AN ENERGY-EFFICIENT SDN/NFV FRAMEWORK FOR LOW-POWER IOT NETWORKS

by

DIPON SAHA

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
August 2020

Table of Contents

List of Tables

List of Figures

Abstract

Software-defined networking (SDN) and Network Function Virtualization (NFV) enable efficient network configuration and management in data centers and enterprises. SDN/NFV based design can also bring innovation in the wireless domain like low-power IoT networks with appropriate domain-specific protocol and architecture design. Low-power IoT devices have limited resources (e.g., power, CPU, memory) and operate in the presence of interference. Thus, in this thesis, we propose an energy-efficient interference-aware SDN/NFV framework for IoT networks. First, we formulate an Integer Linear Programming (ILP) problem to minimize the number of activated NFV nodes and the communication energy consumption. We assign IoT traffic sources to those NFVs over energy and interference-aware routes to minimize the network's overall energy consumption. Then, we develop a heuristic for large IoT networks as the proposed ILP problem is NP-complete. To facilitate the heuristic implementation, we design an SDN/NFV node architecture. We solve the ILP problem using CPLEX and evaluate the heuristic in the Cooja simulator (Contiki OS). Extensive evaluation results over two types of topologies with varying traffic load and network size reveal that the proposed solution uses almost half the communication energy compared to the state-of-the-art schemes. It also offers significantly better packet delivery ratio and network lifetime compared to its counterparts with minimal control overhead.

List of Abbreviation

APS	Assignment and Path Selector
CapEX	Capital Expenditure
CONF	Configuration Messages
COTS	Commercial Off-The-Shelf
DAG	Directed Acyclic Graph
DAO	DODAG Destination Advertisement Object
DIO	DODAG Information Object
DIS	DODAG Information Solicitation
DODAG	Destination Oriented Directed Acyclic Graph
DRS	Dynamic Route Selection
DV	Dependent Variable
ETSI	European Telecommunication Standard Institute
FD	Forwarding Devices
FTQ	Flow Table Query
FTS	Flow Table Set
GAP	Generalized Assignment Problem
IDS	Intrusion Detection System
ILP	Integer Linear Programming
INPP	In Network Packet Processing
IoT	Internet of Things
IV	Independent Variable
LLN	Low-power and Lossy Network
MANO	Management, Automation and Orchestration
MILP	Mixed Integer Linear Programming
NAT	Network Address Translator
NFV	Network Function Virtualization
NFV-CONF	NFV Configuration
NFVI	NFV Infrastructure

NI	Northbound Interface
NMM	NFV Management Module
NOS	Network Operating System
NSU	Node State Update
OF	Objective Function
OpEX	Operational Expenditure
PDR	Packet Delivery Ratio
RMM	Route Management Module
RPL	Routing Protocol for Low-power and Lossy Network
RSSI	Received Signal Strength Indicator
SDN	Software Defined Network
SI	Southbound Interface
SRH	Source Routing Header
TI	Texas Instrument
VNF	Virtualized Network Function
WSN	Wireless Sensor Network

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor, Dr. Israat Haque for the continuous support of my MCS study and research, for her patience, motivation, enthusiasm, and immense knowledge. She has taught me the methodology to carry out the research and to present the research works as clearly as possible. It was a great privilege and honor to work and study under her guidance.

Besides my supervisor, I want to thank all of my fellow lab mates of Programmable and Intelligent Network (PINet) research lab for their constructive criticisms about this work. Specially, I want to express my heartfelt thanks to one of my fellow lab mates, Meysam for his expert opinion and help to complete my research. Last of all, I want to thank Dr. Michael Baddeley for his co-operation and continuous support in the implementation.

Finally, I want to thank my wife her selfless sacrifice and support throughout my graduate studies. Without her patience, perseverance and continuous support, it would be very impossible for me to finish this work. Last but not the least, I want to thank my parents for their prayer and support throughout my whole life.

Chapter 1

Introduction

In a world of globalization, the world is getting smaller because of increased connectivity and communication technology. With the invention of the Internet of Things *(IoT)*, all devices in our home, workplace, car, and the city can be connected. IoT has become one of the key enablers of smart technologies (e.g., smart home, smart city, smart economy, smart health systems, etc.). Billions of IoT devices are being connected to the network to bring new services and meet consumer demands.

