission sequence which is based on the identification of the next router's interface on the path towards the final destination.

Internet Assigned Numbers Authority (IANA) is a global internet co-ordinating body that assigns IP-addresses and Autonomous Systems (AS) numbers to organizations across the world. An organization is allocated a unique AS number together with corresponding unique IP addresses. The organization (service provider) will allocate these IP-addresses to its collection of switches and routers. The AS number which is unique enables identification of the service provider in the internet (global) domain, AS is sometimes called Border Gateway Protocol (BGP) domain this is shown in Figure 2.1.

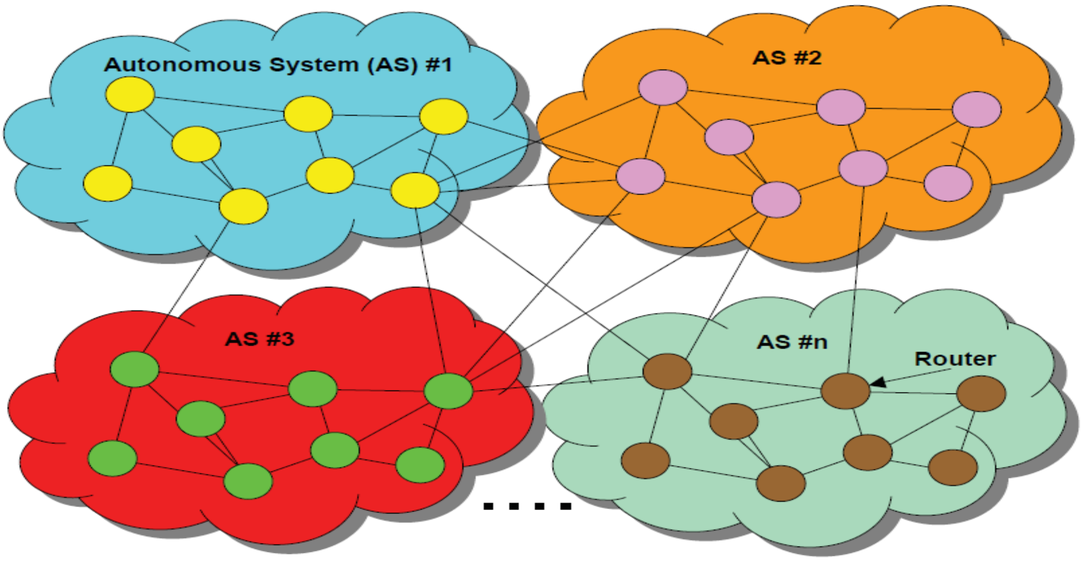


Figure 2-1 The organization of the Internet into Autonomous Systems.

TCP/IP is designed as a layered architecture which groups protocols into layers based on their functionality. The layers are separated from each other through the process of encapsulation. TCP/IP model has four architectural layers being: Link layer, the Internet layer, Transport layer and the Application layer with its architecture shown in Figure 2-2.

TCP/IP ARCHITECTURE					
	Application Layer				
	HTTP	Telnet	DNS	SNMP	SMTP
	Transport Layer				
	TCP		UDP		
	Internet Layer				
	ICMP		IP	IGMP	
	Link Layer				
	Ethernet		ARP	IEEE 802.x	

Figure 2-2 TCP/IP Architecture [2].

The bottom layer in the TCP/IP architecture is the link layer which is responsible for transmission and reception of IP datagrams at the physical layer. The link layer governs the local area networking methods and rules (protocols) for example address resolution protocol (ARP). In essence, TCP/IP architecture was designed to be independent of the underlying layer, access mechanisms, technology and protocols. It is however found in the PoC that this is not the case since a failure at the bottom layer affects above layers meaning the top layer is dependent on the bottom layer. Moreover, TCP/IP assumes an unreliable link layer. Thus, the link layer may constitute any type of LAN and WAN technology.

The Link layer provisions and prepared datagrams for the next layer (Internet layer). The Internet layer is responsible for addressing, encapsulating, and routing packets to their destination. The commonly used protocols of the Internet layer are Control Message Protocol (ICMP), Internet Group Management Protocol (IGMP) and the Internet Protocol (IP).

The Transport layer is responsible for end-to-end communication. In order to achieve this, it is equipped with protocols such as Transmission Control Protocol (TCP) and the User Datagram protocol (UDP). These protocols enable the provisioning of datagrams and communication services to the next layer (Application Layer). TCP and UDP enable a more reliable connection-oriented service.

The application layer enables applications on hosts to exchange of information with other host applications. Common protocols applicable to this layer enable this co-ordination and exchange of information. Examples of application layer protocols include the Simple Mail Transfer Protocol (SMTP), File Transfer Protocol (FTP), Hypertext Transfer Protocol (HTTP) and Telnet to name a few. Most importantly, TCP/IP suite has been the major enabler for the Internet's global success in terms of its widespread adoption, scalability, reliability

Finally, the global adoption of the TCP/IP model which resulted in its huge success can be attributed to its basic layered architectural principles, common addressing scheme and open standards based protocols [12], [13].

However, beside the stated benefits, the TCP/IP model, has quite a few drawbacks being lower layer dependency, lack of effective and efficient network control and management, highly complicated protocol stack and a high signalling overhead. Moreover, TCP/IP's best effort service model is no longer suitable to most of the real-time applications. Although IETF proposed quality of service (QoS) frameworks such as IntServ and DiffServ to enable better

QoS support for the internetwork traffic, they only provide short-term benefits at an expense of more complex protocols. These limitations of the TCP/IP model lead to the growing complexity of the Internet as users try to introduce solutions that will solve today's application demands [14].

These deeply ingrained challenges of inflexibility, scalability, rigidity and the complex nature of TCP/IP makes it unsuitable to cater for today's explosive demands brought by IoT applications. During its development decades ago, these explosive applications were not envisaged and therefore, different approaches are necessary in order to cater for these new demands. TCP/IP is slow and unsuitable for M2M applications because of its high signalling overhead (router-to-router communication). By the time a signalling message of failure is detected in a "remote" manufacturing plant controlled by a distant server/machine, it's too late and a catastrophe would have resulted already. IoT sensor applications require fast, flexible and reliable communication. The S-IoT-N must attempt to address this drawback of TCP/IP. In the next session we explore developments/work that has been done in trying to overcome the drawbacks in TCP/IP. Multiprotocol Label Switching was introduced to try and mitigate the challenges in TCP/IP networks, the next section explores MPLS in an attempt to identify these mitigations.

2.2 Internet Protocol/Multi-Protocol Label Switching (IP/MPLS)

IP/MPLS [15-17] networks make use of IP and MPLS protocols which introduce efficiency and quality of service guarantees, hence the informal use of the term IP/MPLS. In MPLS, packets are forwarded based on labels, each IP network that is reachable through an interface is assigned a unique label. A router configured to run the MPLS protocol will establish a mapping between an incoming label and an outgoing label. This is maintained in the Label Forwarding Information Base (LFIB) table (similar operation as the establishment of FIB table discussed in section 2-1). Each node examines the incoming label, does a table lookup, swaps the incoming label for the outgoing label and then forwards the packet out of the outgoing interface.

In an MPLS network, each packet gets labelled on entry by the ingress router which is known as the Label Edge Router (LER). Furthermore, the LER is responsible for determining the entire Label Switched Path (LSP-tunnel) the packet will take until it reaches the last MPLS domain LER which will strip off the MPLS header and forward the packet as IP-packet to the destination address.

At the MPLS core, routers perform only a label switching function and therefore termed Label Switch Routers (LSRs) because they simply perform packet forwarding based only on those MPLS labels, they don’t look beyond the IP header. This results in fast switching and network efficiency. This efficient switching at the core enables a quick delivery of the MPLS packets to the egress (LER) which removes the MPLS labels (penultimate popping) and forwards the original IP packet toward its final destination.

MPLS uniquely defines itself by the use of labels. Operating between L2/3 (informally termed Layer 2.5), the MPLS label is a four-byte field which pre-defines the MPLS packet's forwarding path (LSP) in an MPLS domain. MPLS labels also carry quality of service (QoS) information indicating a packet's priority level. The experimental (Exp) field in the MPLS header (Figure 2-3) enables the implementation of QoS which make it possible to implement service level agreements (SLAs). MPLS also enables implementation of Class of service (COS) if the service provider offers classified services. It does this by using one or a combination of three protocols that perform label distribution of MPLS labels:

- Label Distribution Protocol (LDP) [16].
- Resource Reservation Protocol (RSVP) [18].
- Border Gateway Protocol (BGP) [19].

Fixed length labels employed on the forwarding table look-ups results in faster switching which gives MPLS advantage over (traditional TCP/IP networks. For the end-to-end QoS support, MPLS classifies the incoming packets into Forwarding Equivalence Classes (FECs) based on a predefined packet header match rules and their ingress ports where each FEC is associated with a class of service and a Label Switched Path (LSP) across the network [14]. Furthermore, MPLS employs existing IP routing protocols while being independent of link layer and network layer routing protocols, which is why MPLS is commonly referred as protocol independent encapsulation.

MPLS is enabled by an encapsulation of a 32-bit header is between layer 2 and layer 3 header fields in a TCP/IP packet, which is why MPLS is commonly referred as operating at Layer 2.5. This header (Figure 2-3) consists of a label stack with a value greater than or equal to one. The MPLS label can be used for various objectives, for example to create LSP tunnel, LSP failure protection, creation of Virtual Private Network (VPN) tunnels, implement QoS support, etc.

In Figure 2-3, the MPLS header format, which is described in [15].

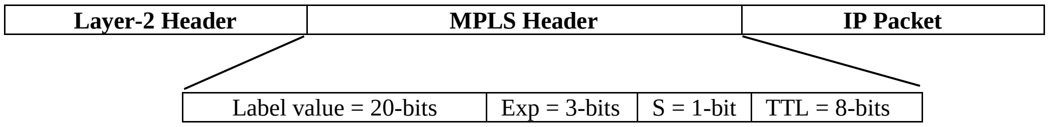


Figure 2-3: MPLS header.

Figure 2-3 shows the MPLS header which is a 4-byte header consisting of:

- A 20-bit which is a label field.
- A 3-bit traffic class (Exp) field which defines the packet's class of service (COS), this field enables implementation of quality of service functions.
- A 1-bit S (Bottom of Stack) It is set to value 1 only if the particular header must interact with the next protocol (i.e. when penultimate popping at the edge LER must be done) and also to indicated whether the header is the last header before the payload.
- 8-bit TTL (Time To Live) field defines the packet's life duration. This is important to ensure packets do not loop continuously in the network which will result in network loops.

For label switching, MPLS uses a push, swap and pop technique to swap labels as the MPLS tagged packets transit the network. In a MPLS domain, the strict name for routers is "Label Switch Routers" (LSRs). An LSR that resides at the edge of a MPLS domain is called a "Label Edge Router" (LER). LERs reside at the entry and exit points of an MPLS domain and are respectively termed ingress and egress LSRs. Where MPLS LERs reside in a service provider domain, they are known as PE (Provider Edge) routers, and core MPLS LSRs are termed P (Provider) routers. During an MPLS operation, when an LSR receives a packet, it performs one or more of the following actions:

- Push: Adds a label, this is typically performed by the ingress router.
- Swap: Replaces a label, this is usually performed by LSRs between the ingress and egress routers.
- Pop: Removes a label, this is most often done by the egress router.

Figure 2-4 shows an example MPLS label switching operation.

Reading Figure 2-4 from left to right, the edge LSR receives a packet for network destination 10.0.0.0/24, it inserts the MPLS header discussed above and imposes label 25 on it and forwards the frame to the LSR residing in the MPLS backbone (in practice there is a number of these in a live network). The LSR swaps label 25 with label 34 and forwards the frame. This process can continue on several LSRs until the last LSR residing at the edge of the MPLS domain (penultimate LER) which will remove the label and additionally strips off the MPLS header (penultimate popping) and then forwards the IP packet to the destination address 10.0.0.0/28.

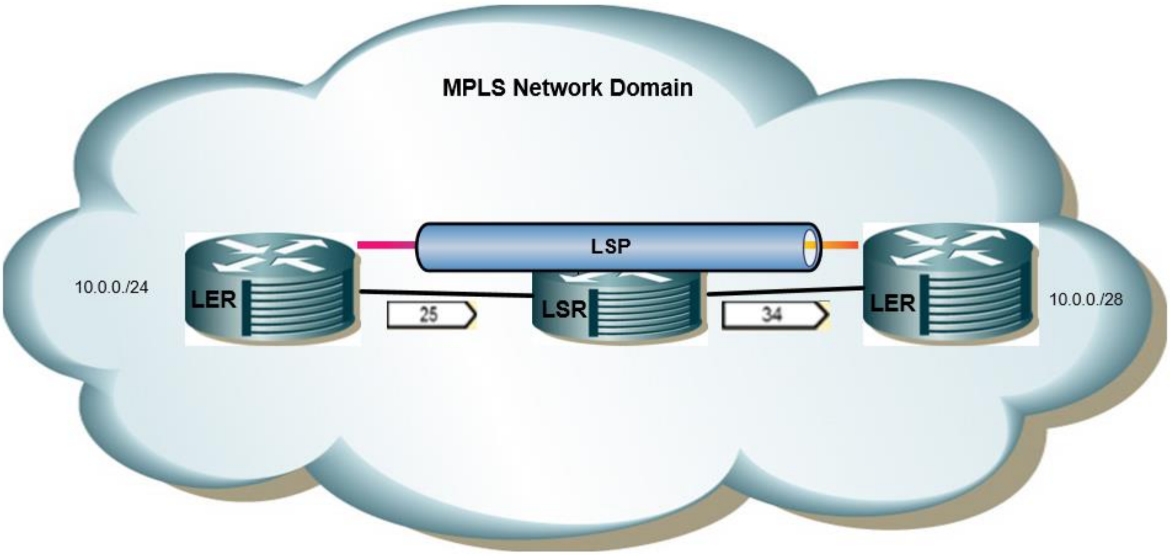


Figure 2-4: Example of MPLS Label Switching Operation.

LSRs and LERs in MPLS can perform both label switching and IP routing. Labels are used as indexes into the LFIB by the LSRs, the LFIB specifies the next hop and a new label for the specific packet. At each LSR, the incoming label is replaced with the new outgoing label as the packet is forwarded to the next hop. When the packet arrives at the last LSR (LER) the egress LER does penultimate hop popping (PHP) which strips the label and forwards the IP-packet to its final destination based on the IP packet header. Moreover, LER nodes also have an additional label to IP prefix binding table called the Label Information Base (LIB) along with an IP forwarding table (FIB) which work together for label binding of IP packets at ingress LER and label removal at the egress LER. These additional tables assist with layer-3 route lookup of IP packets at the egress LER [14].

During LSP setup, the LSR assigns an incoming label to the LSP for the corresponding FEC and informs the next node on the upstream path about the assigned label. These labels have only local significance and are swapped locally with the next node being informed of the new label. As a result of these label distributions, the forwarding tables (LFIBs) are created at the individual LSRs of a MPLS domain. Label Distribution Protocol (LDP) [16] is responsible for path setup and enables label distribution and signalling. LDP and Resource Reservation Protocol-Traffic Engineering (RSVP-TE) [18] are the most used label distribution protocols in MPLS. The advantages of LDP are its simplicity and quick path setup, whereas RSVP-TE is used for its good QoS (traffic engineering) support through its signalling protocol [14]. Combining these two protocols results in the advantages that MPLS offers over traditional TCP/IP networks. Explicit routing offers several advantages over other MPLS routing techniques as it establishes LSPs based on policy and QoS requirements. It can also provision LSPs that can be used for failure protection. This makes explicit routing a preferred choice for LSP route setup. Furthermore, MPLS integrates the advantages of layer 3 routing (scalability and reliability) with that of layer 2 packet forwarding (fast switching in hardware) to enable several benefits like fast switching, traffic engineering, and provisioning of customizable network services (e.g. layer 2 and 3 VPNs) with end-to-end QoS support [14].

In conclusion, MPLS provides flexibility in networks as it enables capabilities of simplified switching hardware with a clear distinction between the edge and the network core. This distinction provides a simplified core that is decoupled from the complex edge functions in order to facilitate the independent evolution of both. However, besides the stated benefits of MPLS, it has a few limitations because of it’s static, domain-specific, high signalling overhead, and non-application aware nature. Most of these limitations are due to the lack of dynamic network control and management in such networks. Software-defined and driven networking (e.g. OpenFlow [20], Path Computation Element (PCE) [21] are being proposed to effectively address and solve these MPLS challenges. However, these concepts are encountering some technical challenges and adoption concerns in terms of scalability, reliability and interoperability and these need to be addressed first in order to successfully realize these new developments.

It is clear that great efforts have been employed in MPLS which are meant to overcome the challenges associated/experienced in TCP/IP networks. As stated above, even though great work has been deployed in MPLS, it still has its drawbacks which are “scalability and signal overhead” which are not suitable for IoT applications/deployment. This study takes cognisance of these drawbacks and attempts to address these in the implementation of the S-IoT-N.

In the next session we explore developments/work that has been done in new innovations such as Software Defined Networking.

2.3 Software Defined Networking (SDN)

The demands brought by IoT has generated modest traffic which has increased the load on data transmission networks. In response to this network capacity demands coupled by an increase in network device performance such as routers and switches. Software-Defined Networking (SDN) architecture proposes the separation (decoupling) of the control plane from the data plane. The control plane is programmable by an external controller via an open interface (Application Plane Interface (API)). This allows open applications which eliminate vendor lock in which is dominant in traditional networks. The basic components of the SDN architecture are the SDN controller and the SDN switch.

The controller resides in the control plane and the switch resides in the data plane. The interface between the two planes is defined by open standards protocols for example Open-Flow [22]. Since the interfaces are open standards-based, the controller can control devices from multiple vendors which can eliminate the vendor lock-in challenge experienced in (traditional) networks. SDN architecture (Figure 2-5) proposes three-layers:

- The “Application Plane/Layer” is where the Network Service Applications reside. Orchestrators, Traffic Steering Applications (TSA), and Security Applications are defined to operate at this layer.
- The “Control Plane/Layer” or the SDN Control Layer, which receives requests from the Application Layer via the Northbound API and sends control request messages to switches in the data plane via the Southbound API.
- The “Data Plane/Layer” forwards user data packets from input ports to output ports in the network switches based on the instructions of the SDN Control Layer where the “SDN Controller” resides.

The motivation to adopt SDN is because of its ability to implement services faster, reliable and most importantly, elimination of vendor lock-in. In most traditional networks (MPLS/TCP/IP in particular), rolling out a new instance of an existing service takes a long time, requires lots of co-ordination and collaboration, and the nightmare of manual configuration. This heavy manual configuration can be mostly automated in SDN. The control functions from many separate nodes with diverse vendor interfaces are condensed into a centralized SDN controller thus simplifying the data plane into simple forwarding switches. In general, there is no single (available) best solution of SDN that suits and fits all as each of them have their own benefits and trade-offs.

Thus, it is up to the network owner to decide on a particular SDN solution or combination of solutions that will suit the need. Figure 2-5 shows a logical view of the basic SDN architecture proposed by the Open Networking Foundation (ONF) [23].

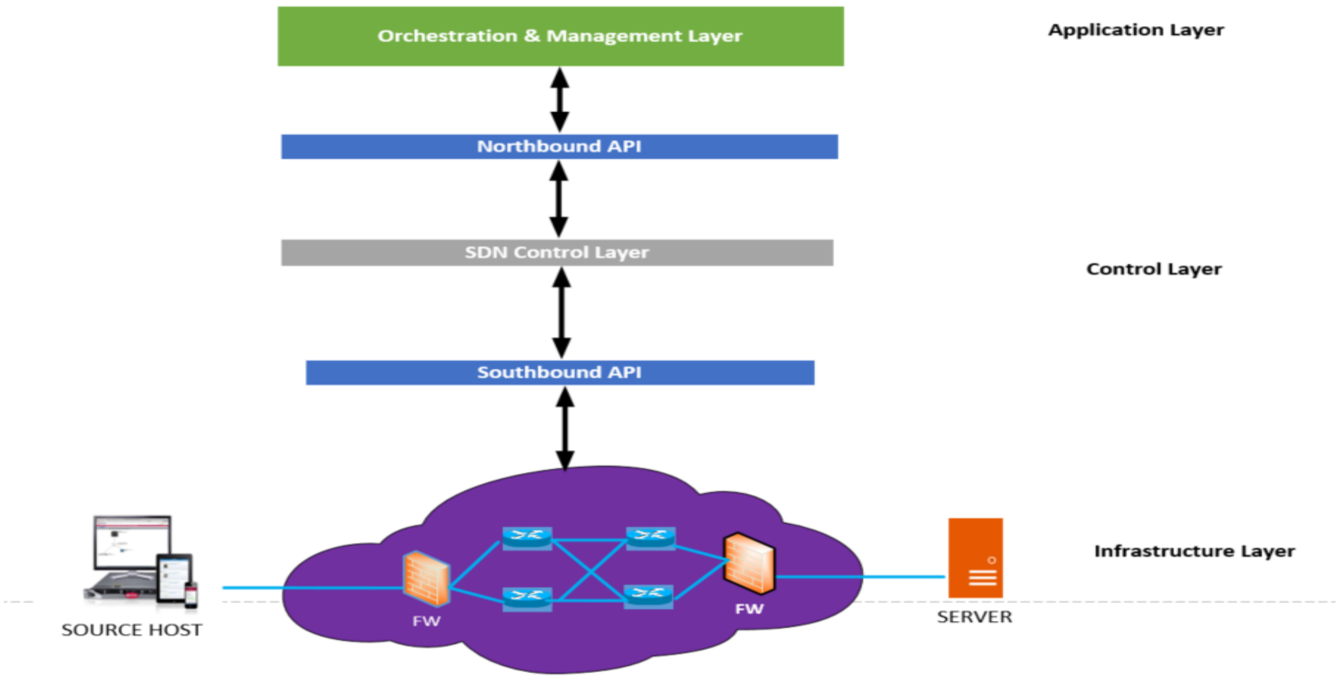


Figure 2-5: A logical view of SDN Architecture

The SDN control plane is decoupled from the underlying data plane (switches/firewalls forming the infrastructure layer) through the process of abstraction. This decoupling enables the centralization and a fully-programmable software platform. Moreover, it introduces innovation which can happen independently in each layer. The control logic interacts with data plane by making control decisions by using open source control protocols (e.g. Open-Flow protocol) via the South Bound API. Moreover, this control software is provisioned and exposed through the Northbound APIs as a set of on-demand, customizable, and innovative network services to other internal and external applications like business software, orchestration software, and security policy engines. [14]

One of the key principles of SDN is centralized control, this enables a holistic network view which allows the controller to make adjustments to the network topology based on real-time events and service demands. In addition to this, the separation of the control and forwarding planes combined with virtualized networking functions enables evolution and hardware upgrades of the networking devices independent of the controller. One other advantage is the ability to use white-box (dumb boxes) networking devices which eliminates the dependency on vendor specific equipment. Lastly, the use of APIs between the application plane and control plane is a key principle that enables the benefit of network programmability. Currently, APIs are used for the Southbound interface between the SDN controller and the forwarding planes while in general there is no standard for the Northbound API between the application and the control layer.

2.3.1 SDN Configuration and Control

SDN supports the concept of a configuration protocol and a control protocol for the switches. Each of these protocols is used in the Southbound API. Simple Network Management Protocol (SNMP) [25], Network Configuration Protocol (NETCONF) [26] and OFConfig [24] are examples of configuration and network management protocols. NETCONF and OFConfig both use the YANG [27] data modelling language to create a model of the data that will be configured on the network device. Examples of control protocols used are OpenFlow, PCE as some of the examples. These protocols are used to connect flows through the switches (from ingress to egress). It is important to realize that although OpenFlow is often used in discussing SDN, it is possible to use other protocols to send control messages to the switches in the SDN network. Figure 2-6 shows typical south bound API configuration and control protocols.