Being a member of Low-power and lossy network *(LLN)*s, the IoT devices suffer from low-power and low-computation capability and unstable communication due to interference in their radio medium [4]. Also, with the massive influx of traffic due to the explosion of IoT devices, the legacy network infrastructure falls short to process the dynamically changing network demand. Moreover, bringing new network services online and constantly configuring the legacy network devices is time-consuming and costly. Thus, in this thesis, we move to propose a dynamic solution of introducing an SDN/NFV framework with energy and interference-aware optimization and service-chaining capabilities. The solution aims to minimize the total energy consumption and the effect of interference of the entire network as well as to provide a dynamic way to configure the network in the event of any change in the network states. To this end, we first formulate the problem as an *Integer Linear Programming (ILP)* and after proving it to be an NP-complete problem, we move to develop a heuristic. To facilitate the evaluation of the solution, we also propose an SDN/NFV based node architecture. In the formulation, we consider four sets of nodes, all of which are different from each other in terms of capabilities (CPU, memory, and energy) and functionalities. Next, we relax the assumption and consider all of the relay nodes as possible candidates for NFV nodes and incorporate that in the ILP formulation. In this chapter, we first focus on the motivation and research objective of our work. Then, we point out the major contribution and how the rest of the thesis is constructed.

1.1 Motivation

Every day billions of IoT devices are getting connected to the communication highway to meet the increasing demand for smart technologies. For instance, the Cisco Annual Internet Report 2020 states that the IoT connections will represent more than half (14.6 billion) of all global connected devices and connections (28.5 billion) by 2022 [5]. The same report also mentions that the IoT applications will represent more than 6% of the worldwide IP traffic by 2022. Moreover, legacy IoT network infrastructure can not handle this massive influx of traffic due to the lack of the ability to dynamically program the network or deploy new services. To facilitate the ever-changing demand of the network, the IoT network needs a dynamic, robust, and complete network framework. We envision that Software-defined Network *(SDN)* [2] can bring the dynamism to the network, while Network Function Virtualization *(NFV)* [6] can bring the robustness in deploying new services. Together, they can offer a complete network framework to withstand future demand and bring the possibility of new cutting-edge technology.

SDN is an emerging concept that brings a change in the traditional network, where network intelligence is coupled with forwarding functions in network devices (e.g., routers, switches, etc.). It decouples the control plane from the data plane and places the control logic in a logically centralized controller to enable network programmability. With the global view of the underlying network, the controller has the ability to enforce dynamic configuration to the data plane forwarding devices to cope with the ever-changing nature of the network. It simplifies new policy introduction, manages the existing resources, and promises smooth protocol evolution [7]. Thus, SDN converts the static legacy network infrastructure into a dynamic one capable of handling any changes in the network with minimal supervision.

On the other hand, NFV decouples the network functions like routing, firewall, DNS caching, data aggregation, etc. from dedicated proprietary hardware and deploys them on-demand by enabling virtualization technology. It implements the virtualized functions on the off-the-shelf devices [8]. This virtualization technique can be used to deploy various services in the network as per the demand of the users. Several virtual network functions can be chained together to form a service-chain to provide

a series of services to the users. The chaining provides a high level automation, on-time recovery, and fault-tolerance [9]. Furthermore, the SDN can act as a platform to serve for Virtualized Network Functions (VNF) deployment, where the controller can act as the orchestrator. Thus, the IoT domain can undoubtedly be benefited by the combined SDN/NFV architecture.

However, both SDN and NFV are mainly designed for the wired network, which is why they are not well suited for meeting the IoT network's challenges. The IoT network domain posses two main obstacles: low-power and low-computation capability and unstable communication medium. Moreover, to facilitate the influx of new devices being added every day, the IoT network comes with a prerequisite of IPv6 addressing. So, it is evident that to introduce the SDN/NFV architecture in the IoT network domain. We need to consider all of these challenges and redesign the integration of SDN and NFV in both architecture and protocol. Baddeley *et al.* [10] propose, μSDN, an SDN based solution for IoT network domain with optimization in architecture, protocol, memory, and controller level to make it comparable with other existing solutions for the IoT networks. They use RPL [11] (the IPv6 Routing Protocol for Low-Power and Lossy Networks) as the base of their communication protocol, while they use IEEE 802.15.4 as the foundation for their architecture. However, μSDN lacks any optimization in energy consumption, which is one of the main challenges of IoT networks.