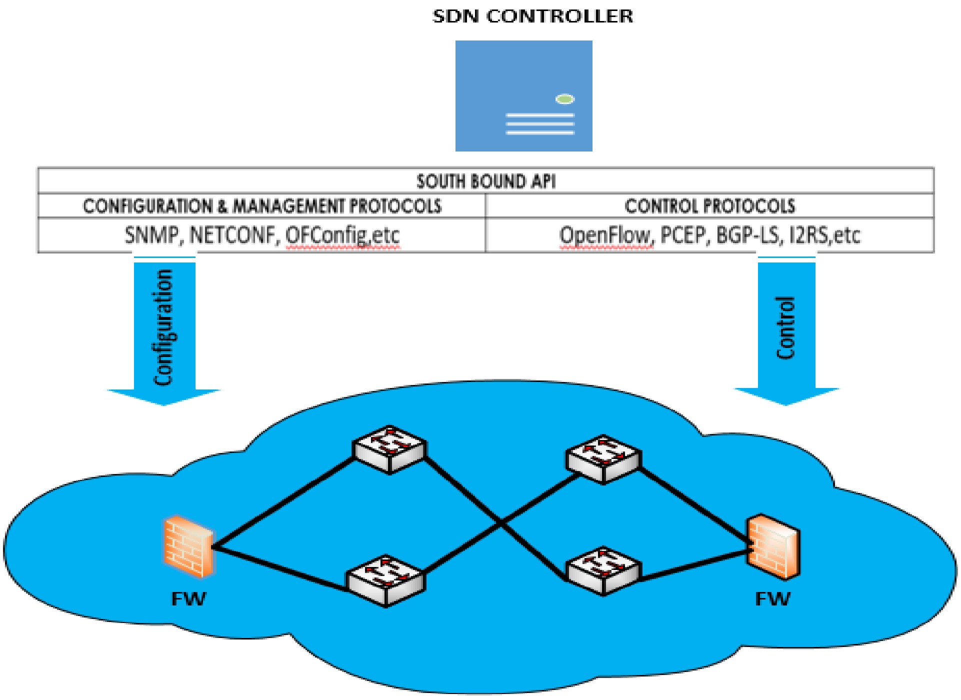


Figure 2-6: Open-Flow South Bound API Configuration and Control Protocol Examples

2.3.2 Open-Flow

Open-Flow is the most widely used SDN protocol because of its user driven and flexible nature and is very much linked to SDN discussions because the value it adds to its innovations. While the Open-Flow protocol is often discussed in conjunction with SDN, it’s important to note that Open-Flow is one of the many protocols used in SDN deployment.

SDN can be better illustrated by considering a centralized controller from the forwarding plane and enabling this controller to program the network topology dynamically via the use of any South Bound API protocol (e.g OpenFlow) as shown in Figure 2-7.

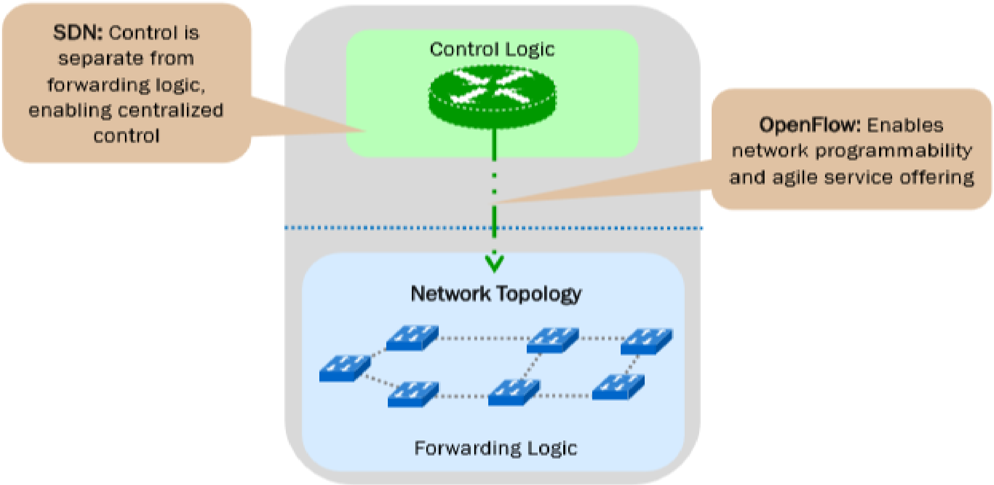


Figure 2-7 Illustration of South Bound API implementation (e.g. Open-Flow).

ONF promotes Open-Flow protocol as its preferred Southbound interface standard. Moreover, ONF maintains and regularly updates the Open-Flow protocol and the switch specifications including the forwarding plane which is responsible for the forwarding of traffic in the data plane based on the rules programmed into the flow tables by the control plane. A logical view of an Open-Flow switch is shown in Figure 2-8. with its specifications found in [22] and [23].

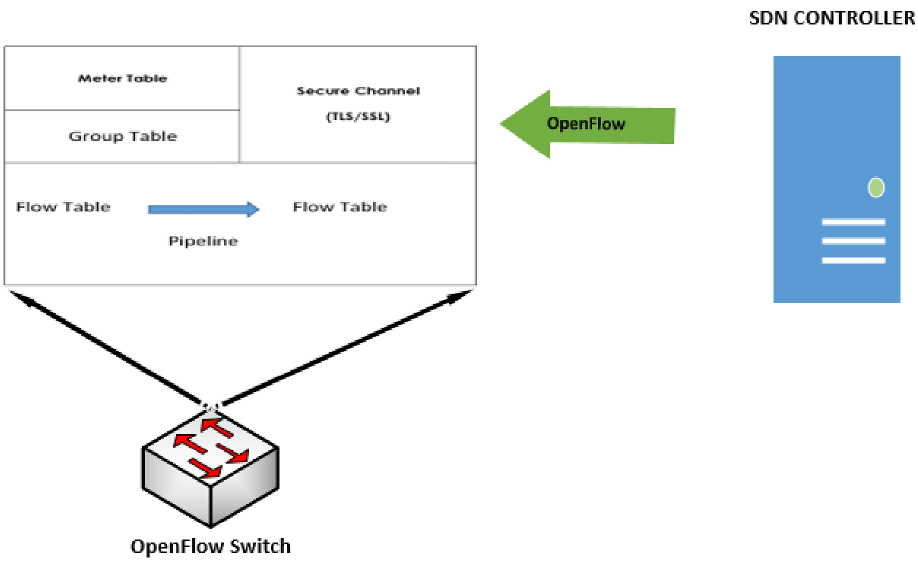


Figure 2-8: Open-Flow Switch Logical View.

The three main components of the Open-Flow switch shown in Figure 2-8 are:

(1) A Flow table, Group table, and the Meter table. The tables describe how to process and/or forward an incoming packet based on rules given by the controller.

(2) A secure channel which can employ either of the security mechanisms (TLS/SSL) secures communication between the switch and the controller which manages the switch and controls its traffic flows by adding, updating and deleting corresponding flow entries in flow tables both reactively (upon packet/flow arrival) and proactively [14].

(3) Open-Flow protocol, which defines communication between the switch and the controller.

Figure 2-9 shows the flow table implemented by the Open-Flow protocol in an Open-Flow switch with the fields described below:

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie

Figure 2-9: Open-Flow Table.

Match Field: To match incoming packets. These consist of the ingress port and packet headers, and optionally metadata.

Priority: describes the matching precedence of a flow entry.

Counters: describes the matched packets to a flow entry and is updated when packets are matched.

Instructions: modify the action set or pipeline processing.

Timeouts: Maximum amount of time or idle time before flow is expired by the switch

Cookie: Maybe used by a controller to filter out flow entry data for example chosen by the external controller.

When a packet is received at an input port, it is matched against the flow table entries. The highest priority flow entry that matches the packet is selected. The counters associated with the selected flow entry are updated and the actions associated with the entry are applied. If there is no entry found in the flow table that matches the packet, or with all match fields this is called a “table-miss” flow entry. The priority of this entry is the lowest (i.e. zero). If there is no table-miss flow entry by default packets that are unmatched, by flow entries are dropped.

Open-Flow also involves pipeline processing which makes use of a group table to which packets are directed by flow entries in flow tables during the pipeline processing to trigger additional methods of forwarding for example flooding, multipath, or link aggregation which is an additional layer of forwarding directionless incoming traffic flows. Each group entry in a group table consists of a group identifier, a group type, counters and action buckets (buckets containing set of actions to apply to matching packets) this is shown in Figure 2-10.

Group Identifier	Group Type	Counters	Action Buckets

Figure 2-10 Group entry main components.

Finally, the Open-Flow pipeline processing involves a meter table (Figure 2-11) to which packets are directed by flow entries in flow tables during pipeline processing to trigger various performance related (QoS) actions on incoming traffic flows. A meter table consists of meter entries that define per-flow meters which enable Open-Flow to implement various QoS operations such as rate limiting and traffic prioritization. A meter measures the rate of packets assigned to it and the rate of those packets. Meters are attached directly to flow entries as opposed to queues which are attached to ports. Any flow entry can specify a meter in its instruction set. Multiple meters can be used in the same table in an exclusive way and also on the same set of packets by using them in successive flow tables.

Figure 2-11 shows a meter table consisting of a meter identifier, meter bands and counters, the meter bands specify various performance specific packet processing types and rates and each meter band is further identified by its band type, meter identifier, and counters as shown in Figure 2-11.

Meter Identifier	Meter Bands	Counters

Figure 2-11: Components of a meter table.

The developments in SDN attempt to introduce open networking innovations that overcomes the concerns of vendor lock-in and inflexibility which is dominant in the (traditional) networks. The introduction of the white boxes (dumb-switches) concept in the data plane introduces the advantage of cheap and affordable equipment which makes it affordable to deploy a network that is affordable run. The ability of ONF to create a labing/open platform which affords the innovations in the development of open protocols like Open-Flow makes SDN a way to go as this results in the development of affordable software which is introduced by innovative ideas.

In this study, advantages brought by these innovations are attempted in the PoC while also capitalizing on the existing advantages in traditional networks.

2.4 Wireless Access Networks

Technologies such as Narrow Band IoT (NB-IoT) [28] or Long Range (LoRa) [29] are used as access technologies in IoT. Typical IoT applications include remote meter reading, smart parking, smart water, environmental monitoring etc. These applications are characterized by low data bandwidth, high latency and large coverage range. A Wi-Fi network naturally features high bandwidth, secure transmission in a LAN and low latency. Therefore, it is an inevitable and feasible way to carry IoT data on a Wi-Fi network. Currently, many vendors have proposed their IoT APs that have built-in Bluetooth [30] or ZigBee [31] modules to transmit IoT device data. For fast deployment of IoT networks, a combination of wired and wireless access networks is a requirement as these access networks connect smart devices to the end systems doing data processing/analytics. In the PoC of this study, an IoT device used accesses the transport network wirelessly. We take a brief discussion on typical wireless access technologies that are commonly used in IoT deployment.

2.4.1 Bluetooth technology

Developed by Ericsson 1994, Bluetooth is a radio technology that supports short-distance device communication (often within 10 m). It allows wireless information to be exchanged between a variety of devices. In recent years, the Bluetooth standard has continuously evolved and expanded to meet the needs of the Internet of Things with the introduction of Bluetooth 5.0 [30] and Bluetooth mesh networking. The broadcast topology available on Bluetooth Low Energy (BLE) enables indoor positioning and location services. For example, beacon-based indoor navigation and way-finding solutions have been introduced, as GPS does not provide good indoor coverage. Beacon-based solutions are also being deployed in enterprises for the purpose of space utilization. The mesh topology on BLE is optimized for creating large-scale device networks, making it particularly well suited for emerging markets in need of a reliable wireless solution to establish large-scale control, monitoring, and automation systems [30].

Bluetooth 5.0 significantly improves the transmission distance, transmission rate, and improves power consumption. Wireless Access Points (APs) of many vendors have built-in Bluetooth modules, which can be used for Bluetooth positioning and geo-fencing when implementing IoT applications.

2.4.2 ZigBee

ZigBee technology is a low data rate, low power consumption, low cost, wireless networking protocol targeted towards automation and remote control applications. It resolves the defects in Bluetooth technology which include high complexity, high power, and small network scale. ZigBee networks have been widely used in various fields such as home automation, home security, industrial field control, environment control, medical care, and transportation. In IoT applications, ZigBee technology is applied in IoT gateways for example in a smart home.

ZigBee attempts to provide low cost and low power connectivity for equipment that needs

battery life that could last for several months to several years but does not require data transfer rates as high as those enabled by Bluetooth. This makes it suitable for smart metering for example since these devices send readings once in a while and should have batteries that should last for a long time. It operates in the un-licenced band spectrum (2.4GHs globally) with data rates of 250kbps at 2.4GHz. A unique feature of ZigBee network layer is communication redundancy eliminating which single point of failure in mesh networks [31].

In comparison, ZigBee works like Bluetooth but is simpler, has a lower data rate and spends most of its time idling. This characteristic means that a node on a ZigBee network should be able to run for six months to two years on just two AA batteries. The operational range of ZigBee is 10-75m compared to 10m for Bluetooth. ZigBee data rate is well below Bluetooth data rate (250Kbps @ 24GHz compared to Bluetooth's 1Mbps in the same frequency range) [31]. Bluetooth's protocol is more complex since it is geared towards handling voice, images and file transfers. The wakeup time for a ZigBee node is around15 milliseconds compared to a wake-up time of 3-seconds in Bluetooth.

2.4.3 Radio Frequency Identification (RFID)

Radio Frequency Identification (RFID) is a form of Automatic Identification and Data Capture (AIDC) technology that uses electric or magnetic fields at radio frequencies to transmit information. An RFID system can be used to identify many types of objects, such as manufactured goods, animals, and people. Each object that needs to be identified has a small object known as an RFID tag affixed to it or embedded within it. The tag has a unique identifier and may optionally hold additional information about the object. Devices known as RFID readers wirelessly communicate with the tags to identify the item connected to each tag and possibly read or update additional information stored on the tag. This communication can occur without any line of sight and over greater distances. RFID technologies support a wide range of applications from asset management and tracking to access control and automated payment to name a few.

RFID uses radio signals to identify a specific target and execute instructions on the identified device. It does not require mechanical or optical contact between the card reader system and a specific target. An RFID reader identifies and sends the product information (passive tag) stored in a chip through induced currents. Alternatively, a tag actively sends the signal (active tag) of a certain frequency and the reader reads and decodes the information and sends it to the central information system for data processing. RFID has been widely used in IoT scenarios such as asset management, access control, and parking lot. Its application market is growing with the development of IoT applications.

Every RFID system includes a radio frequency (RF) subsystem, which is composed of tags and readers. In many RFID systems, the RF subsystem is supported by an enterprise subsystem that is composed of middleware, analytic systems, and networking services. RFID systems that share information across organizational boundaries, such as supply chain applications, also have an inter-enterprise subsystem. Each RFID system has different components and customizations so that it can support a particular business process for an organization, as a result, the security risks for RFID systems and the controls available to address them are highly varied. The enterprise and inter-enterprise subsystems involve common IT components such as servers, databases, and networks and therefore can benefit from typical IT security controls for those components [32].

2.4.4 Light Fidelity (Li-Fi)

Developed by German physicist Harald Hass and his team, Light Fidelity (Li-Fi) is a technology for wireless communication between devices using visible light (such as light from light bulbs) to transmit data. Li-Fi can transmit data by using light from light emitting diodes (LEDs) and provide lighting and wireless networking without causing electromagnetic interference. Data is transmitted in light with high security. Its main limitations include environmental interference and reverse communication between terminals and light sources.

Li-Fi works on the principle of Visible Light Communication (VLC). It uses the visible spectrum as well as ultraviolet and infrared radiations. It allows the data to be transferred by modulating the intensity of the light, which is then received by a photo-sensitive detector. The light signal is then demodulated into electronic form. The simple idea behind this technology is that when the LED is ON (digital 1) the data is transmitted and when it is OFF (digital 0) no data transmission takes place. The modulation of intensity of light is so fast it cannot be perceived by human eye and it appears constant to human. It is hundred times faster than Wi-Fi and it provides speeds in Gigabits per second (Gbps). The data is transmitted through illumination of special LED bulbs and using a photo detector to detect the signal. These kinds of bulbs serve both purposes for transmitting data and also for providing lights. This kind of communication is very cheap and easy to maintain. It is already being used in various fields like military operations, underwater divers, aircrafts etc. [33]

A brief comparison between Li-Fi and Wi-Fi which can be found in [33] is discussed below:

- Data transmission in Li-Fi takes place in the form of light, LED bulbs to be precise while data transmission in Wi-Fi takes place in form of radio waves.
- The technology used in Li-Fi is the present IrDA compliant devices, whereas the technology used in Wi-Fi is WLAN 802.11 a/b/g/n/ac/ad (and now “ax” Wi-Fi-6) standard compliant devices.
- Li-Fi can be applied in airlines, explorations beneath the sea, hospitals, offices, schools, libraries for fast browsing and data communication while Wi-Fi is mainly used for internet browsing with the help of Wi-Fi hotspots.
- Since in case of Li-Fi, light gets obstructed due to any physical object so greater security can be achieved, but in case of Wi-Fi different technologies need to be implemented for further security.

- The data transfer speed of Li-Fi is about 1Gbps whereas Wi-Fi transmits with the speed of 150 Mbps with WLAN and up to 2Gbps with WiGig/Giga_IR technology.
- Li-Fi can cover up distances of up to 10 meters, whereas Wi-Fi has network coverage of 32 meters.
- The major components to implement a Li-Fi system are an LED driver and photo detector while the major components that makes up a Wi-Fi system are: devices (laptop, desktop), router and access point installation. [33]

2.4.5 Wi-Fi HaLow™

In early 2016, the Wi-Fi Alliance announced a new technology Wi-Fi HaLow which aims to master the raising Internet of Things market with its tremendous number of devices. This technology brings revolutionary changes to Wi-Fi, improving transmission reliability and power efficiency in scenarios with thousands of sensor stations being connected to a single access point. [34]

Wi-Fi HaLow is based on the IEEE 802.11ah standard and it addresses low-power, long-range wireless data transmission. It operates in the 900 MHz frequency band, which is lower than the 2.4 GHz and 5 GHz frequency bands of today's Wi-Fi. Wi-Fi HaLow offers low power, long range of up to 1 km, and strong signal strength with less interference. The technology's low power consumption competes with LoRa, ZigBee, Bluetooth. It uses sub 1 GHz (S1G) license-exempt bands allowing the creation of large groups of stations or sensors to provide extended range Wi-Fi networks and support Internet of Things (IoT) applications. [35]

Table 2-1 shows comparison of the various wireless access network technologies with their suited applications which were briefly discussed in this section.

Wireless Access Technologies							
	Bluetooth	ZigBee	RFID	NFC	Li-Fi	Wi-Fi "HaLow"	Wi-Fi
Bandwidth	1-24-Mbps	250-kbps	1 Mbps	106/212/424 –kbps	≥ 10 Gbps	4-Mbps	10-Gbps
Frequency	2.4 Ghz	2.4 Ghz	125-Khz, 13.56 MHz, 433 MHz, 2.4 GHz	13.56 MHz	Optical	900 MHz	2.4 GHz, 5 GHz
Advantages	Low power consumption & low cost	Low power consumption, Self-organizing and low cost	Fast Read & Write speed & good penetration	Low power consumption, quick setup & high security	High speed & High security	Low power consumption, long distance & better penetrability	High transmission rate, simple deployment & low cost
Disadvantages	Short distances & Poor security	Low speeds & poor Security	No standards & poor security	Short Tx distances, low speeds & poor security	RF obstruction& poor RF signalling	Low speeds	Low penetrability
Applications	Short distance RF applications (home, commercial and limited industrial use)	Short distance RF applications, home automation, and industrial filed control & medical care	Assert management, Access Control, parking etc.	Mobile payments	Mainly scientific research	Smart phone industrial control	Home, commercial, industrial etc.
Distance	50 m	10 – 100 m	10 cm – 10 m	20 cm	10 m (light dependent)	100 m – 1 km	50 – 200 m

Table 2-1 Various wireless access network technologies with their suited applications

The few access technologies briefly discussed in this section reveal the potential of the exponential increase in the applications and the demands brought by IoT applications. It is clear that the TCP/IP designed decades ago is unable nor suitable to cope with these current demands. Today's IoT applications require instant communication and therefore continuous improvement/innovation is necessary in the network environment.

2.5 Intermediate System to Intermediate System Routing Protocol (IS-IS)

IS-IS was designed for Connectionless Network Service (CLNS). It is a link state routing protocol that also uses Dijkstra's Shortest Path First (SPF) algorithm (discussed in section 2-1). In IS-IS, the SPF algorithm helps to create the Link State Database (LSDB) which enables each router in the IS-IS domain to have identical topology as other routers in its area. IS-IS refers to a router as an intermediate system (IS) and routing is from IS to another IS hence the IS-IS acronym. Each router is identified by a Network Service Access Point (NSAP) address and uses CLNS to maintain adjacencies and build SPF trees/topologies. Furthermore, there is also an integrated version of IS-IS which accommodates other protocols such as IP. IS-IS uses hello messages to establish adjacencies and Label Switched Paths (LSPs) to exchange link-state information. IS-IS can be sub-divided into two routing levels:

Level-1 routing which occurs within an IS-IS area. Here, routers are identified by their local area and routing table in each router is built according to this. All devices within level-1 routing have the same area address. Routing within an area is accomplished by looking at the locally significant address portion known as the system ID and the lowest cost path is chosen.

Level-2 routers learn the locations of other routing areas and build an inter area routing table. All routers in a level-2 routing area use the destination area address to route traffic using the lowest cost path. To support the routing levels, IS-IS define three types of routers:

Level-1 routers learn their routes within their intra-area

Level-2 routers between the areas (Inter-area)

Level-1&2 routers learn about routes both within and between areas

The router connecting Level-1 and Level-1&2 routers is called the backbone. Figure 2-12 shows the topology of IS-IS routing.

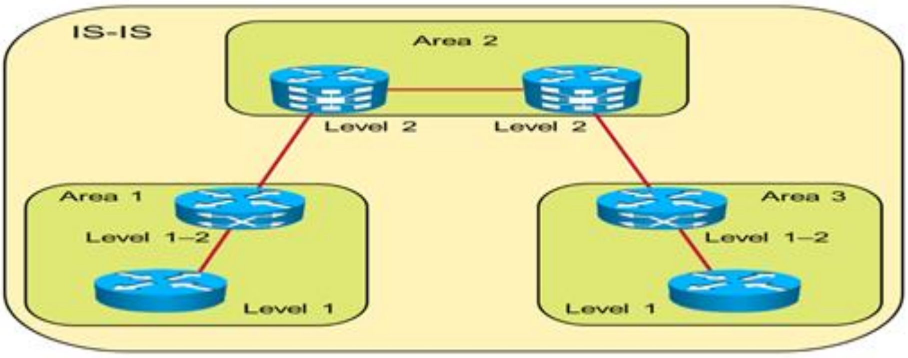


Figure 2-12: IS-IS Topology

Connectionless Network Service (CLNS) Addresses

CLNS addresses used by routers are called Network Service Access Points (NSAP) addresses. Unlike IP addresses, NSAP addresses apply to the entire nodes not interfaces. NSAP address is equivalent to the combination of the IP address and upper layer protocol in an IP header. A typical NSAP header (Figure-2-13) consists of:

- Authority and Format Identifier (AFI)
- 4-byte Area ID
- 6-Byte System ID and a
- A 2-byte NSAP Selector (NSEL)

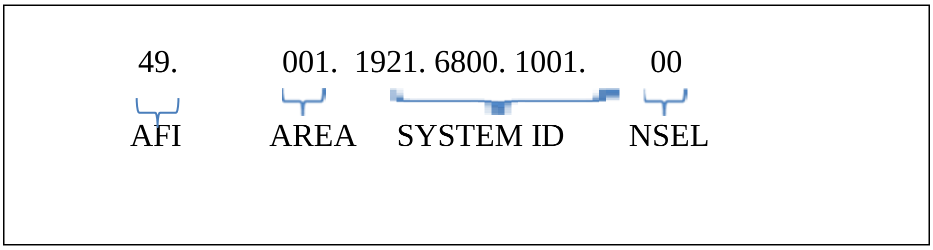


Figure 2-13 NSAP Address Scheme

2.6 Link Layer Discovery Protocol (LLDP)

In an IoT network, various devices (e.g. a camera and its back-end server) must be connected together. It becomes a big task to manually configure network routes so as to create an end-to-end connection between two devices in a network so that they are able to communicate. Considering the scale of IoT devices to be deployed, it becomes nearly impossible to efficiently and quickly deploy and let alone manage this type of a network.

LLDP (IEEE 802.1AB) [36] is an open standard link layer (OSI layer-2) protocol used by network devices for advertising their identity, capabilities, and neighbours on a local area network based on IEEE 802 technology. Information gathered with LLDP can be stored in the device Management Information Base (MIB) and queried with SNMP. The topology of an LLDP-enabled network can be discovered by querying the hosts databases. This enables the discovery and updating of information such as system information, ports, configured VLANs, IP addresses and so on. The enhancement LLDP-Media Discovery End Point (LLDP-MED) enables:

- Auto-discovery of LAN configuration/policies such as VLANs, Differentiated services (Diffserv) settings which enable the possibility of plug and play services.
- Device location discovery to allow creation of location databases.
- Extended and automated power management of Power over Ethernet (PoE) end points and
- Inventory management which allows network administrators to track network devices and determine their characteristics.

LLDP information is sent by devices from each of their interfaces at a fixed interval, in the form of an Ethernet frame. Each frame contains one LLDP Data Unit (LLDPDU). Each LLDPDU is a sequence of type-length-value (TLV) structures. The Ethernet frame used in LLDP typically has its destination MAC address set to a special multicast address. Each LLDP frame starts with a mandatory Type Length Value consisting of a Chassis ID, Port ID, and Time-to-Live. The mandatory TLVs are followed by any number of optional TLVs. The frame ends with a special TLV named end of LLDPDU in which both the type and length fields are 0.

In this study, IS-IS together LLDP are used extensively in automating the configuration for IoT services. They overcome the challenges of scalability by automating service setup which results in instant deployment of IoT services. Furthermore, they enable the automation of the manual configuration which is a requirement in traditional networks (TCP/IP/MPLS) to activate services. This automation therefore plays a critical role in scaling service deployment which makes it possible to implement a much scalable network.

Having discussed the “Relevant Technologies and Architectures”. The challenges, advantages, future improvements have been noted. Chapter-3 discusses “The proposed Evolutionary Approach” in an attempt to introduce ideas that will result in the proposed solution “S-IoT-N” which is demonstrated in Chapter-4.

3. Proposed Evolutionary Approach and Implementation Strategy

The PoC in Chapter-4 will demonstrate that the traditional MPLS network is too cumbersome to configure, manage and maintain. A requirement for automation and a simplified network is necessary which can be achieved by implementing an overlay functionality while leveraging on the existing infrastructures. This chapter presents and discusses the proposed implementation approach together with its strategy in accordance with the thesis problem statement and its objectives

Most importantly, the proposed evolutionary approach, can be implemented incrementally instead of upgrading (changing) the entire network architecture with disruptive automation. The implementation strategy facilitates an incremental deployment scenario incorporating co-existence of the new solution with the existing network technologies (TCP/IP/MPLS). Figure 3-1 shows a generic IoT architecture.

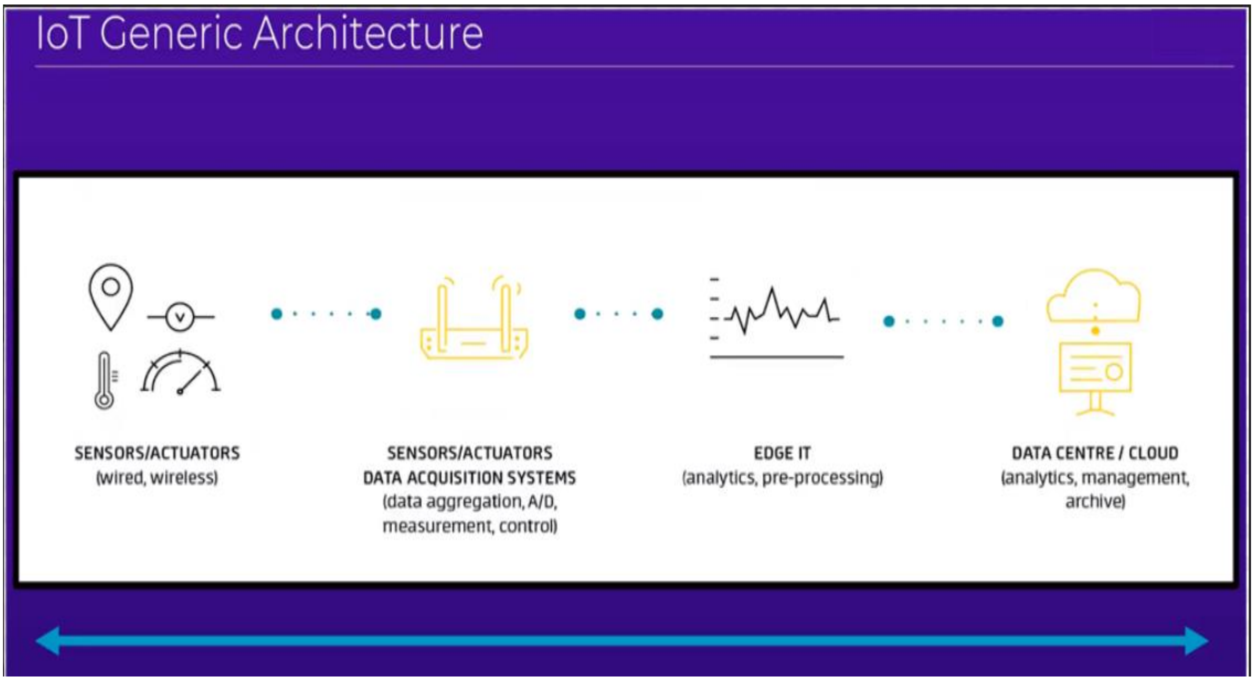


Figure 3-1 Generic IoT Architecture.

In this generic IoT architecture, the sensors/actuators connect to the Data Acquisition System (DAS) using any of various access technologies (Wi-Fi, Zigbee, NB-IoT etc) discussed in Chapter-2. TCP/IP/MPLS/SDN/IPSEC VPN etc also discussed in Chapter-2 can be used for

backhaul connectivity to the data centre. For this study, the backhaul is automated so as to introduce the required scaling, flexibility and reliability in the core network for IoT deployment. The PoC mainly focuses in the automation of the backhaul (Core) transport network by investigating the best way to connect an IoT device to a data centre.
Figure 3-2 shows the network topology of the PoC.

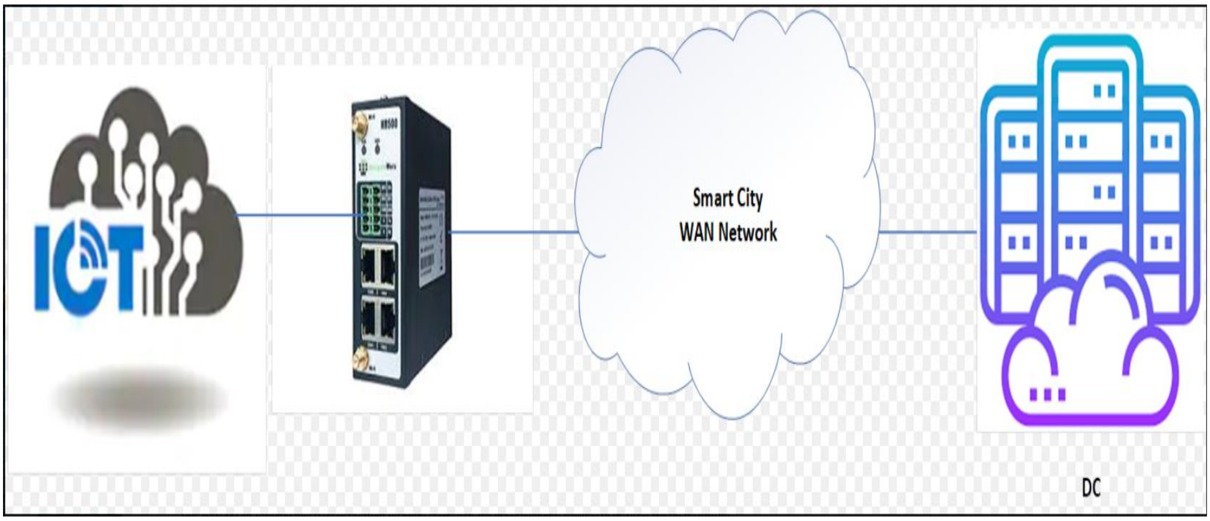


Figure 3-2 Network Topology of the PoC

Our task is to investigate the implementation of a transport network that can accommodate large scale and secured IoT sensor deployment. We propose that sensor integration/connection to the network should be wireless where possible as this will ease quick deployment by using the various wireless technologies discussed in section 2-4. The proposed evolutionary approach seeks to realize network simplicity, scalability and flexibility and vendor neutrality while addressing the challenges over existing transport networks which are rigidity, inflexibility, scalability, reliability and vendor lock-in. Section 3-1 discusses the Data Plane considerations followed by Section 3-2 which discusses the Control and Management Plane considerations. Finally, Section 3-3 proposes the evolutionary approach into implementing the S-IoT-N network automation. The proposed S-IoT-N architecture involves an automated network incorporating the concepts of SDN to build a MPLS like network. This is achieved through the implementation of IS-IS.

3.1 Data Plane Considerations

The proposed implementation strategy involves implementing automation on the existing transport infrastructures either being IP/MPLS/SDN infrastructures by putting intelligence at the edge of the network, a similar concept to Casado *et al* which is reproduced in Figure 3-3

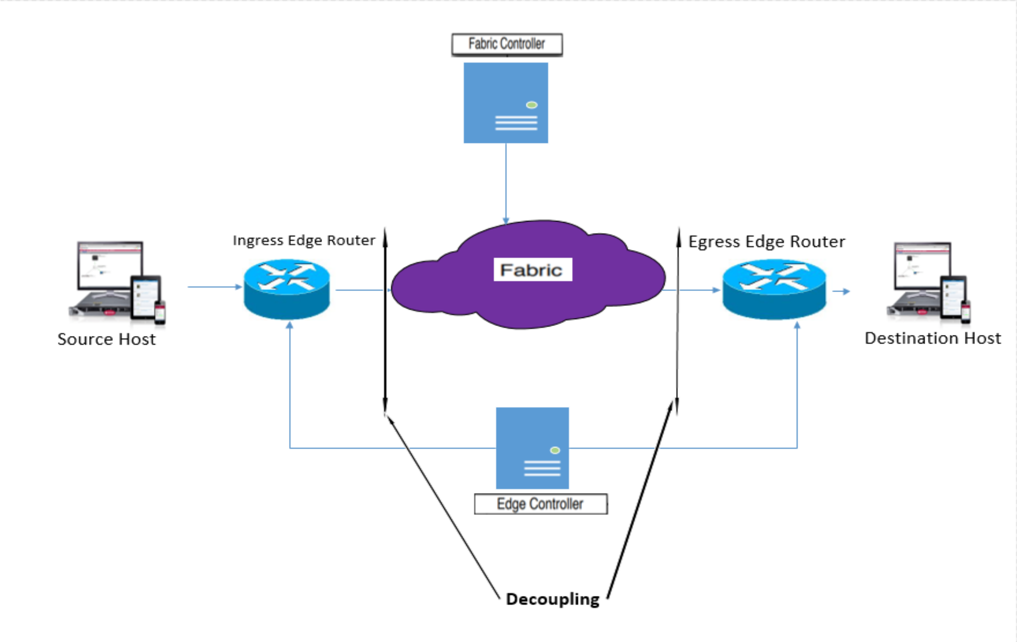


Figure 3-3: The proposed network fabric design by Casado *et al*.

According to Casado *et al*, implementing the network core as a fabric while keeping the complexity at the network edge facilitates flexible and independent evolution at both the network core and edge. The objective of their study was to apply the network fabric design insights to MPLS using the OpenFlow based SDN architecture. In this study we follow a similar approach, however, our proposed implementation strategy involves using IEEE 802.1ah [37-38] in the data plane.

We will recall from Chapter-2 that the Application layer in the TCP/IP stack (Figure 2-2) relies on the underneath layers (link, internet and transport layers). Any failure in underlying layer will result in the collapse of the Application layer services. This layer stacking is one of the disadvantages of the TCP/IP architecture. Moreover, the MPLS header (section 2-2) which is sandwiched between Laye-2 and Layer-3 in the TCP/IP architecture also makes MPLS vulnerable to this layer stacking. For the proposed implementation strategy, this layer stacking concept is replaced by the implementation of IEEE 802.1ah in the data plane which has a flatter architecture and this results in the simplification of the data plane that minimizes interdependency of higher layers.

The 802.1ah Provider Based Bridging (PBB) [37-38] encapsulates or decapsulates end-user traffic on a Backbone Edge Bridge (BEB) at the edge of the Provider Backbone Bridged Network (PBBN). PBB provides scalability to configure higher number of service instances in the network. PBB encapsulates customer's network into 802.1ah headers. These encapsulated packets are exchanged using unique and manually configured backbone address in the core network. This obviates the need for backbone core bridges to learn all MAC addresses of every host and thus achieving the scalability required for IoT applications. Figure 3-4 shows a simplified 802.1ah data plane which is for consideration in implementing the proposed evolutionary approach explained briefly in the following section.

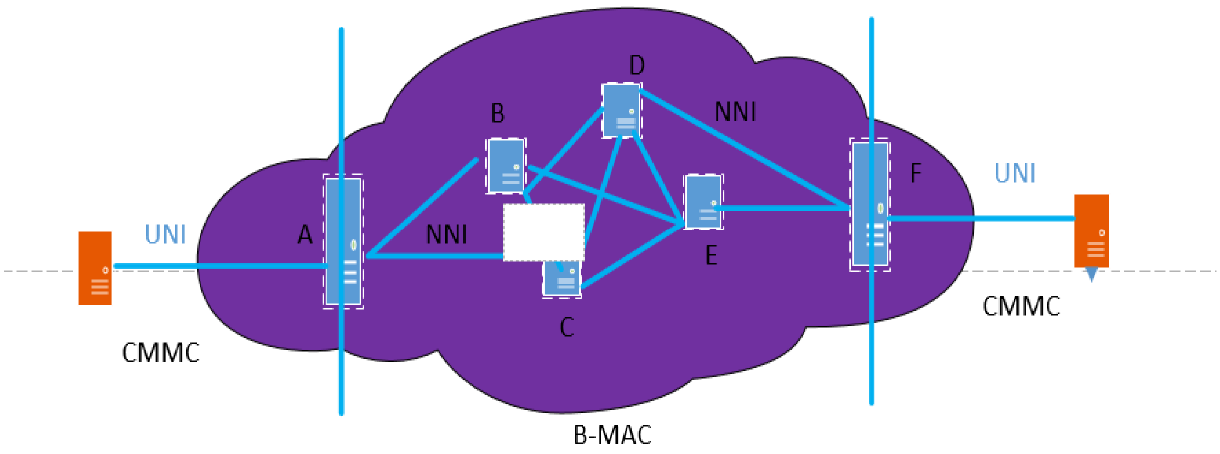


Figure-3-4 The Proposed Evolutionary Approach.

We will recall that at the Ethernet level (IEEE 802.11), all devices/hosts MAC addresses do not have priority/hierarchy. In 802.1ah, the concept of MAC address hierarchy is introduced between the subscriber/client edge and the network service core. The lower level hierarchy is where the client accesses the network core and this is called Client-MAC-Hierarchy (CMHC) which is the normal Ethernet operation. At the network core, the Back-Bone-MAC level (B-MAC) hierarchy is introduced which allows transportation of information across the core of the network and keeping the client edge in a totally separate plane.

Between the client and the core network the demarcation point is the User-to-Network Interface (UNI) and within the core network are Network-to-Network Interfaces (NNI). The NNIs operate at the B-MAC level while the UNIs operate at the C-MAC level (Figure 3-4) which determines the operation of the hierarchy. The hierarchy creates a tunnelling method called

MAC-in-MAC which provides encapsulation that provides the transport mechanism to move information across the cloud (core network).

3.2 Control and Management Plane Considerations

The control and management plane considerations involves the implementation of Intermediate System to Intermediate System (IS-IS) as a control plane. As stated in section 2-5, IS-IS is very extensible. Many available interior gateway routing protocols (OSPF, RIP, etc) are designed to mainly run IPv4 and depend on IP in order to run. IS-IS does not need IP in order to run it moves information by using Layer-2 Packet Data Units (L2-PDUs) which is traditional Ethernet framing.

For this study IS-IS is the proposed control plane of the data plane discussed in 3-1. This enables the separation of the control plane and the data plane (a similar concept employed in SDN discussed in section 2-3). Being the control plane, IS-IS programs the Link State Database (LSDB) that resides in every switch in the core network. The LSDB contains all the knowledge required to handle the data plane. For information to be transported across the network, IS-IS uses Type Length Values (discussed in section 2-6). Recapping from section 2-6, TLVs contain three fields, the Type field which identifies the type of information carried by the PDU for example routing information or L2/L3-services. The Length field which indicates the length of the PDU and then the Value field which indicates the value of the information.

TLVs are also used to move network topology information which provides each core switch with the network map and thereby informing each switch what to do in case of network failure which results in network resiliency and reliability for critical services. TLVs also represent services (L2/L3 IP-VPNs, Multicast, IPv4/IPv6 etc) to be transported. This makes the network suitable to carry (traditional) network traffic and also make it suitable for new IoT applications. Figure 3-5 shows the network stack of the proposed “S-IoT-N” network architecture.

Application Plane (Services)	L2/L3/IP-VPN/MC/IPv6
Control Plane	IS-IS
Data Plane	802.1ah

Figure 3-5 Network stack of the Proposed Network Architecture

3.3 Proposed "S-IoT-N" Architecture

The proposed architecture for the "Scalable Transport Network for IoT Deployment" involves implementing the data and control plane considerations to provision the underlying network resources. This enables a scalable, reliable and flexible network solution to be deployed on the existing TCP/IP/MPLS or SDN infrastructures. Furthermore, IS-IS together with LLDP handles the control layer functions and being open and matured networking standards, this overcomes the vendor lock-in challenge experienced in traditional networks. Figure 3-6 shows the proposed S-IoT-N architecture.

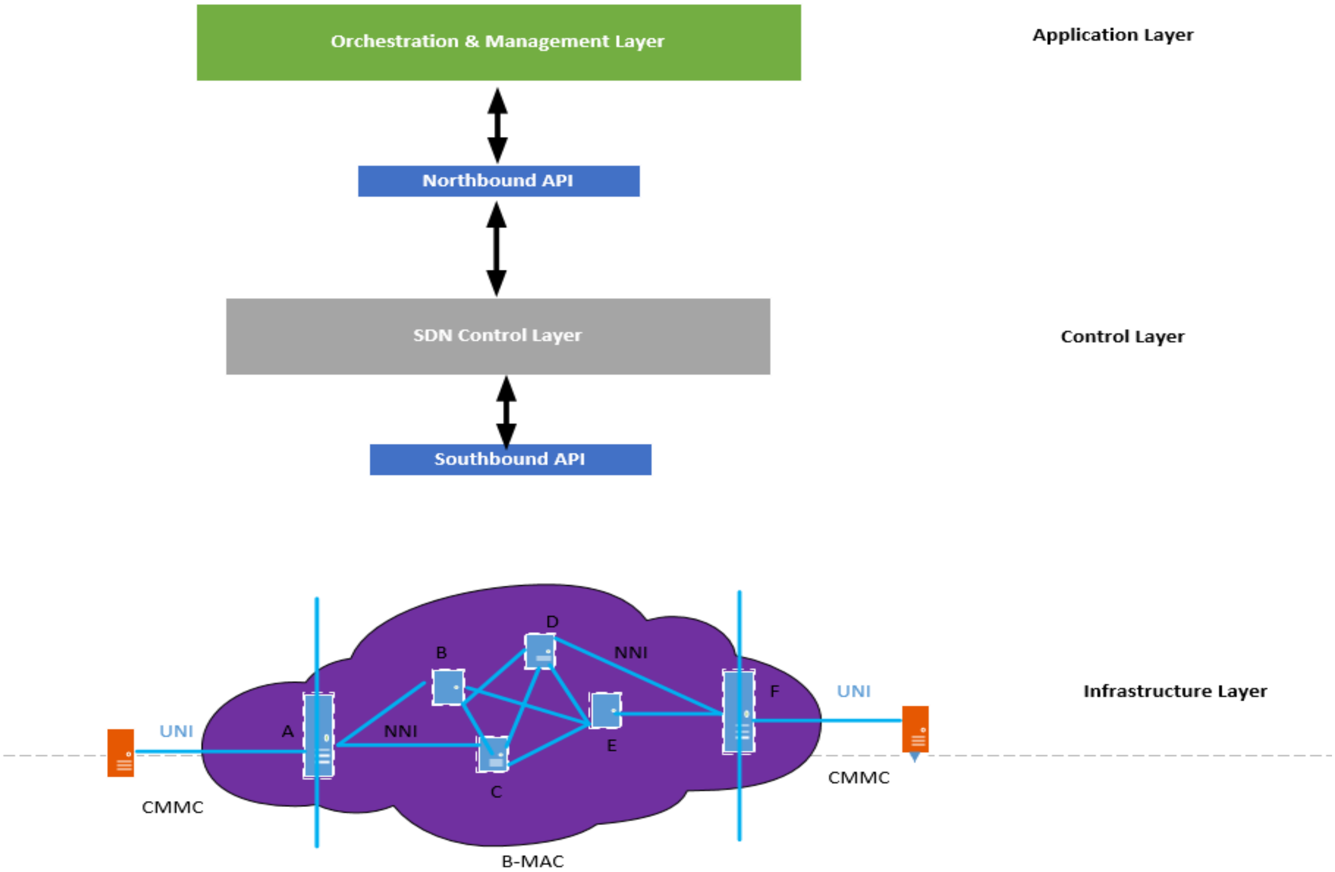


Figure 3-6 Proposed S-IoT-N Architecture

As seen in Figure 3-6, the data plane involves a simple layer-2 network with no IP involved which results in a simplified, reliable, flatter and a scalable network. Accordingly, the control plane is de-coupled from the data plane (SDN concept). The proposed strategy addresses and solves the challenges over the (traditional) network technologies which are TCP/IP and MPLS based on the following:

- Firstly, the proposed S-IoT-N architecture involves a quick and simpler deployment of IoT applications resulting in a scalable network.

- The architecture also involves a simpler and flatter network which is easy to configure, simpler and cheaper to manage.
- Automation brought about by pairing IS-IS and LLDP features results in faster service deployment and minimal network engineer skill required to maintain the S-IoT-N.

Before we conclude this chapter, it is important to discuss the importance and use of the Individual Service Identifier (I-SID) proposed in the S-IoT-N. Figure 3-7 shows the IEEE 802.1ah frame.

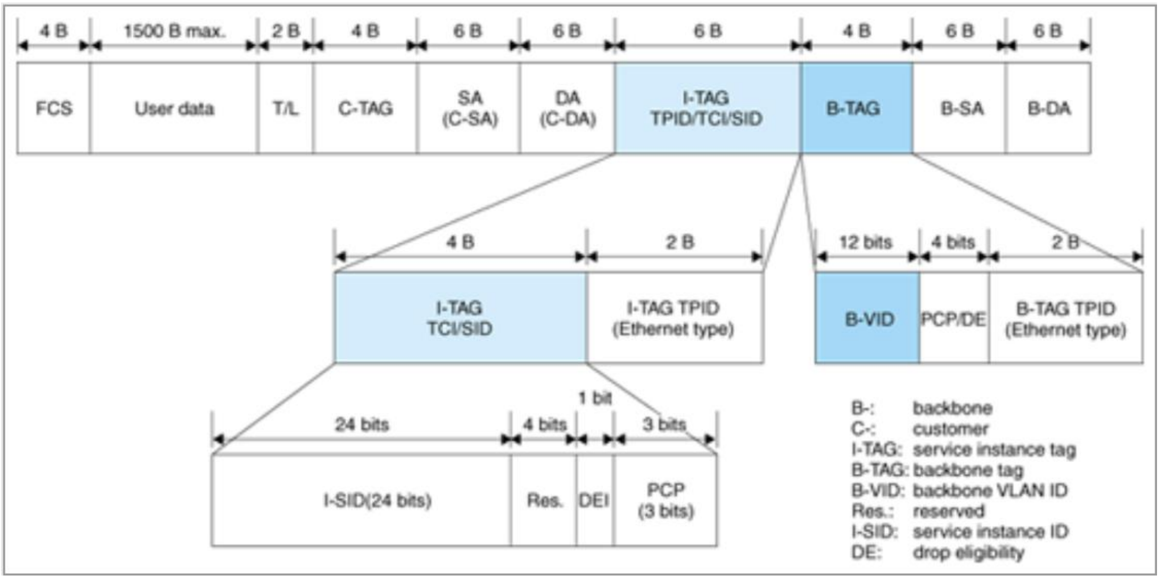


Figure 3-7 IEEE 802.1ah frame format [37]

The 802.1ah frame is the modified (advanced) version of the traditional 802.1 (Ethernet frame).