Galluccio *et al.* [12] propose SDN-WISE, first stateful SDN based solution for WSNs (Wireless Sensor Network). They use traditional SDN Openflow [13] protocol as the base of their communication protocol by adapting it to work in the WSN domain. However, the solution lack any adaptation of reducing the energy consumption of interference prone IoT network domain. Also, the solution requires compatibility with IPv6 addressing. Neither μSDN nor SDN-WISE offers any NFV capability. Later, Anadiotis *et al.* [14] present *SD-WISE*, which is the extended work from [12] with a provision of NFV capability, but it has lacked any optimization to manage the resources of IoT networks. It also does not provide any compatibility with IPv6 or RPL protocol. The authors in [15–19] investigated energy consumption in both SDN-based wired and wireless networks, while the authors in [20–22] investigated the communication interference in traditional wireless sensor and IoT network. The

authors in [23–28] considers different routing approaches in SDN based wireless and mobile adhoc networks. However, to the best of our knowledge, no work has presented an energy-efficient and interference-aware SDN/NFV framework for IoT network with both IPv6 and RPL compatibility.

1.2 Research Objective

To mitigate the IoT network's challenges, we focus on two main aspects: reducing energy consumption and eliminating or curbing the effect of interference on the transmission of data. In our initial investigation [29], we have found that introducing energy-based routing decisions with SDN/NFV based architecture certainly improves the resource utilization of IoT networks. Moreover, by integrating the interference-aware factors in that routing decision will undoubtedly reduce the re-transmission of packets as both the data and control traffic share the same channel to transmit, reducing the control-overhead increases the available bandwidth, which eventually helps the application data to be transmitted more efficiently and improves the resource utilization. Furthermore, we can adopt *data aggregation* [30, 31] as a virtual network function that can help reduce the energy consumption and interference of the network. Lastly, with the reconfigurability feature of SDN, we can tackle any sudden network state change (e.g., node or link failure), which is very common in an IoT network. Thus, we move to propose a novel SDN/NFV framework, which:

- Minimizes the communication energy consumption.
- Mitigates the interference of the communication medium.
- Reduces the control overhead.
- Offers the provision for NFV capability as well as service-chaining.
- Offers energy-efficient and interference-aware communication protocol.
- Can reconfigure the network based on network state change.

1.3 Contribution

In this thesis, we propose an SDN/NFV framework for IoT networks based on IEEE 802.15.4 and compatible with IPv6 and RPL. We first formulate an ILP model that activates the minimum number of NFV enabled IoT devices (nodes) and assign the IoT data sources to those activated NFV nodes over energy-efficient and interference-aware routes. To this end, in our implementation, we use data aggregation as the virtual network function available in the network. However, any standard network functions can be used in this case. We then prove that our proposed ILP model is NP-complete [32], which means the proposed model is not feasible to implement for a large network. To overcome this issue, we design a heuristic. To cope with the dynamic nature (e.g., any node goes out of operational energy or interference in network medium) of the IoT network [26, 33], we also design an algorithm called DRS, which will control and manage the heuristic. Lastly, we consider all of the relay nodes as possible candidates for the NFV nodes, incorporate that into the optimization model, and update the corresponding heuristic to make the framework more general.

To solve our proposed models, we use CPLEX solver and describe its properties. Then, we implement and evaluate our proposed framework (*EA-SDN/NFV*) using *Cooja* simulator, which is based on the *Contiki* OS [34]. We compare the performance of EA-SDN/NFV with its variant (SDN/NFV) with no optimization, μSDN, and SDN-Wise, where the latter two present SDN based solutions. We evaluate with both grid and random topology with varying route length, traffic loads, and network size. The evaluation results show that EA-SDN/NFV performs significantly better than other comparing schemes. Then, we analyze the control-overhead of all four comparing schemes where our proposed solutions EA-SDN/NFV shows comparable results to μSDN but significantly better than SDN-WISE. Moreover, we show the reconfigurability feature where the SDN controller uses the DRS algorithm to invoke on-demand route reconfiguration and NFV nodes activation to offer better network lifetime and PDR. Lastly, we show the performance of our proposed solution with or without considering separate NFV enabled nodes. The performance of the former proves better in terms of communication, energy consumption, and PDR. In summary, all of the contributions are listed below.