Most importantly is the additional fields notably, the I-TAG field which is further expanded to contain the I-SID filed (24-bits), the RES, DEI and PCP. In the S-IoT-N implementation, the I-ISD field is utilized to address the IoT devices resulting in in 2^{24} = 16,777,215 flat space addresses in a single domain network. In other words, 1-port of a switch can service this amount of addresses. If a switch has 48-ports this will mean one switch can service 48 x 16,777,215 = 805,306,320 IoT services/devices. It is clear how scalable is the proposed S-IoT-N can be. As long as the switches/routers have the required ASIC processing power it is clearly evident that this network is hugely scalable. This attests the importance of service automation since human intervention will great hinder fast service deployment. Moreover, without automation, troubleshooting a failed service by human intervention can take enormous amount of time considering the scale of the network.

Thus, the above stated strategy and approach for the proposed S-IoT-N architecture addresses and solves the challenges of scalability, interoperability, reliability, flexibility and vendor lock-in associated with traditional networks which is the key in meeting the research objectives of this study.

Lastly, the proposed architecture involves an incremental deployment of a network that can co-existing with the existing network technologies. The approach also ensures a non-disruption of live services and an incremental migration of existing services to the new infrastructure.

In conclusion, the proposed evolutionary approach realizes the major benefits of scalability, and simplicity. Furthermore, the S-IoT-N uses open standards which allow for vendor neutrality and flexibility which unlocks vendor lock-in while successfully addressing the major challenges over TCP/IP/MPLS technologies.

4. Proof of Concept Implementation

In this chapter, we explore the proof of concept (PoC) of the proposed S-IoT-N. The PoC is demonstrated and built on a physical test bed. Accordingly, at first, Section 4-1 details the PoC design and is followed by Section 4-2 detailing the PoC implementation on a physical network test bed. Finally, the chapter concludes by looking at the performance and validation of the PoC (section 4-3)

4.1 Proof of Concept Design Architecture

For simplicity and to illustrate a logical implementation of the PoC, the S-IoT-N Architecture in Figure 3-6 is slightly modified and simplified for the purposes of demonstration. The routers in the cloud (B &C) are the core switches/routers are shown as in in Figure -4-1. This is so because our LAB has 2-core switches available because of their cost. It is shown in the PoC that the configuration and Link State Data Base (LSDB's) is automated in the same way for all core network components regardless of the number of core network devices. Therefore, the 2-core-switches represented here are adequate for the purposes of demonstrating the PoC.

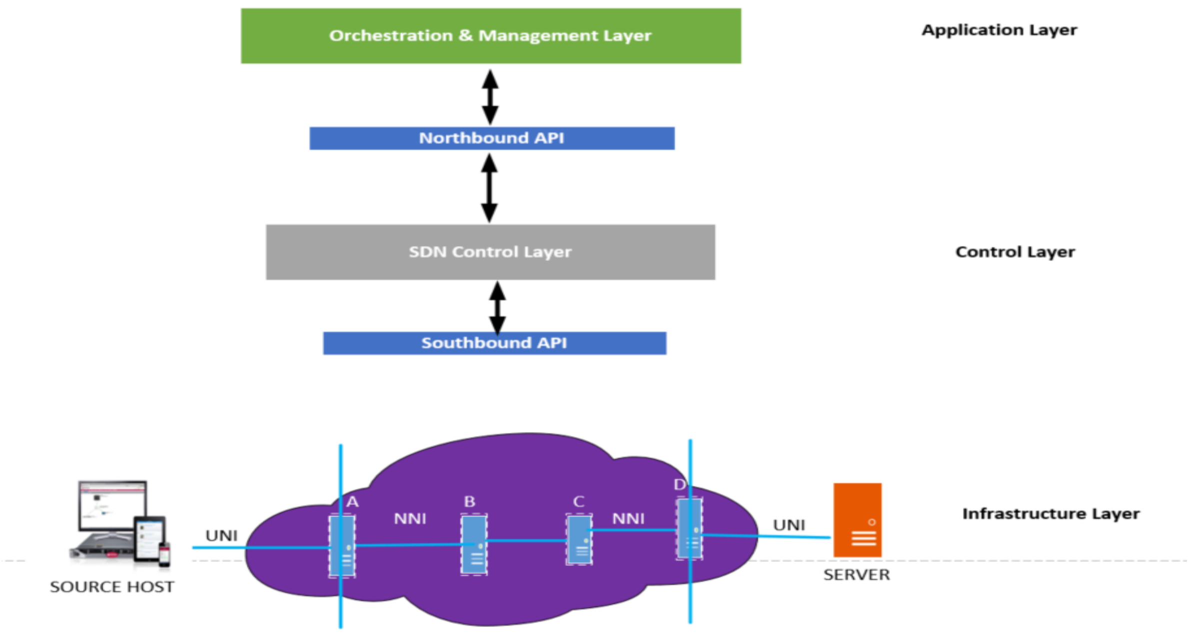


Figure 4-1 The proposed S-IoT-N Architecture

In this chapter, the PoC design components are presented and discussed, further configuration information can be obtained by referring to Appendix A & B. Firstly, as per the data plane considerations, the PoC design is tested and implemented on a physical testbed (section 4-2). In order to achieve the objective of this study, basic network connectivity functions and services

(IP-camera traffic) is implemented as an application in the S-IoT-N platform. Secondly, configurations are disseminated via the Southbound API to the data plane (Infrastructure layer). Finally, the PoC design also involves some low level configuration of the access/edge and the core network, and for this, vendor specific Command Line Interface (CLI) is used.

4.2 Implementation on a Physical Network Testbed

We begin this section by setting up a physical test bed which consists of 5 Extreme X460-G2-48t-10GE4s (3x used as MPLS Cores and 2x used as Edge devices) and 2x AVAYA VSP-8404s (used as S-IoT-N Core devices referred to as Backbone Edge Bridges (BEB-1 & 2)). Figure 3-2 is re-introduced in this section as Figure 4-2.

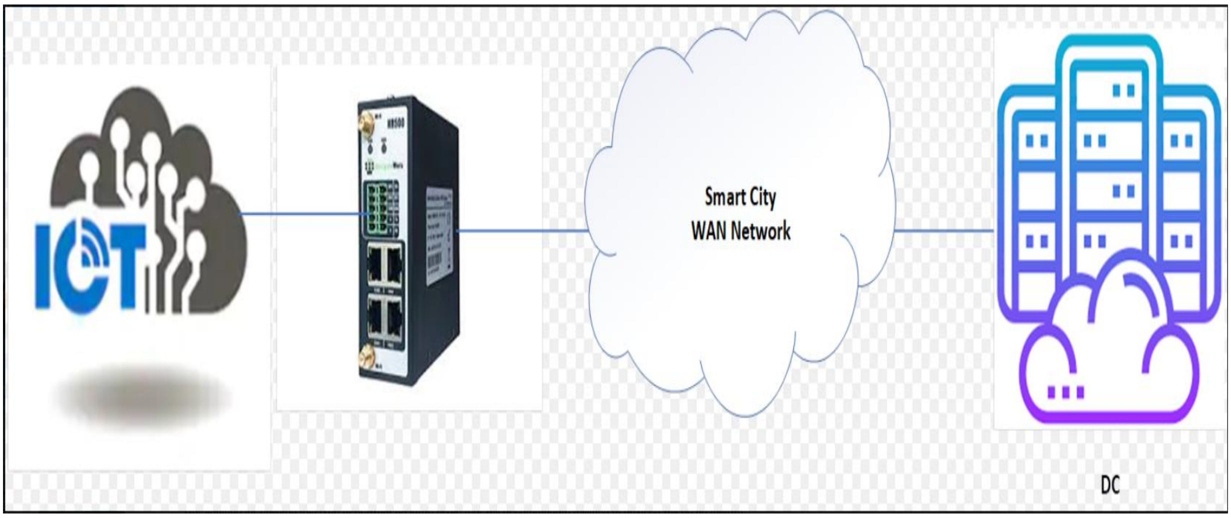


Figure 4-2 Network Topology of the PoC

Accordingly, the PoC logical network representing Figure 4-2 is shown in Figure 4-3 together with the corresponding physical testbed shown in Figure 4-4

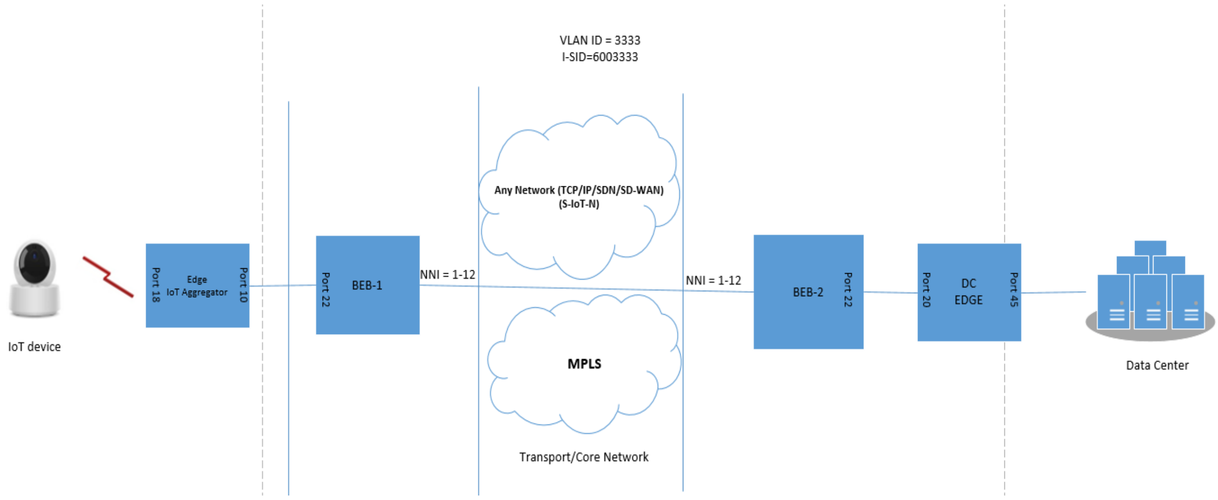


Figure 4-3 Logical network diagram of the PoC.

The IoT device is represented by an IP-camera as seen on top of the cabinet (left picture) in Figure 4-4. The “Edge Aggregator” connects the IP-device (camera). Here, any of the access network technologies discussed in section 2-4 can be used to connect the IoT device to the edge aggregator. Essentially, these access networks convert (L1/L2) messaging/signalling between the IoT device and the edge switch to L3/L2 which can then be transported by the transport networks (MPLS/IS-IS/SDN/ S-IoT-N).

The “DC Edge” connects data centre servers. The edge aggregator and the DC edge devices reside on the edges of the network and demarcate the access, transport and core network layers.

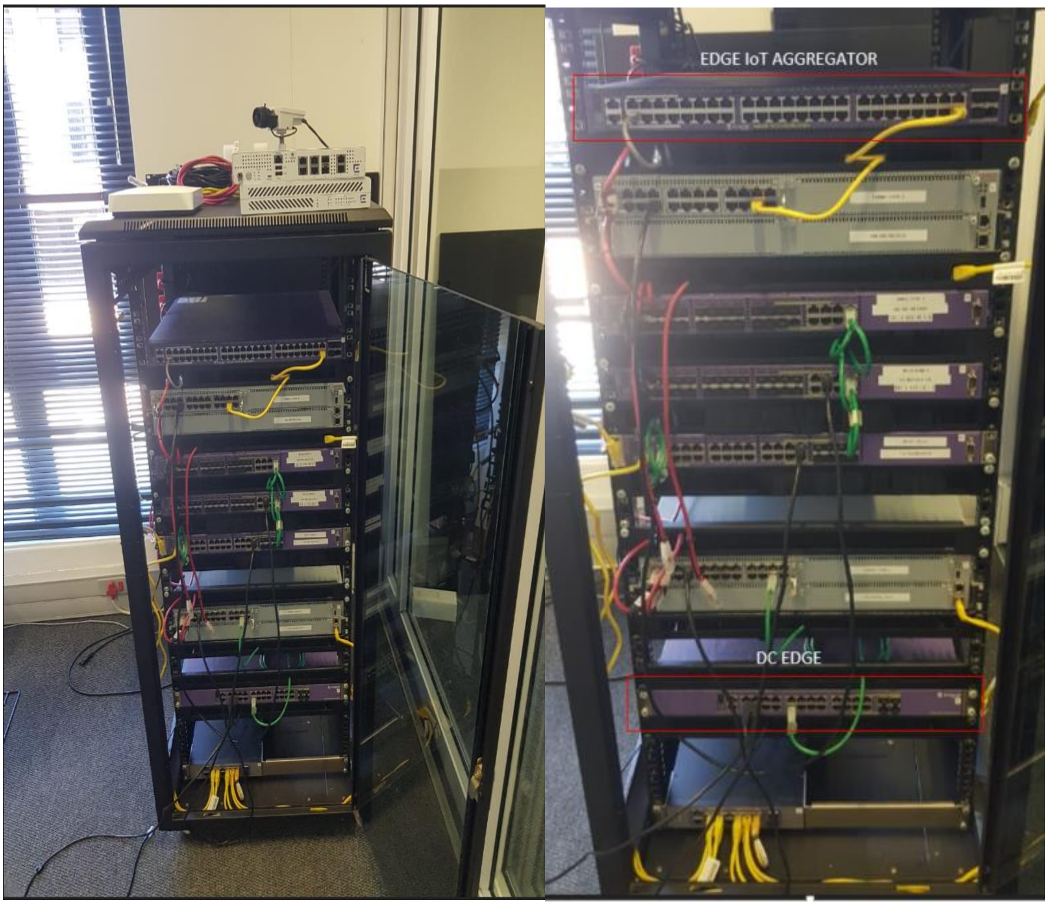


Figure 4-4 Physical Network testbed.

Figure 4-3 also shows the IP/MPLS cloud representing the MPLS service provider or MPLS core network linking the IP-edge and the datacentre. This is represented by the 3 x middle switches (left picture) as shown in Figure 4-5. The “Any network” cloud can be any network (SDN, TCP/IP, SD-WAN/S-IoT-N etc).

The concept here is to demonstrate that only network edges that needs minimal configurations which results in easy scaling, flexibility and network resiliency.

Figure 4-5 Testbed showing physical network components.

Figures 4-3 & 4-5 respectively represent (logically and physically) the 2 types of the transport networks being investigated in this study which are MPLS (represented by the 3-switches in figure 4-5) and S-IoT-N represented by BEB-1 and BEB-2. Section 4-2-1 looks at adding a simple IoT device in the traditional IP/MPLS network. Here, the study aims to demonstrate the complexity of configuring the MPLS network so as to activate a simple IoT service in traditional networks. Moreover, the aim is also to demonstrate to the reader that configuring thousands of devices in this way is very complex, cumbersome and very difficult to scale. In section 4-3-1, certain links in the core network are disconnected with the objective of testing network resiliency (convergence) and service availability. In IP/MPLS, it is demonstrated that services do fail because of the time the MPLS network takes to converge after a link failure due to link re-calculations and the updating of routing tables which is performed by the interior gateway protocol e.g. OSPF or RIP running in the IP/MPLS domain.

Section 4-2-2 looks at the implementation of the S-IoT-N which is further tested for its performance in section 4-3-2. It is observed that because of the automatic creation and updating of LSDB's by IS-IS, switches (BEB-1 &2 in this case) simple forward the traffic to other routes since all switches are aware of each switch's LSDB (they all have the same network topology). This results in no services failure thereby introducing network resilience and reliability.

4.2.1 PoC Implementation in IP/MPLS

Figure 4-6 shows a logical view with a simplified IoT service to be implemented in MPLS. To link the IoT device, any of the access networks discussed in section 2-4 can be used. The customer premises equipment (CPE) resides at the edge of the MPLS network and provides L3/IP service at the entry point of the MPLS core/cloud.

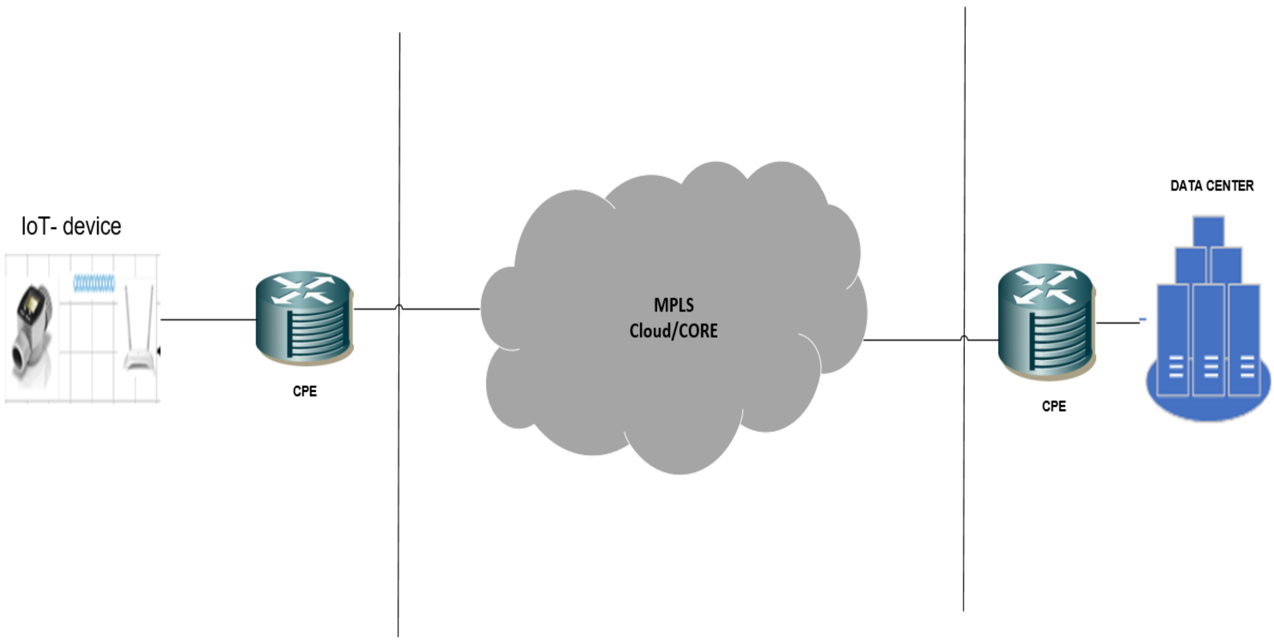


Figure 4-6 Logical view of the MPLS Testbed

Figure 4-7 is the logical representation of the PoC.

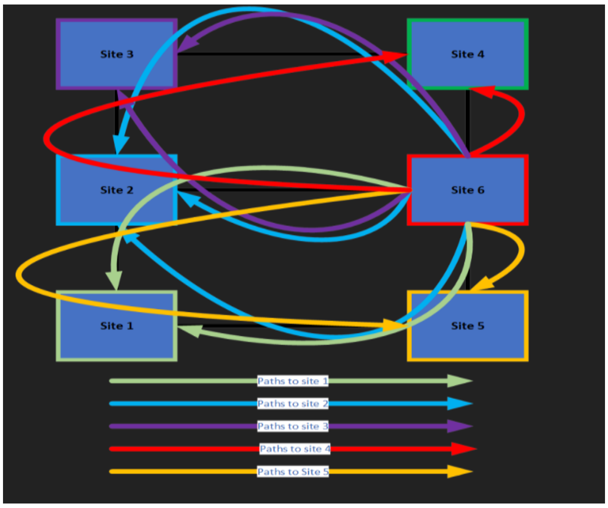


Figure 4-7 Logical representation of PoC MPLS core network.

For the MPLS cloud to work and have full redundancy, a mesh network must be created. The network engineer needs to configure every single core router in the cloud and manually create paths (routes) to each core device/site (L2-VPLS). Appendix-A shows the configuration that was done to create a Layer-2 Virtual Private Lan Service (L2-VPLS) which is essentially a MPLS mesh network represented by the different colour lines leaving Node-6 in Figure 4-7. As it can be noticed by the reader, because of its complexity and many lines of configuration required to achieve this objective, this configuration could be included in this section as it would make this discussion unnecessarily long and the reader would lose the message being conveyed. Moreover, every service that needs to be added, configuration must be manually done to each and every MPLS node. Appendix-A is only showing configuration of one node (node-6) as depicted by the lines connecting this site to other sites. It is clear that this is complicated, cumbersome, prone to error and not scalable and also, there must be planned network downtime that will allow for reconfigurations since the network will have to re-converge when implementing the new changes/additions. This is not a way to go for IoT deployment.

4.2.2 PoC Implementation S-IoT-N

Having demonstrated the complexities of implementing IoT services in IP/MPLS (section 4-2-1). We demonstrate the implementation of S-IoT-N in this section. We start by configuring the “DC Edge” device so as to establish a service to the server. The network connections together with the relevant ports are shown in Figure 4-8. This section is meant to demonstrate how the PoC was developed and tested, detailed LSDBs of the core switches of this PoC is documented in Appendix-B.

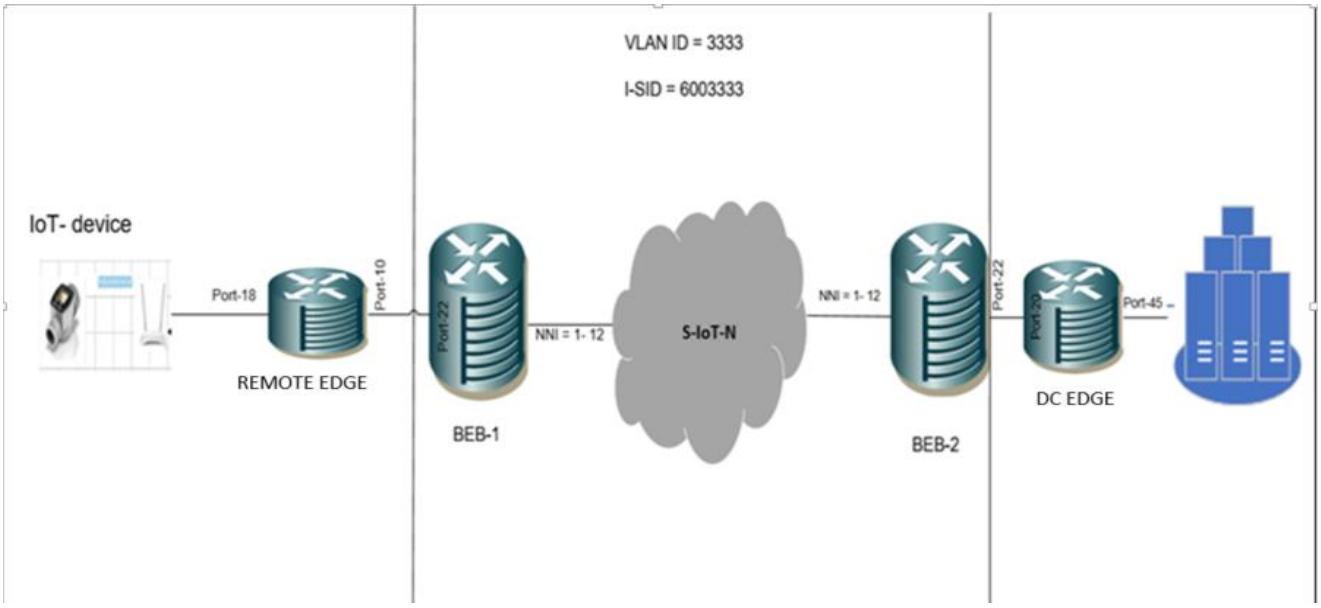


Figure 4-8 Logical view of the S-IoT-N testbed

When the “BEB-2” device is plugged into port-20 of the “DC EDGE” device the following automatically happens:

- BEB-2 informs DC EDGE that it is its neighbouring router/switch
- BEB-2 also advertises its MAC address
- Informs the neighbouring device which port is it connected to.
- Advertises TTL of the broadcast and informs how long will it wait for the EDGE device to respond.
- It tells it its name and its equipment type and
- As well as its role in this network configuration.

IS-IS takes responsibility of this automated function in conjunction with LLDP, this is shown in Figure-9.

```
* DC-EDGE.7 # show lldp neighbors detailed

-----------------------------------------------------------------------------
LLDP Port 20 detected 1 neighbor
  Neighbor: 64:6A:52:A5:7C:00/1/22, age 2 seconds
    - Chassis ID type: MAC address (4)
      Chassis ID      : 64:6A:52:A5:7C:00
    - Port ID type: ifName (5)
      Port ID      : "1/22"
    - Time To Live: 120 seconds
    - System Name: "FABRIC_CORE_2"
    - System Description: "VSP-8404 (7.1.0.0)"
    - System Capabilities : "Bridge, Router"
      Enabled Capabilities: "Bridge, Router"
    - Port Description: "Extreme Networks Virtual Services Platform 8404 Mod\
                         ule 8424XT - 10GBaseT Port 1/22"
    - Avaya/Extreme Fabric Attach element
      Element Type    : 4
      State           : 8
      Management Vlan: 0
      SystemId        : 64:6a:52:a5:7c:00
      Link Info       : 00-00-00-d5
```

Figure 4-9 DC EDGE Link State Database Table.

The DC EDGE device also connects the server (a virtual machine on a PC was created for this) which runs the IP-camera application. For the service to come up, the following needs to be configured on the edge device:

- IoT/Server VLAN.
- Assigning of a port to IoT/Server VLAN.
- Assigning of a I-SID to IoT/Server VLAN.

We use VLAN = 3333 and I-SID= 6003333 as depicted in Figure 4-8. This minimal configuration to add the above parameters is depicted in Figure 4-10.

```
* DC-EDGE.9 # create vlan 3333
* DC-EDGE.10 # configure vlan 3333 add ports 45 untagged
VLAN 3333 VLAN_3333:  Port 45 untagged has been auto-moved from VLAN "VLAN_2123" to "VLAN_3333".

* DC-EDGE.11 # configure vlan 3333 add isid 6003333
```

Figure 4-10 Configuration of the “DC EDGE” device.

It is important to observe and compare this configuration with MPLS as demonstrated in section 4-2-1. Essentially, only 2-lines were added on the device compared to the many lines of configuration added in the MPLS device. The reason for this is the automation brought by IS-IS working today with LLDP. Figure 4-11 verifies that the configuration has been applied on the edge device.

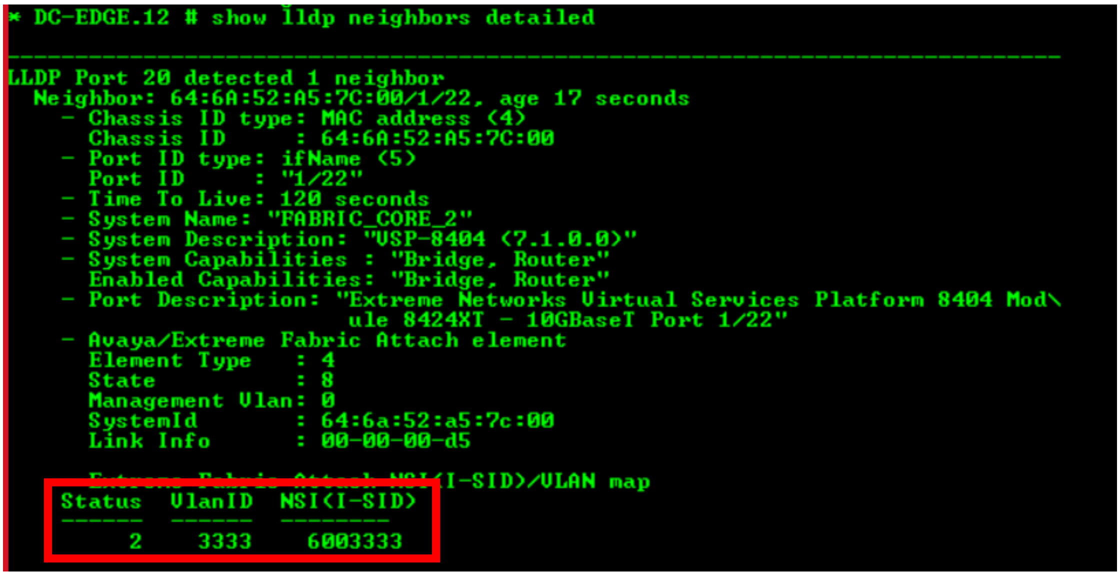


Figure 4-11 Verification of configuration applied to the "DC Edge" device

The requirement for the S-IoT-N is that it must be scalable, automated and reliable. We need to observe if this is the case. The next step is to verify if the chosen control plane (IS-IS) is able to achieve this (i.e. automatically create LSDBs and detect services). This will mean the goal of automation resulting in scaling up IoT deployment is achieved. Network resiliency is dealt with in section 4-3 where the performance of this network is tested. We move to the core network device BEB-1 (host name: FABRIC_CORE_2) and we verify the data plane has no live/working services so as to ensure that the work we are implementing is correctly reflected. As shown in figure 4-12, the only I-SID that exists is the default control plane I-SID.

Figure 4-12 Verification of BEB-1 Link State Database showing the data plane.

Most importantly, as shown in Figure 4-13, the core devices (BEB-1 & BEB-2) were able to detect each other due to the control plane (IS-IS) being able to automatically create LSDBs on the core devices.

```
FABRIC_CORE_2:1#show isis adjacencies

================================================================================================
                                ISIS Adjacencies
================================================================================================
INTERFACE        L STATE         UPTIME PRI HOLDTIME SYSID              HOST-NAME        STATUS
------------------------------------------------------------------------------------------------
Port1/1          1 UP          01:07:40 127       24 00db.face.0003     FABRIC_CORE_3    ACTIVE
Port1/2          1 UP          01:07:34 127       24 00db.face.0003     FABRIC_CORE_3    BACKUP

--------------------------------------------------------------------------------
 2 out of 2 interfaces have formed an adjacency
--------------------------------------------------------------------------------
```

```
FABRIC_CORE_2:1#show isis lsdb

================================================================================================
                                  ISIS LSDB
================================================================================================
LSP ID                        LEVEL    LIFETIME  SEQNUM     CHKSUM     HOST-NAME
------------------------------------------------------------------------------------------------
00db.face.0002.00-00          1        416       0xf        0xff18     FABRIC_CORE_2
00db.face.0002.00-01          1        1077      0x10       0x1c40     FABRIC_CORE_2
00db.face.0002.00-02          1        416       0xb        0x4844     FABRIC_CORE_2
00db.face.0003.00-00          1        815       0x16       0xda97     FABRIC_CORE_3
00db.face.0003.00-01          1        815       0x14       0x814e     FABRIC_CORE_3
00db.face.0003.00-02          1        815       0x12       0x3e45     FABRIC_CORE_3
```

Figure 4-13 BEB-1 Link State Database showing the control plane.

As the commands are executed from BEB-1 (FABRIC_CORE-2), it is observed that this device can see BEB-2 (FABRIC_CORE_3) as its adjacent neighbour and in its LSDB, it sees this neighbour as IS-IS Level-1 neighbour (as discussed in section 2-5). Moreover, what we also observe is that as soon as the VLAN and I-ISD configuration is implemented on the edge device, the I-SID is automatically learned via LLDP from the edge device and placed into the data plane of BEB-1 (Figure 4-14)

```
FABRIC_CORE_2:1#show i-sid

==========================================================================================
                                    Isid Info
==========================================================================================
ISID         ISID                         PORT               MLT                ORIGIN
ID           TYPE         VLANID          INTERFACES         INTERFACES
------------------------------------------------------------------------------------------
6003333      ELAN         N/A             c3333:1/22         -                  DISC_LOCAL
16777001     ELAN         N/A             -                  -                  CONFIG
```

```
TLV:144  SUB-TLV  3        ISID:
                  Instance: 0
                  Metric: 0
                  B-MAC: 02-2d-b2-ff-ff-ff
                  BVID:4051
                  Number of  ISID's:1
                           16777215<None>

                  Instance: 0
                  Metric: 0
                  B-MAC: 02-2d-b2-ff-ff-ff
                  BVID:4052
                  Number of  ISID's:1
                           16777215<None>

                  Instance: 0
                  Metric: 0
                  B-MAC: 00-db-fa-ce-00-02
                  BVID:4051
                  Number of  ISID's:1
                           6003333<Both>

                  Instance: 0
                  Metric: 0
                  B-MAC: 00-db-fa-ce-00-02
                  BVID:4052
                  Number of  ISID's:1
                           6003333<Rx>
```

Figure 4-14 BEB-1 Link State Database showing the data plane.

If we want to see network elements/nodes belonging to the service with I-SID 600333, we observe in Figure-4-15 that there is a device which has advertised its MAC address, and we see that this MAC address has been learned automatically via port-22 of this device and what is connected in this port as per the network topology (Figure4-8) is the DC EGDE device.

```
FABRIC_CORE_2:1#show i-sid mac-address-entry 6003333

==========================================================================================
                                  I-SID Fdb Table
==========================================================================================
I-SID     STATUS  MAC-ADDRESS        INTERFACE      TYPE      DEST-MAC           BVLAN DEST-SYSNAME
------------------------------------------------------------------------------------------
6003333   learned 00:04:96:99:21:7e  c3333:1/22     LOCAL     00:00:00:00:00:00  0
```

Figure 4-15 BEB-1 Link State Database showing the data plane.

Let us investigate events that have occurred on BEB-2 (FABRIC_CORE_3) as we connect and configure the edges of the network. Initially, we can see that there is no service in the forwarding plane except for the default control service (Figure 4-16).

```
FABRIC_CORE_3:1(config)#show i-sid
==========================================================================================
                                        Isid Info
==========================================================================================
ISID        ISID                      PORT               MLT                ORIGIN
ID          TYPE       VLANID         INTERFACES         INTERFACES
------------------------------------------------------------------------------------------
16777001    ELAN       N/A            -                  -                  CONFIG
```

Figure 4-16 BEB-2 Link State Database showing the data plane.

After adding a service on the DC EDGE device, it was automatically detected by BEB-1 which in turn advertised the service to other BEBs in the network (in this case BEB-2) We can then observe that in the control plane of BEB-2 has automatically learned about the new service that is available on BEB-1 (FABRIC_Core_2) but the forwarding plane is not yet extended to the BEBs this is shown in Figure 4-17.

```
Level-1 LspID: 00db.face.0002.00-01     SeqNum: 0x00000012      Lifetime:    574
        Chksum: 0x9c50  PDU Length: 99
        Host_name: FABRIC_CORE_2                              Source
        Attributes:     IS-Type 1


TLV:144 SUB-TLV 3       ISID:
                Instance: 0
                Metric: 0
                B-MAC: 02-2d-b2-ff-ff-ff
                BVID:4051
                Number of ISID's:1
                        16777215(None)

                Instance: 0
                Metric: 0
                B-MAC: 02-2d-b2-ff-ff-ff
                BVID:4052
                Number of ISID's:1

                        16777215(None)

                Instance: 0
                Metric: 0
                B-MAC: 00-db-fa-ce-00-02
                BVID:4051
                Number of ISID's:1
                        6003333(Both)

                Instance: 0
                Metric: 0
                B-MAC: 00-db-fa-ce-00-02
                BVID:4052
                Number of ISID's:1
                        6003333(Rx)
```

Figure 4-17 BEB-2 Link State Database showing the data plane.

We now move to the REMOTE EDGE (IOT_EGDE) device and observe that there are no services yet as shown in Figure- 4-18. It has however detected the MAC address of BEB-2 device the moment it was plugged in in port-10 and automatically updated its LSDB following the sequence as discussed for the DC EDGE device.

```
* IOT_EDGE.10 # show lldp neighbors detailed

-----------------------------------------------------------------------------
LLDP Port 10 detected 1 neighbor
  Neighbor: 64:6A:52:A6:08:00/1/22, age 4 seconds
    - Chassis ID type: MAC address (4)
      Chassis ID      : 64:6A:52:A6:08:00
    - Port ID type: ifName (5)
      Port ID     : "1/22"
    - Time To Live: 120 seconds
    - System Name: "FABRIC_CORE_3"
    - System Description: "VSP-8404 (7.1.0.0)"
    - System Capabilities : "Bridge, Router"
      Enabled Capabilities: "Bridge, Router"
    - Port Description: "Extreme Networks Virtual Services Platform 8404 Mod\
                         ule 8424XT - 10GBaseT Port 1/22"
    - Avaya/Extreme Fabric Attach element
      Element Type    : 4
      State           : 8
      Management Vlan: 0
      SystemId        : 64:6a:52:a6:08:00
      Link Info       : 00-00-00-d5

* IOT_EDGE.11 #
```

Figure 4-18 IoT-Edge Link State Database showing the data plane.

This edge device will connect a camera. For the service to come up, the following needs to be configured on it as well:

- IoT-device VLAN
- Assigning of a port to IoT-device Vlan
- Assigning of a I-SID to IoT-device Vlan (this is shown in Figure 4-19).

The same VLAN = 3333 and I-SID= 6003333 as depicted in the logical network diagram will be used.

```
* IOT_EDGE.11 # create vlan 3333
* IOT_EDGE.12 # configure vlan 3333 add ports 18 untagged
* IOT_EDGE.13 # configure i-s
                             ^

%% Invalid input detected at '^' marker.
* IOT_EDGE.13 # configure vlan 3333 add isid 6003333
* IOT_EDGE.14 #
```

Figure 4-19 Configuration of the “IoT Edge” device

As soon as the service is configured, LLDP immediately advertise the service to the network. This is shown in Figure 4-20.

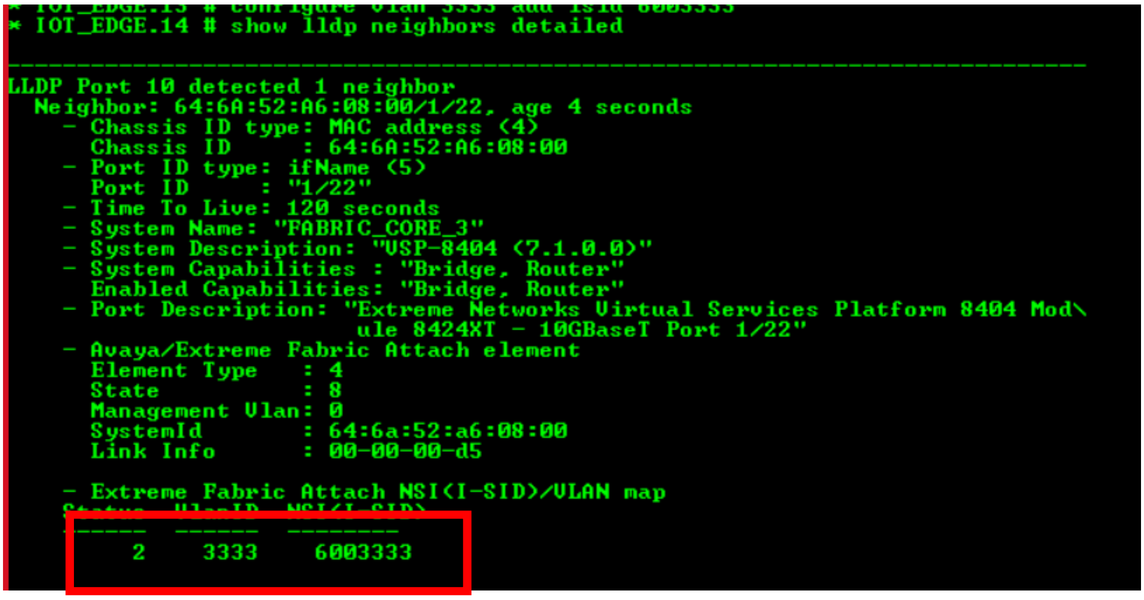


Figure 4-20 “IoT Edge” LSDB

Most importantly, as soon VLAN and the I-SID were created on the edge device, the service immediately came up as shown in Figure 4-21

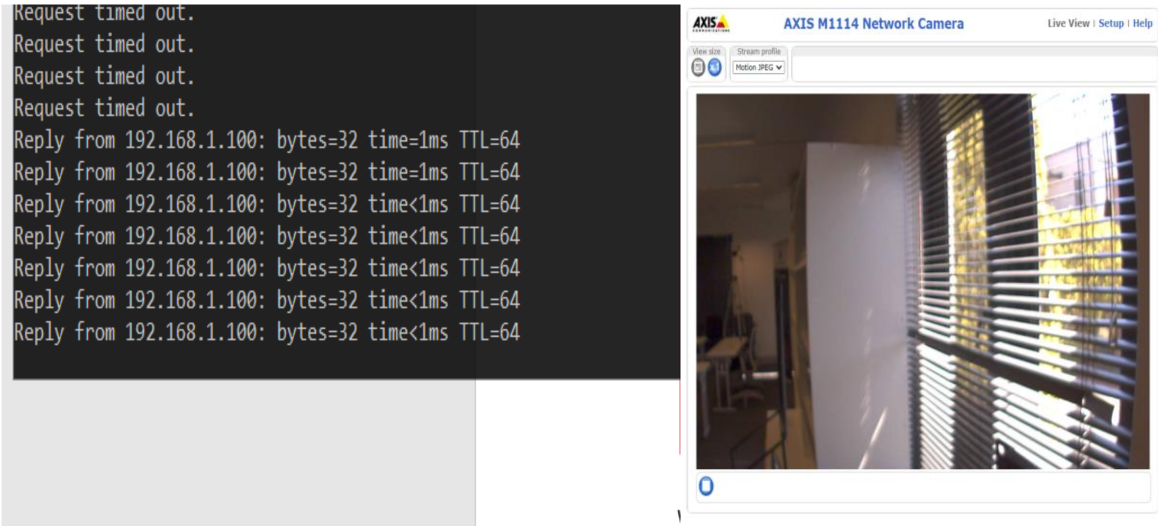


Figure 4-21 IP-camera service.

What is crucial to notice in this PoC is that the network core was untouched and only the E=edge devices were configured by just adding service parameters (only VLAN & I-SID). The control plane automatically created LSDBs and programmed the data plane. Only the network edges were configured and with minimal configuration. Furthermore, any IoT device that will work on this service (VLAN 3333 and I-SID 6003333) can simply be plugged in on the edge device and will automatically work with no extra configuration. The service is only created once and will accommodate any IoT device for that service. This results in quick and scalable deployment. Figure 4-22 shows the LSDB of BEB-2 and we can notice that the I-SID created on the edge devices was automatically detected on port-22 via LLDP.

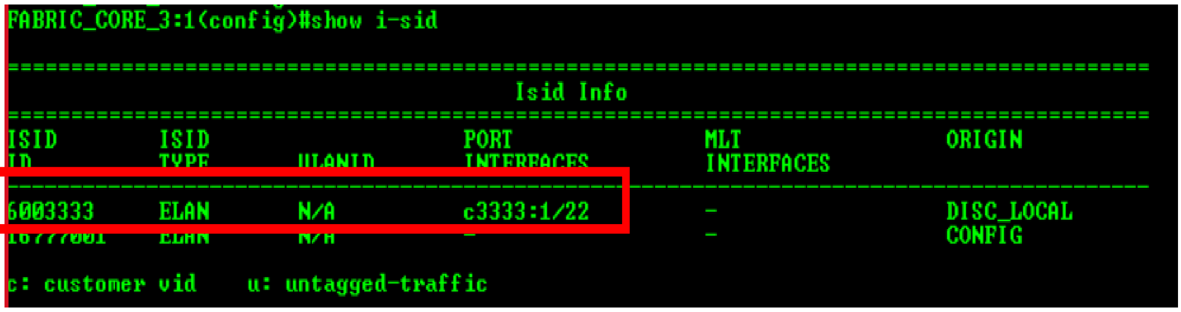


Figure 4-22 BEB-2 LSDB showing automatic detection if I-SID.

The control plane also automatically programmed the forwarding plane of the core devices so that they are able to transport the services as shown in Figure-4-23

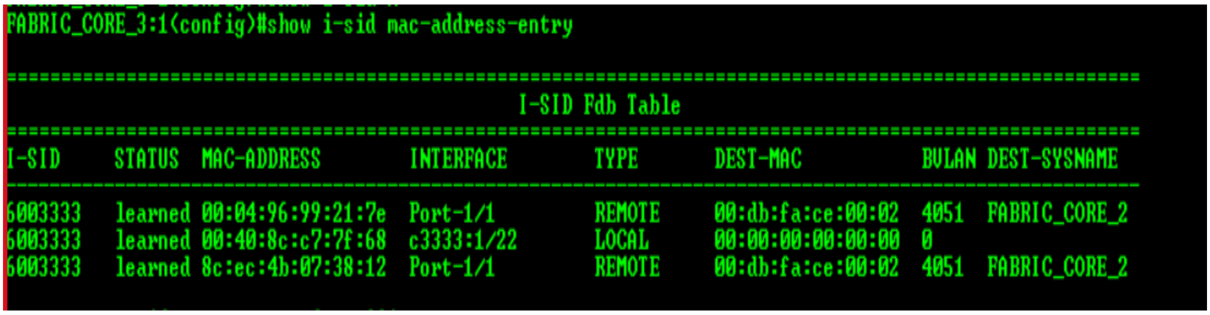


Figure 4-23 BEB-2 forwarding plane.

We can also see from Figure 4-24 that BEB-2 is aware of all core devices and how to get there (i.e. the full network topology). This will enable automatic switching over of traffic to a different route should there be a link failure. If a link that is connected to any of the 6-ports that connect to FABRIC_CORE_3 fails, services will use any of the available links thereby achieving service reliability.

```
FABRIC_CORE_2:1#show isis adj
FABRIC_CORE_2:1#show isis adjacencies

================================================================================================
                                ISIS Adjacencies
================================================================================================
INTERFACE          L STATE        UPTIME PRI HOLDTIME SYSID              HOST-NAME         STATUS
------------------------------------------------------------------------------------------------
Port1/1            1 UP         00:04:01 127       23 00db.face.0003     FABRIC_CORE_3     ACTIVE
Port1/2            1 UP         00:03:51 127       21 00db.face.0003     FABRIC_CORE_3     BACKUP
Port1/5            1 UP         00:03:39 127       22 00db.face.0003     FABRIC_CORE_3     BACKUP
Port1/6            1 UP         00:03:48 127       19 00db.face.0003     FABRIC_CORE_3     BACKUP
Port1/7            1 UP         00:03:36 127       25 00db.face.0003     FABRIC_CORE_3     BACKUP
Port1/9            1 UP         00:03:38 127       19 00db.face.0003     FABRIC_CORE_3     BACKUP

------------------------------------------------------------------------------
 6 out of 6 interfaces have formed an adjacency
------------------------------------------------------------------------------

FABRIC_CORE_2:1#
```

Figure 4-24 Network Topology as seen by BEB-1

The above discussion also applies to all the BEBs in the network. What is important is to note that. There was not a single line of configuration added to these network core devices everything was automatic.

4.3 Performance and Validation

In this section, the PoC performance evaluation and validation of the S-IoT-N is presented and discussed. The section will test and validate if the S-IoT-N achieves the benefits of simplicity, flexibility and vendor-neutrality while successfully addressing the challenges associated with the traditional (legacy) technologies. The focus will be to evaluate if the S-IoT-N does address the stated concerns and the envisaged benefits. Accordingly, at first, intentional disruption in the IP/MPLS domain is implemented and observations are carried out in order to monitor network behaviour and service reliability. A major observation (as expected) is that the services are interrupted when convergence takes places as the routers recalculate new routes and updating their forwarding information base. This is discussed in section 4-3-1.

Section 4-3-2 looks at the behaviour of the S-IoT-N when similar tests performed as per section 4-3-1 which are implemented on the S-IoT-N test bed. Basic IP testing using the ICMP (ping) is implemented in both tests. A continuous ping to the camera is done with the intention to see if services to fail since a “request time out” indicates a loss of service. More IP testing tools (packet tracer for instance) can be used and packets and this data can be further plotted in MATLAB for further analysis.

4.3 1 IP/MPLS PoC Performance

Figure 4-7 is repeated again in this section as Figure 4-25 to recap and illustrate IP/MPLS performance. As explained in the PoC, configurations which created depicting the various coloured lines of site-6 (configurations in Appendix-A). In order to test network performance, the link between site-6 and site-4 is disconnected.