- We design an SDN/NFV node architecture with VNF service-chain provisioning for resource-constraint and interference-prone IoT network. We also develop the corresponding communication protocol for its smooth operation [29].
- We formulate an ILP model that minimizes the number of activated NFV enabled nodes and assigns the source nodes to them over energy-efficient and interference-aware routes [4].
- We prove that our proposed ILP model is NP-complete [4].
- We design a heuristic to facilitate a large IoT network [4].
- We perform an extensive evaluation of our proposed SDN/NFV framework on the Cooja simulator and Contiki OS. The evaluation results reveal that EA-SDN/NFV improves 1.39x and 1.6x communication energy consumption compared to μ SDN and SDN-WISE, respectively. It furthermore offers 1.36x, and 1.58x better packet delivery ratio (PDR) compared to μSDN and SDN-WISE, respectively. The proposed scheme also outperforms its counterparts in network lifetime, control-overhead, and reconfigurability due to network state change [4].
- We generalize our solution by considering all of the relay nodes as possible candidates for the NFV nodes. We evaluate and compare our solution with and without considering separate NFV enabled nodes, which reveals that without separate NFV enabled nodes, our solution performs slightly better.
- Finally, we provide a detailed implementation of our proposed solution in [35].

1.4 Thesis Outline

The rest of this thesis is organized as follows: Chapter 2 presents two sections: Section 2.1 introduces the necessary background to understand this thesis, and in Section 2.2, we discuss relevant research works. Chapter 3 presents the design and methodology of the thesis. It contains four sections: Section 3.1 describes the research methodology, while in Section 3.2, we define our problem and formulate the corresponding ILP model. In Section 3.3, we design the heuristic, while in Section 3.4, we present our SDN/NFV node architecture and corresponding communication protocol

to implement the heuristic. Chapter 4 presents two parts: the first part (Section 4.1) presents necessary setup and description of parameters for the evaluation, while the second part (Section 4.2) presents the findings. Chapter 5 presents the generalization of our proposed solution where we consider all of the relay nodes having NFV capabilities. Lastly, the thesis concludes in Chapter 6.1 with future research directions.

Chapter 2

Background and Related Work

In this chapter, we first address necessary background on some topics to better understand this thesis. Then, we review existing literature related to our work.

2.1 Background

2.1.1 Internet of Things (IoT)

The *Internet of Things (IoT)* has been a driving force behind the realization of smart technologies. From the smart thermostat to smart light, all the devices around us that are connected to the internet and make our life easier are part of the IoT. Technically, all the "things" in the Internet of Things mean any object connected to the internet. However, nowadays, it is considered that any object that can "talk" to each other is an IoT device [36]. So, in a nutshell, any device that can collect data, transmit them over the internet, and act upon them when directed is called the internet of things. In a broader sense, IoT is a concept or paradigm that considers a variety of objects present in our surroundings and through wired or wireless connection coupled with unique addressing schemes can interact with each other to attain a common objective [37].

IoT ecosystem has three parts (Fig.2.1): Collect data through sensing, transfer the data, and analyze it. The fundamental characteristics of IoT are summarized below [37]:

- In the context of IoT, anything with the ability to communicate can be interconnected providing connectivity and accessibility to the network.
- IoT provides objects or things related to services such as temperature monitoring, privacy protection, etc.
- The devices in IoT are heterogeneous based on service platforms and networks.