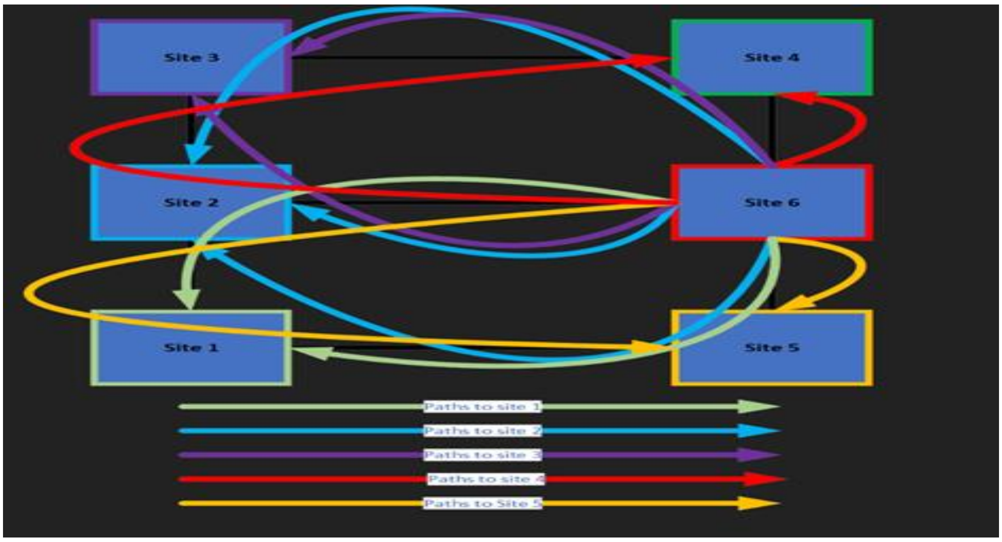
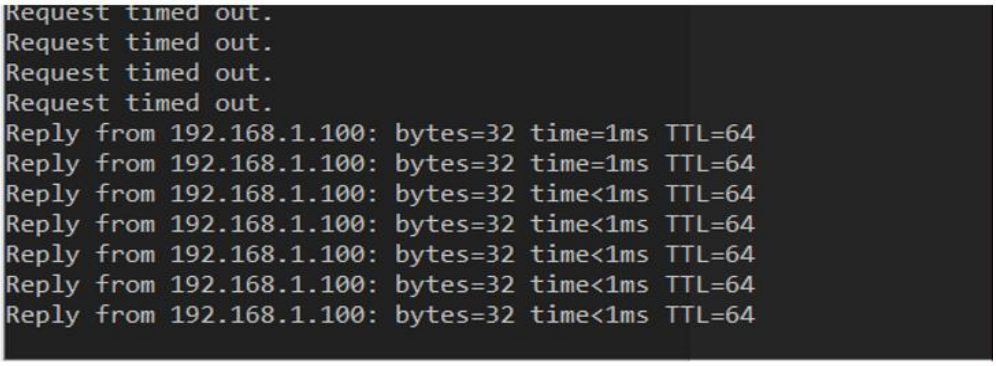


Figure 4-25 IP/MPLS Testbed

The result is that the service fails and later recovers as shown in Figure-4-25. The services are lost when the link is disconnected because router-6 detects a link failure towards router-4 and router-6’s OSPF calculation calculates the optimum routes to all the sites via the backup link to router-5. It then updates its FIB, broadcast this new information to other routers to inform them that the link between router-6 and router-4 is no longer in service and other routers must in turn recalculate their optimum routes and update their FIBs. This is the process of network convergence.

In Figure 4-26, Internet Control Message Protocol (ICMP) continuous ping operation from the source (camera side) towards the destination (server side) is performed. The ping requests are sent to the destination at a rate of one frame per second. Based on this test, it is observed that a connection is lost from the source to the destination due to network re-convergence as explained in the previous paragraph. The service did recover after 4-seconds.



```
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Reply from 192.168.1.100: bytes=32 time=1ms TTL=64
Reply from 192.168.1.100: bytes=32 time=1ms TTL=64
Reply from 192.168.1.100: bytes=32 time<1ms TTL=64
Reply from 192.168.1.100: bytes=32 time<1ms TTL=64
Reply from 192.168.1.100: bytes=32 time<1ms TTL=64
Reply from 192.168.1.100: bytes=32 time<1ms TTL=64
Reply from 192.168.1.100: bytes=32 time<1ms TTL=64
```

Figure 4-26 IP/MPLS Testbed

The graphical view of the network response is represented by Figure 4-27

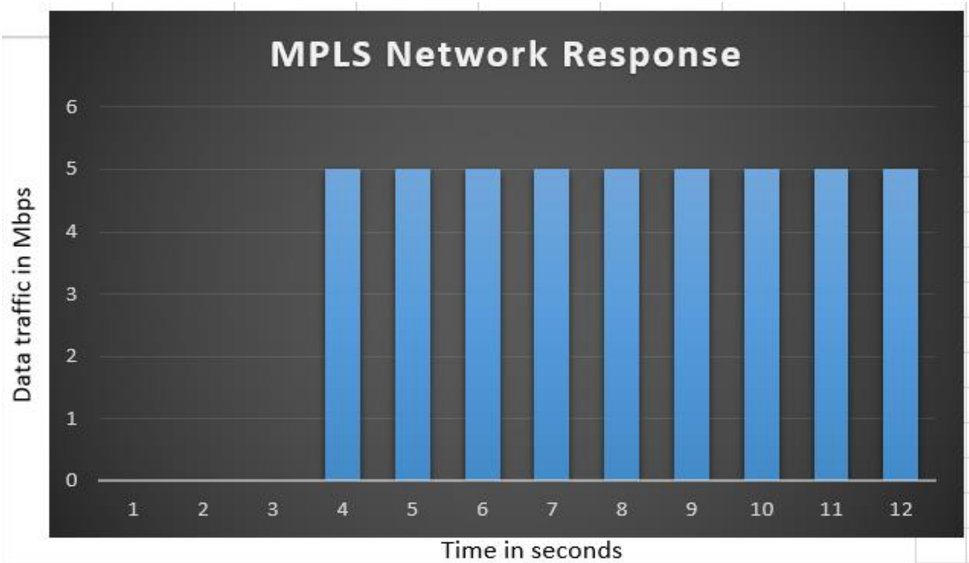


Figure 4-27 MPLS Network Response

4.3.2 S-IoT-N PoC Performance

Firstly, the PoC implementation in section 4-2-2 revealed that in terms of service implementation (i.e. configuration of services) time and configuration effort has been reduced greatly by automation brought by IS-IS and LLDP. It is noted that only the edges of the network are configured and with very little parameters. The result of automation is the huge scaling and quick service deployment brought by the S-IoT-N. This achieves the goal of scalability and quick service deployment required in IoT.

The simplicity of only adding just a VLAN and an I-SID which are mandatory in order to identify and separate different kinds of services running on any network results in a simplified network which achieves the goal of simplicity. Simple in terms of implementation and in the case of troubleshooting should there be service failures, they will be easy to identify and fix.

Again, automation tremendously speeds up service deployment which will assist organizations in meeting their IoT customer demands and service delivery

In order to test network reliability, a similar test done on the IP/MPLS domain in section 4-3-1 was carried out in the S-IoT-N (Figure 4-28) to test its performance. The domain tested was its "Core" network represented by the BEB-1 and BEB-2 devices and the interest was mainly to observe the network behaviour in the Network-to-Network Interface (NNI) links. The links are connected from port-1 to port-12 between the 2-devices (Figure 4-28).

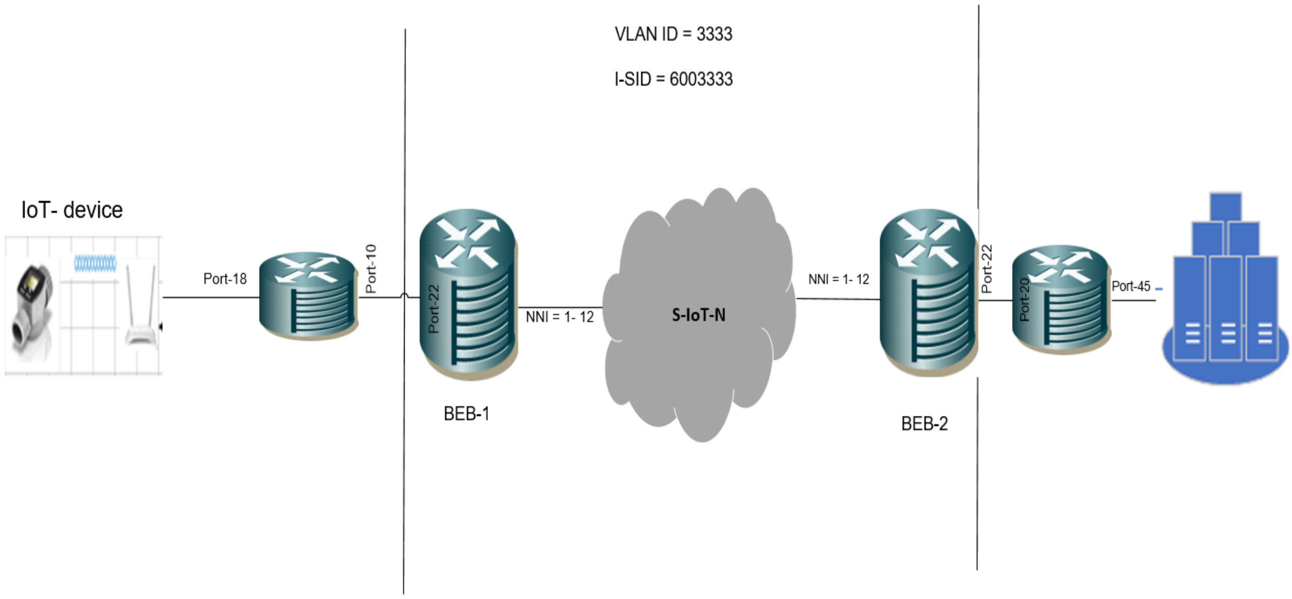


Figure 4-28 S-IoT-N PoC

Again, Internet Control Message Protocol (ICMP) continuous ping operation from the source (camera side) towards the destination (server side) is performed. The ping requests are sent to the destination at a rate of one frame per second. Based on this test, it is observed that a connection not is lost from the source to the destination. This is because the control plane IS-IS has created a copy of network path (topology) on each and every core router (BEBs). The BEBs know alternatives of how to get to each router because they have the same LSDBs (as shown in Figure 4-24 for BEB-1 LSDB). When a link was disconnected traffic was automatically re-routed to the available links and there was no service failure. This is shown in Figure 4-29.

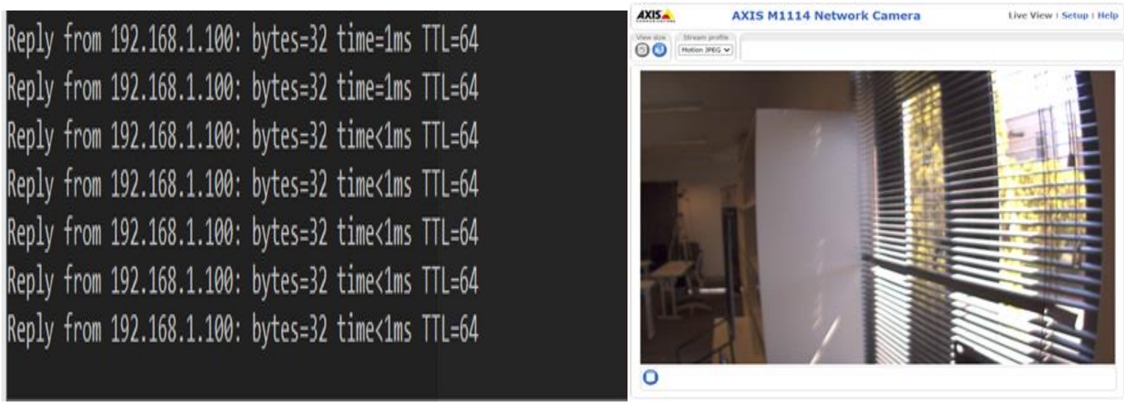


Figure 4-29 Illustration of Service behaviour when links are failed.

This is again shown graphically in Figure 4-30

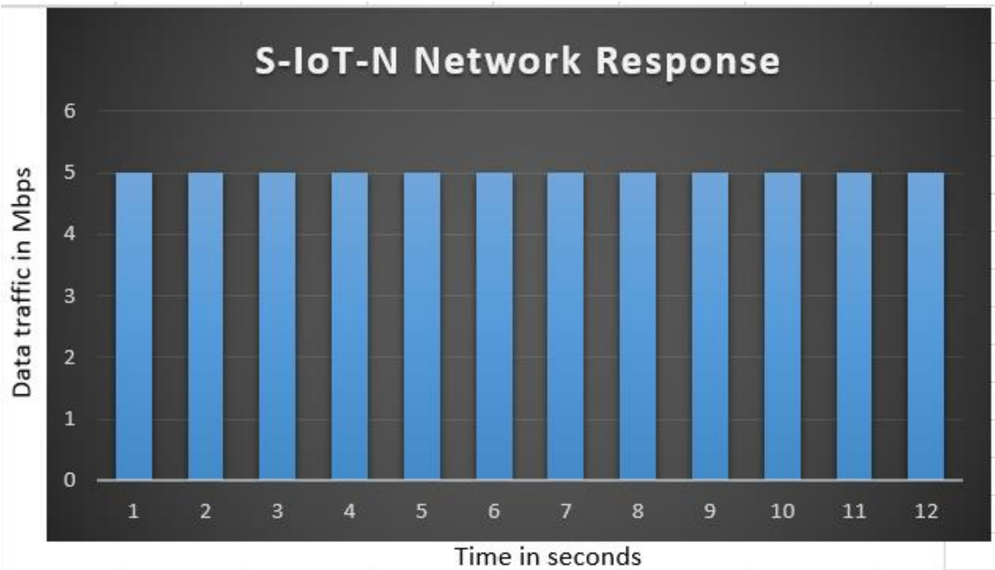


Figure 4-30 S-IoT-N Network response

In conclusion, the proposed evolutionary approach meets the objectives of this study which are to overcome the challenges of reliability, scalability, interoperability, inflexibility and vendor lock-in. The S-IoT-N results in superior network performance than its TCP/IP/MPLS counterparts. Furthermore, the automation that is realised by the implementation of IS-IS & LLDP results in minimal configuration. This enables quicker service deployment, a much scalable network and minimal configuration errors. Most importantly, the separation of network core from edge along with the automation of network control plane greatly simplifies service deployment and opens up opportunities for further developmental studies.

Finally, the quick configuration on the edges and a zero-touch core proves that this evolutionary approach is suitable for IoT deployments and can scale very quickly. The LSDB capability of keeping copies of the network topology in each router results in a reliable network core and 100-percent service availability. The study has proven that the incremental deployment of services is possible and existing services can be migrated to newly created network.

4.3.3 Validation

As discussed in the above sections, the following conclusions can be made. It can be safely validated that the proposed S-IoT-N does realize the benefits such as simplicity, flexibility and vendor-neutrality as it uses open and simple standards. The concerns over traditional transport technologies of reliability, scalability, interoperability, inflexibility and vendor lock-in are effectively addressed as demonstrated in the POC and its performance.

The use of an I-SID resulting in 805,306,320 services that can be served by 1 x 48-port switch (discussed in section 3-3) realizes a Scalable Transport Network for IoT deployment. The importance of Plug & Play ability makes it possible to deploy enormous amount of services to meet IoT demands. The use of IS-IS and LLDP which are open standards results in overcoming the vendor lock-in, inflexibility and the rigid state of traditional networks. The demarcation of the access and core in the S-IoT-N allows for the independent evolution of both the access and the core infrastructures.

The experimental demonstration showed that the minimal configuration implemented at the network edges and the zero-touch cores results in quick deployment of services coupled with scalability which is introduced by the implementation of the I-SID. Furthermore, Figure 4-31 concludes the superior network performance of the proposed evolutionary approach over traditional network performance.

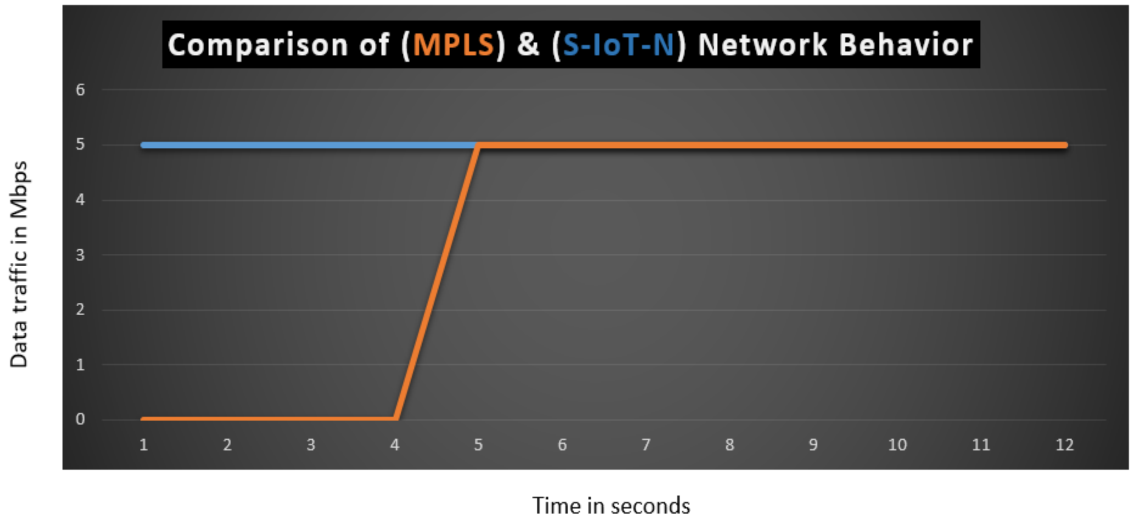


Figure 4-31 Comparison of MPLS & S-IoT-N

Figure 4-32 show the average link utilization in the S-IoT-N the IoT traffic is running on a 1Gps link. This is the data plane traffic which is the actual IoT traffic together with transmission signalling/overhead which can be noticed by the two peaks. As it can be noticed, the link is virtually running empty meaning it has enormous potential to accommodate and transport thousands of IoT services that can be deployed on it.

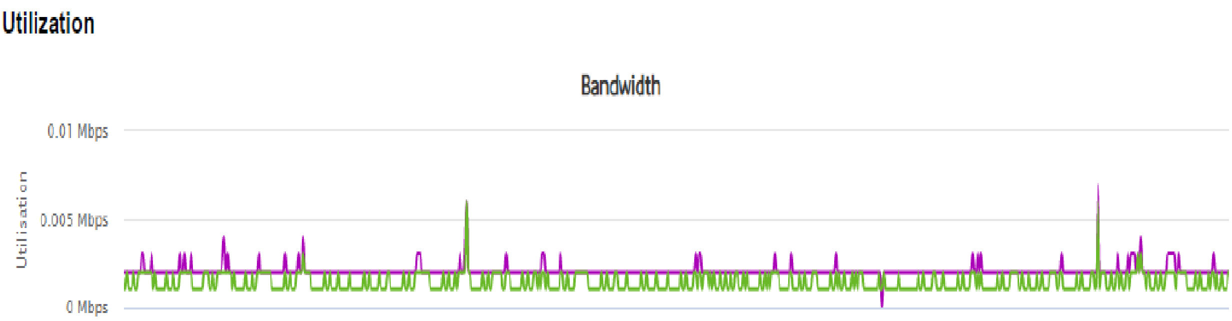


Figure 4-32 S-IoT-N Link Traffic Utilization in a 1-Gbps link

Thus, it can be validated and concluded that the S-IoT-N achieves the study objectives of implementing a scalable, reliable, and a flexible Transport Network for IoT deployment and overcomes the concerns over traditional transport technologies of reliability, scalability, interoperability, inflexibility and vendor lock-in and these have been effectively addressed as demonstrated.

Chapter-5 Conclusion and Future Work

5.1 Conclusion

In this study, an innovative approach into implementing a Scalable Transport Network for IoT deployment has been presented and discussed. Most importantly, the proposed evolutionary approach realizes the stated benefits such as simplicity, flexibility and vendor-neutrality by implementing existing open standards (IS-IS & LLDP) in a basic and a simplified form. Moreover, scalability and flexibility resulted in simplified and speedy deployment of IoT services while at the same time they have addressed the stated concerns over existing transport network infrastructures being IP/MPLS, TCP/IP.

Furthermore, the proposed S-IoT-N and its implementation strategy has addresses and attempted to solve the stated concerns over IP/MPLS. Moreover, the innovation approach in IoT service deployment has facilitated a flexible and an independent evolution of both the network core and edge.

Fundamentally, the proposed approach used the existing (open and already developed) control plane standards (IS-IS & LLDP) in an innovative way. This approach allows for co-existence with existing network infrastructures for example, IP/MPLS as demonstrated in Chapter-4. Existing network services can be migrated from the legacy transport networks into the newly deployed S-IoT-N without fundamentally changing the existing network architecture and infrastructure. Accordingly, the control and data forwarding planes are decoupled from each other thereby enabling integration of SDN concepts in future studies.

The proof of concept which has been demonstrated on a physical network testbed along with the relevant innovate service provisioning and configuration of the basic IoT services showed that it is practically possible to revolutionize the existing network infrastructures in order to meet the current demands for IoT deployment. The implementation of the network testbed as two different scenarios being (IP/MPLS and the proposed S-IoT-N) promoted and demonstrated that the proposed S-IoT-N architecture is able to support a flexible and innovative revolution of both network core and edge.

Furthermore, the proposed approach which has been validated through gained experience in this study and also through experimental performance evaluation of the PoC in two relevant directions (performance analysis and the basic IoT network connectivity services). The validation also looked at innovative provisioning of network services, vendor-neutrality, simplicity, and flexibility, scalability, reliability and interoperability. Through this process, the results have sufficiently proven that the proposed evolutionary approach is progression towards the stated benefits.

5.2 Future Work

From the lessons learned in this study, further improvements are still required in order to improve real-world production networks. There is clear evidence of hard work in the relevant studies that have been done. They guide a clear direction in enabling innovation in IoT service provisioning/deployment while facilitating flexible and independent evolution of both the network core and edge.

Furthermore, it is still a challenge to co-ordinate and integrate the new innovations into the existing network infrastructures. Service providers need assurance in deploying proven reliable systems, and rightfully so, it is important to protect critical network operations aiding to continued and critical business operations. Nevertheless, the proposed evolutionary approach and its implementation strategy still needs to be further improved in order to implement and validate it in real-world environments. Thus, the future work directions for the proposed evolutionary approach are:

1. There is still a high level of scarce skills in the field of troubleshooting IS-IS and network providers may find it difficult to fully deploy an IS-IS control plane across large networks. More investment is required to upskill network engineers in order to be familiar in this space. Considered as an old technology, IS-IS has proven in this study that is worth further revisiting it and be considered in solving the current exponential IoT growth demands.
2. The network core is fully automated, however little has been done or investigated in securing the network against attacks (implementation of service aware firewalls). Attacking the control plane will bring everything to a halt, this remains the greatest risk.
3. Integration and enhancement of the developed (legacy) protocols onto the future network technologies (SDN, SD-WAN). There has been little research work in re-introducing the powerful control plane protocols (IS-IS, BGP, and the relevant IGPs (OSPF, RIP) etc) into new and future technologies like SDN, NFV etc. For successful

deployment of highly scalable, resilient and flexible transport networks, it is crucial to leverage on the power of existing control plane protocols and improvement of their slow convergence so as to be able to accommodate time dependent application which are on the rise.

It is hoped that this study will contribute towards innovation and continued learning.

References

[1] Huawei, "Unleashing the Power of Wi-Fi Enterprise-Grade Wi-Fi 6 Forecast for 2019_2023 White Paper," 2019.

[2] A. M. A.-M. and S. L. K. K. Nair, "Analysis of the Narrow Band Internet of Things (NB-IoT) Technology," [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8703630.

[3] U. M. M. A. Ikpehai, Bamidele Adebisi, Khaled M. Rabie, Kelvin Anoh,Ruth E. Ande, Mohammad Hammoudeh, Haris Gacanin, "Low-Power Wide Area Network Technologies for Internet-of-Things: A Comparative Review," [Online]. Available: https://ieeexplore.ieee.org/document/8550722.

[4] Open Networking Foundation, "Software-defined networking: the new norm for networks," ONF Paper, Open Networking Foundation, Apr. 2012. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

[5] "Network functions virtualisation (nfv): network operator perspectives on industry progress," Network Functions Virtualisation - Update White Paper, Oct. 2013. [Online]. Available: http://portal.etsi.org/NFV/NFV_White_Paper2.pdf

[6] M. Casado, T. Koponen, S. Shenker, and A. Tootoonchian, "Fabric: a retrospective on evolving sdn," in Proceedings of the first workshop on Hot topics in software defined networks. ACM, 2012, pp. 85–90.

[7] G. Hampel, M. Steiner, and T. Bu, "Applying software-defined networking to the telecom domain," in Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on. IEEE, 2013, pp. 133–138.

[8] J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, A. R. Curtis, and S. Banerjee, "Devoflow: cost-effective flow management for high performance enterprise networks," in Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks. ACM, 2010, p. 1.

[9] R. Gouveia, J. Aparicio, J. Soares, B. Parreira, S. Sargento, and J. Carapinha, “Sdn framework for connectivity services,” in Communications (ICC), 2014 IEEE International Conference on. IEEE, 2014, pp. 3058–3063.

[10] A. R. Sharafat, S. Das, G. Parulkar, and N. Mckeown, “MPLS-TE and MPLS VPNs with OpenFlow,” pp. 452–453.

[11] Edsger W. Dijkstra, “Dijkstra’s Shortest Path First algorithm, SPF algorithm.”

[12] R. Braden, “Requirements for internet hosts – communication layers,” RFC1122 (INTERNET STANDARD), Internet Engineering Task Force, Oct. 1989, updated by RFCs 1349, 4379, 5884, 6093, 6298, 6633, 6864. [Online]. Available: http://www.ietf.org/rfc/rfc1122.txt

[13] “Requirements for internet hosts – application and support,” RFC 1123 (INTERNET STANDARD), Internet Engineering Task Force, Oct. 1989, updated by RFCs 1349, 2181, 5321, 5966. [Online]. Available: http://www.ietf.org/rfc/rfc1123.txt

[14] M.P.V. Manthena “Network-as-a-Service Architecture with SDN and NFV” A Proposed Evolutionary Approach for Service Provider Networks, 2015.

[15] E. Rosen, A. Viswanathan, R. Callon, “Multiprotocol label switching architecture,” RFC 3031 (PROPOSED STANDARD), Internet Engineering Task Force, Jan. 2001, updated by RFCs 6178, 6790. [Online]. Available: http://www.ietf.org/rfc/rfc3031.txt.

[16] R. C. E. Rosen, A. Viswanathan, “Multiprotocol Label Switching Architecture.” [Online]. Available: https://www.hjp.at/doc/rfc/rfc3031.html.

[17] R. for C. R. IETF, Network Working Group, “Multiprotocol Label Switching Architecture,” 2001. [Online]. Available: https://tools.ietf.org/pdf/rfc3031.pdf.

[18] R. for C. 4975 IETF,Network Working Group, “Extensions to Resource Reservation Protocol - Traffic Engineering (RSVP-TE),” 2013. [Online]. Available: https://tools.ietf.org/pdf/rfc4875.pdf.

[19] R. for C. 4271 IETF,Network Working Group, “A Border Gateway Protocol 4 (BGP-4),” 2019. [Online]. Available: https://tools.ietf.org/pdf/rfc4271.pdf.

[20] W. Braun and M. Menth, “Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices,” Futur. Internet, vol. 6, no. 2, pp. 302–336, 2014, doi: 10.3390/fi6020302.

[21] J. A. A. Farrel, J.-P. Vasseur, “A Path Computation Element (PCE)-Based Architecture.” [Online]. Available: https://www.hjp.at/doc/rfc/rfc4655.html.

[22] Open Networking Foundation, “OpenFlow Switch Specification (Version 1.5.1),” Current, vol. 0, pp. 1–36, 2015, [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf.

[23] “Openflow switch specification: version 1.3.0 (wire protocol 0x04),” OpenFlow, Open Networking Foundation, June 2012. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf

[24] ONF, “OpenFlow Management and Configuration Protocol,” OF-CONFIG 1.2 OpenFlow Manag. Config. Protoc., pp. 1–44, 2014, [Online]. Available: http://opennetworking.wpengine.com/wp-content/uploads/2013/02/of-config-1.2.pdf.

[25] D. Harrington, R. Presuhn, B. Wijnen, “An architecture for describing simple network management protocol (snmp) management frameworks,” RFC 3411 (INTERNET STANDARD), Internet Engineering Task Force, Dec. 2002, updated by RFCs 5343, 5590 and Obsoletes RFC 2571. [Online]. Available: http://www.ietf.org/rfc/rfc3411.txt

[26] R. for C. 6241 IETF,Network Working Group, “Network Configuration Protocol (NETCONF),” 2013. [Online]. Available: https://tools.ietf.org/pdf/rfc6241.pdf.

[27] R. for C. R. IETF,Network Working Group, “YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF),” 2013. [Online]. Available: https://tools.ietf.org/pdf/rfc6020.pdf.

[28] K. K. Nair, A. M. Abu-Mahfouz and S. Lefophane, "Analysis of the Narrow Band Internet of Things (NB-IoT) Technology," 2019 Conference on Information Communications Technology and Society (ICTAS), 2019, pp. 1-6, doi: 10.1109/ICTAS.2019.8703630.

[29] U. M. M. A. Ikpehai, Bamidele Adebisi, Khaled M. Rabie, Kelvin Anoh,Ruth E. Ande, Mohammad Hammoudeh, Haris Gacanin, “Low-Power Wide Area Network Technologies for Internet-of-Things: A Comparative Review,” [Online]. Available: https://ieeexplore.ieee.org/document/8550722.

[30] N. G. IEEE, “Bluetooth 5.0 and Beyond [Standards].” [Online]. Available:

https://ieeexplore.ieee.org/abstract/document/8721261.

[31] Ergen, S. C. (2004). ZigBee/IEEE 802.15. 4 Summary. UC Berkeley, September, 10, 17.

[32] T. Karygiannis, B. Eydt, G. Barber, L. Bunn, and T. Phillips, “Guidelines for Securing Radio Frequency Identification (RFID) Systems,” NIST Spec. Publ., vol. 800, p. 98, 2007,[Available:http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Guidelines+for+Securing+Radio+Frequency+Identification+(RFID)+Systems#0.

[33] A. C. St and A. N. St, “Latest advancement in Light Fidelity (Li-Fi) Technology International Journal of Advance Research in Latest advancement in Light Fidelity (Li-Fi) Technology,” no. January, 2018.

[34] D. Bankov, E. Khorov, A. Lyakhov and E. Stepanova 2017 IEEE International Conference on Communications (ICC), Paris, 2017, pp. 1-6, “Fast centralized authentication in Wi-Fi HaLow networks,” [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7996510.

[35] L. Qiao, Z. Zheng, W. Cui and L. Wang, "A Survey on Wi-Fi HaLow Technology for Internet of Things," 2018 2nd IEEE Conference on Energy Internet and Energy System Integration (EI2), 2018, pp. 1-5, doi: 10.1109/EI2.2018.8582141.

[36] Congdon, Paul. "Link layer discovery protocol and mib." V1. 0 May 20.2002 (2002): 1-20.

[37] 802.1ah Provider Backbone Bridges Available: https://www.ieee802.org/1/pages/802.1ah.html.

[38] Cisco, “Understanding Basic 802 . 1ah Provider Backbone Bridge,” [Online]. Available: https://www.cisco.com/c/en/us/support/docs/routers/asr-9000-series-aggregation-services-routers/212882-understanding-basic-802-1ah-provider-bac.html

Appendix-A MPLS Configuration

The configuration below is an example of the configuration for Site 6:

1. **Configure the various VLANS/IP's between all nodes.**

   configure vlan peering-data add ports 11-18,20-21,23-24,41,43-48 untagged
   configure vlan peering-voice add ports 40 untagged
   configure vlan VLAN_1102 add ports 49 tagged
   configure vlan VLAN_1301 add ports 2 tagged
   configure vlan VLAN_1501 add ports 61 tagged
   configure vlan LoopBack ipaddress X.Y.Z.10 255.255.255.255
   enable ipforwarding vlan LoopBack
   configure vlan Mgmt ipaddress Y.X.Z.106 255.255.255.224
   configure vlan VLAN_1102 ipaddress A.B.C.233 255.255.255.254
   enable ipforwarding vlan VLAN_1102
   configure ip-mtu 9216 vlan VLAN_1102
   configure vlan VLAN_1301 ipaddress A.B.C.228 255.255.255.254
   enable ipforwarding vlan VLAN_1301
   configure ip-mtu 9216 vlan VLAN_1301
   configure vlan VLAN_1501 ipaddress A.B.C.235 255.255.255.254
   enable ipforwarding vlan VLAN_1501
   configure ip-mtu 9216 vlan VLAN_1501

2. **Configure the IGP, in this case OSPF:**

   configure ospf routerid A.B.C.10
   enable ospf mpls-next-hop
   enable ospf
   configure ospf add vlan LoopBack area 0.0.0.0
   configure ospf add vlan VLAN_1102 area 0.0.0.0 link-type point-to-point
   configure ospf add vlan VLAN_1301 area 0.0.0.0 link-type point-to-point
   configure ospf add vlan VLAN_1501 area 0.0.0.0 link-type point-to-point

3. **Enable MPLS on the VLANS**

   configure mpls add vlan "c1site3-s2site3"
   enable mpls vlan "c1site3-s2site3"
   enable mpls ldp vlan "c1site3-s2site3"
   configure mpls add vlan "c2site3-s2site3"
   enable mpls vlan "c2site3-s2site3"
   enable mpls ldp vlan "c2site3-s2site3"
   configure mpls add vlan "LoopBack"
   enable mpls vlan "LoopBack"
   enable mpls ldp vlan "LoopBack"
   configure mpls add vlan "s2site3-s1ABC"
   enable mpls vlan "s2site3-s1smr"
   enable mpls ldp vlan "s2site3-s1ABC"

configure mpls add vlan "VLAN_1102"
enable mpls vlan "VLAN_1102"
enable mpls rsvp-te vlan "VLAN_1102"
enable mpls ldp vlan "VLAN_1102"
configure mpls add vlan "VLAN_1301"
enable mpls vlan "VLAN_1301"
enable mpls rsvp-te vlan "VLAN_1301"
enable mpls ldp vlan "VLAN_1301"
configure mpls add vlan "VLAN_1501"
enable mpls vlan "VLAN_1501"
enable mpls rsvp-te vlan "VLAN_1501"
enable mpls ldp vlan "VLAN_1501"

4. **Statically define all the paths between each node in the network:**

create mpls rsvp-te path "path-site1"
create mpls rsvp-te path "path-site2"
create mpls rsvp-te path "path-site22"
create mpls rsvp-te path "path-site3"
create mpls rsvp-te path "path-site4"
create mpls rsvp-te path "path-site5"
create mpls rsvp-te path "path1-sw1.site1"
create mpls rsvp-te path "path1-sw1.site2"
create mpls rsvp-te path "path1-sw2.site2"
create mpls rsvp-te path "path2-sw1.site1"
create mpls rsvp-te path "path2-sw1.site2"
create mpls rsvp-te path "path2-sw2.site2"
create mpls rsvp-te path "path3-sw2.site2"

5. **Define the path route hop by hop:**

configure mpls rsvp-te path path1-sw1.site1 add ero include Y.X.Z.229/32 strict order 100
configure mpls rsvp-te path path1-sw1.site1 add ero include Y.X.Z.239/32 strict order 101
configure mpls rsvp-te path path1-sw1.site2 add ero include Y.X.Z.232/32 strict order 100
configure mpls rsvp-te path path1-sw1.site2 add ero include Y.X.Z.230/32 strict order 101
configure mpls rsvp-te path path1-sw2.site2 add ero include Y.X.Z.232/32 strict order 100
configure mpls rsvp-te path path2-sw1.site1 add ero include Y.X.Z.232/32 strict order 100
configure mpls rsvp-te path path2-sw1.site1 add ero include Y.X.Z.237/32 strict order 101
configure mpls rsvp-te path path2-sw1.site2 add ero include Y.X.Z.234/32 strict order 100
configure mpls rsvp-te path path2-sw1.site2 add ero include Y.X.Z.224/32 strict order 101

configure mpls rsvp-te path path2-sw2.site2 add ero include Y.X.Z.229/32 strict order 100
configure mpls rsvp-te path path2-sw2.site2 add ero include Y.X.Z.239/32 strict order 101
configure mpls rsvp-te path path2-sw2.site2 add ero include Y.X.Z.236/32 strict order 102
configure mpls rsvp-te path path3-sw2.site2 add ero include Y.X.Z.234/32 strict order 100
configure mpls rsvp-te path path3-sw2.site2 add ero include Y.X.Z.224/32 strict order 101
configure mpls rsvp-te path path3-sw2.site2 add ero include Y.X.Z.231/32 strict order 102

6. **Create Mulitple LSP's to each site (For load balance) that will use the paths created above:**

create mpls rsvp-te lsp "lsp-test" destination Y.Z.X.11
create mpls rsvp-te lsp "lsp1-site1" destination Y.Z.X.16
create mpls rsvp-te lsp "lsp1-site2" destination Y.Z.X.14
create mpls rsvp-te lsp "lsp1-site22" destination Y.Z.X.11
create mpls rsvp-te lsp "lsp1-site3" destination Y.Z.X.15
create mpls rsvp-te lsp "lsp1-sw1.site1" destination Y.Z.X.16
create mpls rsvp-te lsp "lsp1-sw1.site2" destination Y.Z.X.14
create mpls rsvp-te lsp "lsp1-sw2.site2" destination Y.Z.X.11
create mpls rsvp-te lsp "lsp1-site4" destination Y.Z.X.12
create mpls rsvp-te lsp "lsp1-Site2" destination Y.Z.X.13
create mpls rsvp-te lsp "lsp10-sw2.site2" destination Y.Z.X.11
create mpls rsvp-te lsp "lsp11-sw2.site2" destination Y.Z.X.11
create mpls rsvp-te lsp "lsp12-sw2.site2" destination Y.Z.X.11
create mpls rsvp-te lsp "lsp2-site1" destination Y.Z.X.16
create mpls rsvp-te lsp "lsp2-site2" destination Y.Z.X.14
create mpls rsvp-te lsp "lsp2-site22" destination Y.Z.X.11
create mpls rsvp-te lsp "lsp2-site3" destination Y.Z.X.15
create mpls rsvp-te lsp "lsp2-sw1.site1" destination Y.Z.X.16
create mpls rsvp-te lsp "lsp2-sw1.site2" destination Y.Z.X.14
create mpls rsvp-te lsp "lsp2-sw2.site2" destination Y.Z.X.11
create mpls rsvp-te lsp "lsp2-site4" destination Y.Z.X.12
create mpls rsvp-te lsp "lsp2-Site2" destination Y.Z.X.13
create mpls rsvp-te lsp "lsp3-site2" destination Y.Z.X.14
create mpls rsvp-te lsp "lsp3-site22" destination Y.Z.X.11
create mpls rsvp-te lsp "lsp3-site3" destination Y.Z.X.15
create mpls rsvp-te lsp "lsp3-sw1.site1" destination Y.Z.X.16
create mpls rsvp-te lsp "lsp3-sw1.site2" destination Y.Z.X.14
create mpls rsvp-te lsp "lsp3-sw2.site2" destination Y.Z.X.11
create mpls rsvp-te lsp "lsp3-site4" destination Y.Z.X.12
create mpls rsvp-te lsp "lsp3-smr" destination Y.Z.X.13
create mpls rsvp-te lsp "lsp4-site2" destination Y.Z.X.14
create mpls rsvp-te lsp "lsp4-site22" destination Y.Z.X.11
create mpls rsvp-te lsp "lsp4-site3" destination Y.Z.X.15
create mpls rsvp-te lsp "lsp4-sw1.site1" destination Y.Z.X.16
create mpls rsvp-te lsp "lsp4-sw1.site2" destination Y.Z.X.14

create mpls rsvp-te lsp "lsp4-sw2.site2" destination Y.Z.X.11
create mpls rsvp-te lsp "lsp4-site4" destination Y.Z.X.12
create mpls rsvp-te lsp "lsp4-Site2" destination Y.Z.X.13
create mpls rsvp-te lsp "lsp5-sw1.site1" destination Y.Z.X.16
create mpls rsvp-te lsp "lsp5-sw1.site2" destination Y.Z.X.14
create mpls rsvp-te lsp "lsp5-sw2.site2" destination Y.Z.X.11
create mpls rsvp-te lsp "lsp6-sw1.site1" destination Y.Z.X.16
create mpls rsvp-te lsp "lsp6-sw1.site2" destination Y.Z.X.14
create mpls rsvp-te lsp "lsp6-sw2.site2" destination Y.Z.X.11
create mpls rsvp-te lsp "lsp7-sw1.site1" destination Y.Z.X.16
create mpls rsvp-te lsp "lsp7-sw1.site2" destination Y.Z.X.14
create mpls rsvp-te lsp "lsp7-sw2.site2" destination Y.Z.X.11
create mpls rsvp-te lsp "lsp8-sw1.site1" destination Y.Z.X.16
create mpls rsvp-te lsp "lsp8-sw1.site2" destination Y.Z.X.14
create mpls rsvp-te lsp "lsp8-sw2.site2" destination Y.Z.X.11
create mpls rsvp-te lsp "lsp9-sw2.site2" destination Y.Z.X.11

7. **MAP the LSP's to the Paths:**

configure mpls rsvp-te lsp "lsp-test" add path "test1" primary
configure mpls rsvp-te lsp "lsp1-site1" add path "path-site1" primary
configure mpls rsvp-te lsp "lsp1-site2" add path "path-site2" primary
configure mpls rsvp-te lsp "lsp1-site22" add path "path-site22" primary
configure mpls rsvp-te lsp "lsp1-site3" add path "path-site3" primary
configure mpls rsvp-te lsp "lsp1-sw1.site1" add path "path1-sw1.site1" primary
configure mpls rsvp-te lsp "lsp1-sw1.site2" add path "path1-sw1.site2" primary
configure mpls rsvp-te lsp "lsp1-sw2.site2" add path "path1-sw2.site2" primary
configure mpls rsvp-te lsp "lsp1-site4" add path "path-site4" primary
configure mpls rsvp-te lsp "lsp1-smr" add path "path-smr" primary
configure mpls rsvp-te lsp "lsp10-sw2.site2" add path "path3-sw2.site2" primary
configure mpls rsvp-te lsp "lsp11-sw2.site2" add path "path3-sw2.site2" primary
configure mpls rsvp-te lsp "lsp12-sw2.site2" add path "path3-sw2.site2" primary
configure mpls rsvp-te lsp "lsp2-site1" add path "path-site1" primary
configure mpls rsvp-te lsp "lsp2-site2" add path "path-site2" primary
configure mpls rsvp-te lsp "lsp2-site22" add path "path-site22" primary
configure mpls rsvp-te lsp "lsp2-site3" add path "path-site3" primary
configure mpls rsvp-te lsp "lsp2-sw1.site1" add path "path1-sw1.site1" primary
configure mpls rsvp-te lsp "lsp2-sw1.site2" add path "path1-sw1.site2" primary
configure mpls rsvp-te lsp "lsp2-sw2.site2" add path "path1-sw2.site2" primary
configure mpls rsvp-te lsp "lsp2-site4" add path "path-site4" primary
configure mpls rsvp-te lsp "lsp2-smr" add path "path-smr" primary
configure mpls rsvp-te lsp "lsp3-site2" add path "path-site2" primary
configure mpls rsvp-te lsp "lsp3-site22" add path "path-site22" primary
configure mpls rsvp-te lsp "lsp3-site3" add path "path-site3" primary
configure mpls rsvp-te lsp "lsp3-sw1.site1" add path "path1-sw1.site1" primary
configure mpls rsvp-te lsp "lsp3-sw1.site2" add path "path1-sw1.site2" primary
configure mpls rsvp-te lsp "lsp3-sw2.site2" add path "path1-sw2.site2" primary

```
configure mpls rsvp-te lsp "lsp3-site4" add path "path-site4" primary
configure mpls rsvp-te lsp "lsp3-smr" add path "path-smr" primary
configure mpls rsvp-te lsp "lsp4-site2" add path "path-site2" primary
configure mpls rsvp-te lsp "lsp4-site22" add path "path-site22" primary
configure mpls rsvp-te lsp "lsp4-site3" add path "path-site3" primary
configure mpls rsvp-te lsp "lsp4-sw1.site1" add path "path1-sw1.site1" primary
configure mpls rsvp-te lsp "lsp4-sw1.site2" add path "path1-sw1.site2" primary
configure mpls rsvp-te lsp "lsp4-sw2.site2" add path "path1-sw2.site2" primary
configure mpls rsvp-te lsp "lsp4-site4" add path "path-site4" primary
configure mpls rsvp-te lsp "lsp4-smr" add path "path-smr" primary
configure mpls rsvp-te lsp "lsp5-sw1.site1" add path "path2-A.B.C.site1"
primary
configure mpls rsvp-te lsp "lsp5-sw1.site2" add path "path2-sw1.site2" primary
configure mpls rsvp-te lsp "lsp5-sw2.site2" add path "path2-sw2.site2" primary
configure mpls rsvp-te lsp "lsp6-sw1.site1" add path "path2-sw1.site1" primary
configure mpls rsvp-te lsp "lsp6-sw1.site2" add path "path2-sw1.site2" primary
configure mpls rsvp-te lsp "lsp6-sw2.site2" add path "path2-sw2.site2" primary
configure mpls rsvp-te lsp "lsp7-sw1.site1" add path "path2-sw1.site1" primary
configure mpls rsvp-te lsp "lsp7-sw1.site2" add path "path2-sw1.site2" primary
configure mpls rsvp-te lsp "lsp7-sw2.site2" add path "path2-sw2.site2" primary
configure mpls rsvp-te lsp "lsp8-sw1.site1" add path "path2-sw1.site1" primary
configure mpls rsvp-te lsp "lsp8-sw1.site2" add path "path2-sw1.site2" primary
configure mpls rsvp-te lsp "lsp8-sw2.site2" add path "path2-sw2.site2" primary
configure mpls rsvp-te lsp "lsp9-sw2.site2" add path "path3-sw2.site2" primary
```

8. **Create the Layer 2 VPLS service:**

```
create l2vpn vpls Management_Service fec-id-type pseudo-wire 1000
configure l2vpn vpls Management_Service add service vman jinx-mngmnt
enable l2vpn vpls Management_Service health-check vccv
create l2vpn vpls Data_Service fec-id-type pseudo-wire 1234
configure l2vpn vpls Data_Service add service vlan peering-data
enable l2vpn vpls Data_Service health-check vccv
create l2vpn vpls Voice_Service fec-id-type pseudo-wire 1235
configure l2vpn vpls Voice_Service add service vlan peering-voice
enable l2vpn vpls Voice_Service health-check vccv
configure l2vpn vpls Management_Service add peer Y.Z.X.21 core full-mesh
configure l2vpn vpls Management_Service add peer Y.Z.X.20 core full-mesh
configure l2vpn vpls Management_Service add peer Y.Z.X.19 core full-mesh
configure l2vpn vpls Management_Service add peer Y.Z.X.18 core full-mesh
configure l2vpn vpls Management_Service add peer Y.Z.X.16 core full-mesh
configure l2vpn vpls Management_Service add peer Y.Z.X.11 core full-mesh
configure l2vpn vpls Management_Service add peer Y.Z.X.13 core full-mesh
configure l2vpn vpls Management_Service add peer Y.Z.X.12 core full-mesh
configure l2vpn vpls Management_Service add peer Y.Z.X.15 core full-mesh
configure l2vpn vpls Management_Service add peer Y.Z.X.14 core full-mesh
configure l2vpn vpls Data_Service add peer Y.Z.X.21 core full-mesh
configure l2vpn vpls Data_Service add peer Y.Z.X.20 core full-mesh
configure l2vpn vpls Data_Service add peer Y.Z.X.19 core full-mesh
configure l2vpn vpls Data_Service add peer Y.Z.X.18 core full-mesh
```

configure l2vpn vpls Data_Service add peer Y.Z.X.16 core full-mesh
configure l2vpn vpls Data_Service add peer Y.Z.X.11 core full-mesh
configure l2vpn vpls Data_Service add peer Y.Z.X.12 core full-mesh
configure l2vpn vpls Data_Service add peer Y.Z.X.13 core full-mesh
configure l2vpn vpls Data_Service add peer Y.Z.X.14 core full-mesh
configure l2vpn vpls Data_Service add peer Y.Z.X.15 core full-mesh
configure l2vpn vpls Voice_Service add peer Y.Z.X.21 core full-mesh
configure l2vpn vpls Voice_Service add peer Y.Z.X.20 core full-mesh
configure l2vpn vpls Voice_Service add peer Y.Z.X.19 core full-mesh
configure l2vpn vpls Voice_Service add peer Y.Z.X.18 core full-mesh
configure l2vpn vpls Voice_Service add peer Y.Z.X.16 core full-mesh
configure l2vpn vpls Voice_Service add peer Y.Z.X.11 core full-mesh
configure l2vpn vpls Voice_Service add peer Y.Z.X.12 core full-mesh
configure l2vpn vpls Voice_Service add peer Y.Z.X.15 core full-mesh
configure l2vpn vpls Voice_Service add peer Y.Z.X.14 core full-mesh
configure l2vpn vpls Voice_Service add peer Y.Z.X.13 core full-mesh

9. **Map the VPLS Service to the LSP's to each site**
configure l2vpn vpls Management_Service peer Y.Z.X.16 add mpls lsp lsp1-site1
configure l2vpn vpls Management_Service peer Y.Z.X.16 add mpls lsp lsp2-site1
configure l2vpn vpls Data_Service peer Y.Z.X.16 add mpls lsp lsp1-sw1.site1
configure l2vpn vpls Data_Service peer Y.Z.X.16 add mpls lsp lsp2-sw1.site1
configure l2vpn vpls Data_Service peer Y.Z.X.16 add mpls lsp lsp3-sw1.site1
configure l2vpn vpls Data_Service peer Y.Z.X.16 add mpls lsp lsp4-sw1.site1
configure l2vpn vpls Data_Service peer Y.Z.X.16 add mpls lsp lsp5-sw1.site1
configure l2vpn vpls Data_Service peer Y.Z.X.16 add mpls lsp lsp6-sw1.site1
configure l2vpn vpls Data_Service peer Y.Z.X.16 add mpls lsp lsp7-sw1.site1
configure l2vpn vpls Data_Service peer Y.Z.X.16 add mpls lsp lsp8-sw1.site1
configure l2vpn vpls Data_Service peer Y.Z.X.11 add mpls lsp lsp1-sw2.site2
configure l2vpn vpls Data_Service peer Y.Z.X.11 add mpls lsp lsp10-sw2.site2
configure l2vpn vpls Data_Service peer Y.Z.X.11 add mpls lsp lsp11-sw2.site2
configure l2vpn vpls Data_Service peer Y.Z.X.11 add mpls lsp lsp12-sw2.site2
configure l2vpn vpls Data_Service peer Y.Z.X.11 add mpls lsp lsp2-sw2.site2
configure l2vpn vpls Data_Service peer Y.Z.X.11 add mpls lsp lsp3-sw2.site2
configure l2vpn vpls Data_Service peer Y.Z.X.11 add mpls lsp lsp4-sw2.site2
configure l2vpn vpls Data_Service peer Y.Z.X.11 add mpls lsp lsp5-sw2.site2
configure l2vpn vpls Data_Service peer Y.Z.X.11 add mpls lsp lsp6-sw2.site2
configure l2vpn vpls Data_Service peer Y.Z.X.11 add mpls lsp lsp7-sw2.site2
configure l2vpn vpls Data_Service peer Y.Z.X.11 add mpls lsp lsp8-sw2.site2
configure l2vpn vpls Data_Service peer Y.Z.X.11 add mpls lsp lsp9-sw2.site2
configure l2vpn vpls Data_Service peer Y.Z.X.12 add mpls lsp lsp1-site4
configure l2vpn vpls Data_Service peer Y.Z.X.12 add mpls lsp lsp2-site4
configure l2vpn vpls Data_Service peer Y.Z.X.12 add mpls lsp lsp3-site4
configure l2vpn vpls Data_Service peer Y.Z.X.12 add mpls lsp lsp4-site4
configure l2vpn vpls Data_Service peer Y.Z.X.13 add mpls lsp lsp1-smr
configure l2vpn vpls Data_Service peer Y.Z.X.13 add mpls lsp lsp2-smr
configure l2vpn vpls Data_Service peer Y.Z.X.13 add mpls lsp lsp3-smr
configure l2vpn vpls Data_Service peer Y.Z.X.13 add mpls lsp lsp4-smr

configure l2vpn vpls Data_Service peer Y.Z.X.14 add mpls lsp lsp1-sw1.site2
configure l2vpn vpls Data_Service peer Y.Z.X.14 add mpls lsp lsp2-sw1.site2
configure l2vpn vpls Data_Service peer Y.Z.X.14 add mpls lsp lsp3-sw1.site2
configure l2vpn vpls Data_Service peer Y.Z.X.14 add mpls lsp lsp4-sw1.site2
configure l2vpn vpls Data_Service peer Y.Z.X.14 add mpls lsp lsp5-sw1.site2
configure l2vpn vpls Data_Service peer Y.Z.X.14 add mpls lsp lsp6-sw1.site2
configure l2vpn vpls Data_Service peer Y.Z.X.14 add mpls lsp lsp7-sw1.site2
configure l2vpn vpls Data_Service peer Y.Z.X.14 add mpls lsp lsp8-sw1.site2
configure l2vpn vpls Data_Service peer Y.Z.X.15 add mpls lsp lsp1-site3
configure l2vpn vpls Data_Service peer Y.Z.X.15 add mpls lsp lsp2-site3
configure l2vpn vpls Data_Service peer Y.Z.X.15 add mpls lsp lsp3-site3
configure l2vpn vpls Data_Service peer Y.Z.X.15 add mpls lsp lsp4-site3
configure l2vpn vpls Voice_Service peer Y.Z.X.16 add mpls lsp lsp1-sw1.site1
configure l2vpn vpls Voice_Service peer Y.Z.X.16 add mpls lsp lsp2-sw1.site1
configure l2vpn vpls Voice_Service peer Y.Z.X.16 add mpls lsp lsp3-sw1.site1
configure l2vpn vpls Voice_Service peer Y.Z.X.16 add mpls lsp lsp4-sw1.site1
configure l2vpn vpls Voice_Service peer Y.Z.X.16 add mpls lsp lsp5-sw1.site1
configure l2vpn vpls Voice_Service peer Y.Z.X.16 add mpls lsp lsp6-sw1.site1
configure l2vpn vpls Voice_Service peer Y.Z.X.16 add mpls lsp lsp7-sw1.site1
configure l2vpn vpls Voice_Service peer Y.Z.X.16 add mpls lsp lsp8-sw1.site1
configure l2vpn vpls Voice_Service peer Y.Z.X.11 add mpls lsp lsp1-sw2.site2
configure l2vpn vpls Voice_Service peer Y.Z.X.11 add mpls lsp lsp10-sw2.site2
configure l2vpn vpls Voice_Service peer Y.Z.X.11 add mpls lsp lsp11-sw2.site2
configure l2vpn vpls Voice_Service peer Y.Z.X.11 add mpls lsp lsp12-sw2.site2
configure l2vpn vpls Voice_Service peer Y.Z.X.11 add mpls lsp lsp2-sw2.site2
configure l2vpn vpls Voice_Service peer Y.Z.X.11 add mpls lsp lsp3-sw2.site2
configure l2vpn vpls Voice_Service peer Y.Z.X.11 add mpls lsp lsp4-sw2.site2
configure l2vpn vpls Voice_Service peer Y.Z.X.11 add mpls lsp lsp5-sw2.site2
configure l2vpn vpls Voice_Service peer Y.Z.X.11 add mpls lsp lsp6-sw2.site2
configure l2vpn vpls Voice_Service peer Y.Z.X.11 add mpls lsp lsp7-sw2.site2
configure l2vpn vpls Voice_Service peer Y.Z.X.11 add mpls lsp lsp8-sw2.site2
configure l2vpn vpls Voice_Service peer Y.Z.X.11 add mpls lsp lsp9-sw2.site2
configure l2vpn vpls Voice_Service peer Y.Z.X.12 add mpls lsp lsp1-site4
configure l2vpn vpls Voice_Service peer Y.Z.X.12 add mpls lsp lsp2-site4
configure l2vpn vpls Voice_Service peer Y.Z.X.12 add mpls lsp lsp3-site4
configure l2vpn vpls Voice_Service peer Y.Z.X.12 add mpls lsp lsp4-site4
configure l2vpn vpls Voice_Service peer Y.Z.X.15 add mpls lsp lsp1-site3
configure l2vpn vpls Voice_Service peer Y.Z.X.15 add mpls lsp lsp2-site3
configure l2vpn vpls Voice_Service peer Y.Z.X.15 add mpls lsp lsp3-site3
configure l2vpn vpls Voice_Service peer Y.Z.X.15 add mpls lsp lsp4-site3
configure l2vpn vpls Voice_Service peer Y.Z.X.14 add mpls lsp lsp1-sw1.site2
configure l2vpn vpls Voice_Service peer Y.Z.X.14 add mpls lsp lsp2-sw1.site2
configure l2vpn vpls Voice_Service peer Y.Z.X.14 add mpls lsp lsp3-sw1.site2
configure l2vpn vpls Voice_Service peer Y.Z.X.14 add mpls lsp lsp4-sw1.site2
configure l2vpn vpls Voice_Service peer Y.Z.X.14 add mpls lsp lsp5-sw1.site2
configure l2vpn vpls Voice_Service peer Y.Z.X.14 add mpls lsp lsp6-sw1.site2
configure l2vpn vpls Voice_Service peer Y.Z.X.14 add mpls lsp lsp7-sw1.site2

configure l2vpn vpls Voice_Service peer Y.Z.X.14 add mpls lsp lsp8-sw1.site2
configure l2vpn vpls Voice_Service peer Y.Z.X.13 add mpls lsp lsp1-smr
configure l2vpn vpls Voice_Service peer Y.Z.X.13 add mpls lsp lsp2-smr
configure l2vpn vpls Voice_Service peer Y.Z.X.13 add mpls lsp lsp3-smr
configure l2vpn vpls Voice_Service peer Y.Z.X.13 add mpls lsp lsp4-smr

The above configuration is only for one node and needs to be completed on each node in the MPLS node in the network. If a single new node is added, the above configuration needs to completed for the new node and the following configuration needs to be added to every existing node in the network. The following configuration needs to be applied to all nodes directly connected to the new node:

1. **Configure the various VLANS/IP's between all nodes.**
2. **Configure the IGP, in this case OSPF:**
3. **Enable MPLS on the VLANS**

The following configuration needs to be manually defined on every node in the MPLS network:

1. **Statically define all the paths between each node in the network:**
   create mpls rsvp-te path "path1-New"
   create mpls rsvp-te path "path2-New"
   create mpls rsvp-te path "path3-New"
   create mpls rsvp-te path "path4-New"

2. **Define the path route hop by hop:**
   configure mpls rsvp-te path path1-New add ero include Y.X.Z.234/32 strict order 100
   configure mpls rsvp-te path path1-New add ero include Y.X.Z.224/32 strict order 101
   configure mpls rsvp-te path path2-New add ero include Y.X.Z.229/32 strict order 100
   configure mpls rsvp-te path path2-New add ero include Y.X.Z.239/32 strict order 101
   configure mpls rsvp-te path path2-New add ero include Y.X.Z.236/32 strict order 102
   configure mpls rsvp-te path path3-New add ero include Y.X.Z.236/32 strict order 100
   configure mpls rsvp-te path path3-New add ero include Y.X.Z.236/32 strict order 101
   configure mpls rsvp-te path path3-New add ero include Y.X.Z.236/32 strict order 102
   configure mpls rsvp-te path path4-New add ero include Y.X.Z.234/32 strict order 100
   configure mpls rsvp-te path path4-New add ero include Y.X.Z.224/32 strict order 101
   configure mpls rsvp-te path path4-New add ero include Y.X.Z.231/32 strict order 102

3. **Create Mulitple LSP's to each site (For load balance) that will use the paths created above:**
   create mpls rsvp-te lsp "lsp1-New" destination Y.Z.X.14
   create mpls rsvp-te lsp "lsp2-New" destination Y.Z.X.14
   create mpls rsvp-te lsp "lsp3-New" destination Y.Z.X.14
   create mpls rsvp-te lsp "lsp4-New" destination Y.Z.X.14
   create mpls rsvp-te lsp "lsp5-New" destination Y.Z.X.14
   create mpls rsvp-te lsp "lsp6-New" destination Y.Z.X.14
   create mpls rsvp-te lsp "lsp7-New" destination Y.Z.X.14
   create mpls rsvp-te lsp "lsp8-New" destination Y.Z.X.14

4. **MAP the LSP's to the Paths:**
   configure mpls rsvp-te lsp "lsp1-new" add path "path1-New" primary
   configure mpls rsvp-te lsp "lsp2-new" add path "path1-New" primary
   configure mpls rsvp-te lsp "lsp3-new" add path "path2-New" primary
   configure mpls rsvp-te lsp "lsp4-new" add path "path2-New" primary
   configure mpls rsvp-te lsp "lsp5-new" add path "path3-New" primary
   configure mpls rsvp-te lsp "lsp6-new" add path "path3-New" primary
   configure mpls rsvp-te lsp "lsp7-new" add path "path4-New" primary
   configure mpls rsvp-te lsp "lsp8-new" add path "path4-New" primary

5. **Map the VPLS Service to the LSP's to each site**
   configure l2vpn vpls Data_Service peer Y.Z.X.16 add mpls lsp lsp1-new
   configure l2vpn vpls Data_Service peer Y.Z.X.16 add mpls lsp lsp2-new
   configure l2vpn vpls Data_Service peer Y.Z.X.16 add mpls lsp lsp3-new
   configure l2vpn vpls Data_Service peer Y.Z.X.16 add mpls lsp lsp4-new
   configure l2vpn vpls Data_Service peer Y.Z.X.16 add mpls lsp lsp5-new
   configure l2vpn vpls Data_Service peer Y.Z.X.16 add mpls lsp lsp6-new
   configure l2vpn vpls Data_Service peer Y.Z.X.16 add mpls lsp lsp7-new
   configure l2vpn vpls Data_Service peer Y.Z.X.16 add mpls lsp lsp8-new

The above configuration is all for a single VPLS service only, for every new VPLS service, the same process needs to be completed on every node that the service is present on. In summary, the MPLS network is slow and error prone and it is clear that its not scalable for the thousands of IoT devices to be deployed. Innovative ways of implementing scalable IoT networks is required, appendix-B explores this.

Appendix-B Detailed PoC Configuration

Below is the link state database of the core devices before implementation of the IoT service.

FABRIC_CORE_2:1#show isis lsdb detail 1 2021-02-03T19:53:32.922Z FABRIC_CORE_2 CP1 - 0x002d060a - 00000000 GlobalRouter LLDP INFO LLDP Neighbor Deleted on interface 1/22

===

ISIS LSDB (DETAIL)

===

Level-1 LspID: 00db.face.0002.00-00 SeqNum: 0x00000012 Lifetime: 1149

Chksum: 0xf91b PDU Length: 126

Host_name: FABRIC_CORE_2

Attributes: IS-Type 1

TLV:1 Area Addresses: 1

10.01

TLV:22 Extended IS reachability:

Adjacencies: 1

TE Neighbors: 1

00db.face.0003.00 (FABRIC_CORE_3) Metric:10

SPBM Sub TLV:

port id: 192 num_port 1

Metric: 10

TLV:129 Protocol Supported: SPBM

TLV:137 Host_name: FABRIC_CORE_2

TLV:147 Chassis MAC: 64:6a:52:a5:7c:00

TLV:144 SUB-TLV 1 SPBM INSTANCE:

Instance: 0

bridge_pri: 0

OUI: 00-2d-b2

num of trees: 2

vid tuple : u-bit 1 m-bit 1 ect-alg 0x80c201 base vid 4051

vid tuple : u-bit 1 m-bit 1 ect-alg 0x80c202 base vid 4052

Level-1 LspID: 00db.face.0002.00-01 SeqNum: 0x00000015 Lifetime: 1174

Chksum: 0x1245 PDU Length: 63

Host_name: FABRIC_CORE_2

Attributes: IS-Type 1

TLV:144 SUB-TLV 3 ISID:

Instance: 0

Metric: 0

B-MAC: 02-2d-b2-ff-ff-ff

BVID:4051

Number of ISID's:1

16777215(None)

Instance: 0

Metric: 0

B-MAC: 02-2d-b2-ff-ff-ff

BVID:4052

Number of ISID's:

16777215(None)

Level-1 LspID: 00db.face.0002.00-02 SeqNum: 0x0000000e Lifetime: 1147

Chksum: 0x4247 PDU Length: 38

Host_name: FABRIC_CORE_2

Attributes: IS-Type 1

TLV:135 TE IP Reachability: 1

Metric: 1 Metric Type: Internal Prefix Length: 32

UP/Down Bit: FALSE Sub TLV Bit: FALSE

IP Address: 10.0.0.2

Level-1 LspID: 00db.face.0003.00-00   SeqNum: 0x00000018   Lifetime: 646

Chksum: 0xd699 PDU Length: 126

Host_name: FABRIC_CORE_3

Attributes: IS-Type 1

TLV:1   Area Addresses: 1

10.01

TLV:22 Extended IS reachability:

Adjacencies: 1

TE Neighbors: 1

00db.face.0002.00 (FABRIC_CORE_2)   Metric:10

SPBM Sub TLV:

port id: 192 num_port 1

Metric: 10

TLV:129 Protocol Supported: SPBM

TLV:137 Host_name: FABRIC_CORE_3

TLV:147 Chassis MAC: 64:6a:52:a6:08:00

TLV:144 SUB-TLV 1   SPBM INSTANCE:

Instance: 0

bridge_pri: 0

OUI: 00-3d-b3

num of trees: 2

vid tuple : u-bit 1 m-bit 1 ect-alg 0x80c201 base vid 4051

vid tuple : u-bit 1 m-bit 1 ect-alg 0x80c202 base vid 4052

Level-1 LspID: 00db.face.0003.00-01   SeqNum: 0x0000001f   Lifetime: 1175

Chksum: 0x5ecb PDU Length: 63

Host_name: FABRIC_CORE_3

Attributes: IS-Type 1

TLV:144 SUB-TLV 3   ISID:

Instance: 0

Metric: 0

B-MAC: 02-3d-b3-ff-ff-ff

BVID:4051

Number of ISID's:1

16777215(None)

Instance: 0

Metric: 0

B-MAC: 02-3d-b3-ff-ff-ff

BVID:4052

Number of ISID's:1

16777215(None)

Level-1 LspID: 00db.face.0003.00-02 SeqNum: 0x00000014 Lifetime: 645

Chksum: 0x3a47 PDU Length: 38

Host_name: FABRIC_CORE_3

Attributes: IS-Type 1

TLV:135 TE IP Reachability: 1

Metric: 1 Metric Type: Internal Prefix Length: 32

UP/Down Bit: FALSE Sub TLV Bit: FALSE

IP Address: 10.0.0.3

FABRIC_CORE_2:1#

Below is the link state database after the IoT service has been implemented

FABRIC_CORE_2:1#show isis lsdb detail

==

ISIS LSDB (DETAIL)

==

Level-1 LspID: 00db.face.0002.00-00 SeqNum: 0x00000012 Lifetime: 998

Chksum: 0xf91b PDU Length: 126

```
    Host_name: FABRIC_CORE_2
    Attributes:    IS-Type 1
TLV:1   Area Addresses: 1
        10.01
TLV:22  Extended IS reachability:
    Adjacencies: 1
    TE Neighbors: 1
        00db.face.0003.00 (FABRIC_CORE_3)      Metric:10
            SPBM Sub TLV:
                port id: 192 num_port 1
                Metric: 10
TLV:129 Protocol Supported: SPBM
TLV:137 Host_name: FABRIC_CORE_2
TLV:147 Chassis MAC: 64:6a:52:a5:7c:00
TLV:144 SUB-TLV 1      SPBM INSTANCE:
        Instance: 0
        bridge_pri: 0
        OUI: 00-2d-b2
        num of trees: 2
        vid tuple : u-bit 1 m-bit 1 ect-alg 0x80c201 base vid 4051
        vid tuple : u-bit 1 m-bit 1 ect-alg 0x80c202 base vid 4052
Level-1 LspID: 00db.face.0002.00-01     SeqNum: 0x00000016     Lifetime:  1122
    Chksum: 0x9454  PDU Length: 99
    Host_name: FABRIC_CORE_2
    Attributes:    IS-Type 1
TLV:144 SUB-TLV 3      ISID:
        Instance: 0
        Metric: 0
        B-MAC: 02-2d-b2-ff-ff-ff
        BVID:4051
        Number of ISID's:1
            16777215(None)
        Instance: 0
        Metric: 0
        B-MAC: 02-2d-b2-ff-ff-ff
        BVID:4052
        Number of ISID's:1
```

```
                16777215(None)
            Instance: 0
            Metric: 0
            B-MAC: 00-db-fa-ce-00-02
            BVID:4051
            Number of ISID's:1
                6003333(Both)
            Instance: 0
            Metric: 0
            B-MAC: 00-db-fa-ce-00-02
            BVID:4052
            Number of ISID's:1
                6003333(Rx)
Level-1 LspID: 00db.face.0002.00-02     SeqNum: 0x0000000e      Lifetime:   995
        Chksum: 0x4247  PDU Length: 38
        Host_name: FABRIC_CORE_2
        Attributes:     IS-Type 1
TLV:135 TE IP Reachability: 1
            Metric: 1 Metric Type: Internal        Prefix Length: 32
            UP/Down Bit: FALSE             Sub TLV Bit: FALSE
            IP Address: 10.0.0.2
Level-1 LspID: 00db.face.0003.00-00     SeqNum: 0x00000018      Lifetime:   494
        Chksum: 0xd699  PDU Length: 126
        Host_name: FABRIC_CORE_3
        Attributes:     IS-Type 1
TLV:1   Area Addresses: 1
            10.01
TLV:22  Extended IS reachability:
        Adjacencies: 1
        TE Neighbors: 1
            00db.face.0002.00 (FABRIC_CORE_2)       Metric:10
                SPBM Sub TLV:
                    port id: 192 num_port 1
                    Metric: 10
TLV:129 Protocol Supported: SPBM
TLV:137 Host_name: FABRIC_CORE_3
TLV:147 Chassis MAC: 64:6a:52:a6:08:00
```

TLV:144 SUB-TLV 1   SPBM INSTANCE:

Instance: 0

bridge_pri: 0

OUI: 00-3d-b3

num of trees: 2

vid tuple : u-bit 1 m-bit 1 ect-alg 0x80c201 base vid 4051

vid tuple : u-bit 1 m-bit 1 ect-alg 0x80c202 base vid 4052

Level-1 LspID: 00db.face.0003.00-01   SeqNum: 0x00000020   Lifetime: 1121

Chksum: 0x5564 PDU Length: 99

Host_name: FABRIC_CORE_3

Attributes: IS-Type 1

TLV:144 SUB-TLV 3   ISID:

Instance: 0

Metric: 0

B-MAC: 02-3d-b3-ff-ff-ff

BVID:4051

Number of ISID's:1

16777215(None)

Instance: 0

Metric: 0

B-MAC: 02-3d-b3-ff-ff-ff

BVID:4052

Number of ISID's:1

16777215(None)

Instance: 0

Metric: 0

B-MAC: 00-db-fa-ce-00-03

BVID:4051

Number of ISID's:1

6003333(Both)

Instance: 0

Metric: 0

B-MAC: 00-db-fa-ce-00-03

BVID:4052

Number of ISID's:1

6003333(Rx)

Level-1 LspID: 00db.face.0003.00-02 SeqNum: 0x00000014 Lifetime: 488

Chksum: 0x3a47 PDU Length: 38

Host_name: FABRIC_CORE_3

Attributes: IS-Type 1

TLV:135 TE IP Reachability: 1

Metric: 1 Metric Type: Internal Prefix Length: 32

UP/Down Bit: FALSE Sub TLV Bit: FALSE

IP Address: 10.0.0.3

FABRIC_CORE_2:1#

It is worth noting that this is a display of the background work that was done by the control plane (IS-IS) with no manual configuration added as per the case in MPLS network. This solution auto-configures everything, resulting in:

- No human error as no configuration ever needs to be done on the network (Only the edge were the service connects)
- New services is deployed in a few seconds rather than weeks/months
- Failover is micro seconds.