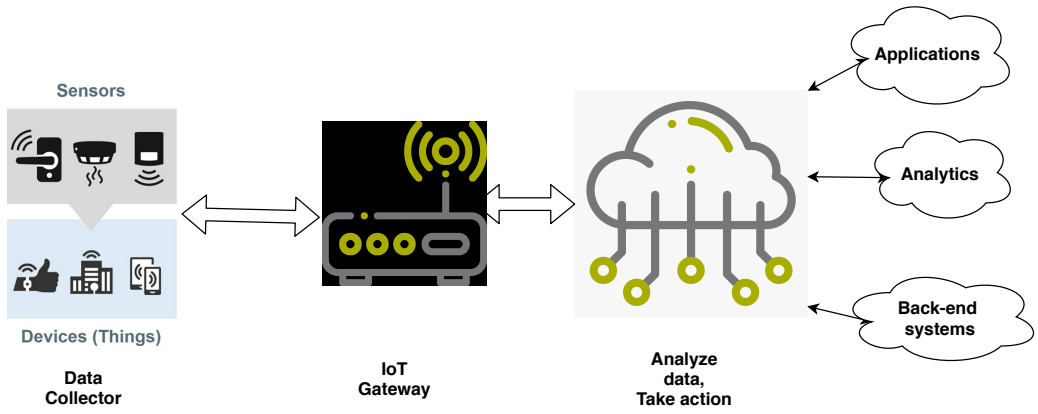


Figure 2.1: An example of IoT ecosystem [1].

- The state of the devices in IoT dynamically changes.
- The number of devices connected to the internet is enormous. Also, the amount of generated data is huge. Managing this data is critical.
- Connectivity and accessibility to the network.

Along with the promises to a future networking era, the IoT network domain also presents a set of challenges. The main two challenges an IoT network domain presents are resource constraints (such as low-power, low-computation, low-storage, etc.) and interference due to the instability of the communication medium. Among other challenges, interoperability, security, data management, and cost vs. usability are noteworthy [37].

2.1.2 Overview of RPL

Routing Protocol for Low power and lossy network (RPL) is an IPv6 compatible routing protocol for the family of *Low power and Lossy Network (LLN)*. IETF develops it as a standard routing protocol for this family of network devices. RPL is a proactive distance vector-based protocol, which helps to reduce the memory requirement at the network nodes to carry out the routing operation. RPL uses a tree-like topology called *Directed Acyclic Graph (DAG)*. Each node is associated with a parent that acts as a gateway of that node. The parents are calculated using *Objective Functions (OF)* dictated by the roots of the DAG [38].

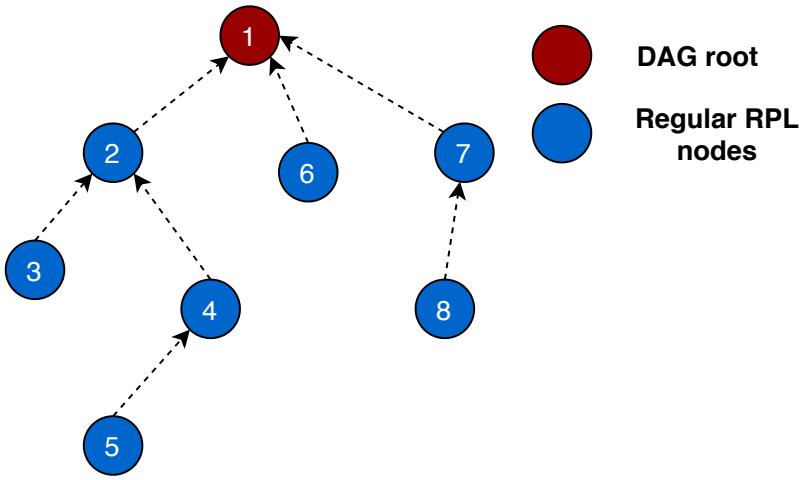


Figure 2.2: An example of RPL topology.

The topology information is kept in a *Destination Oriented Directed Acyclic Graph (DODAG)* and maintains routes towards the destination or sink node. RPL uses three main types of control messages:

- DODAG Information Object (DIO),
- DODAG Information Solicitation (DIS) and
- DODAG Destination Advertisement Object (DAO).

The DODAG is constructed using two control messages DIO and DIS. The root provides the DAG information using the DIO message. DIO message can be initiated by root or in response to the sensor nodes' DIS control message. The DIO message is then re-broadcast through the entire network until every node joins the DAG root. The DIO message contains an OF and the routing metrics (e.g., link quality, residual energy). Each node uses OF to select a preferred parent to reach the root. Each node also calculates a rank based on their relative positions, which helps both choose the parents and avoid any loop formation. This way, an upward route is constructed for each node. Now, to announce their presence to the DAG root, each sensor nodes must send control message DAO to the root through its parent. Thus, all the nodes in the DAO path get updated with the necessary information to reach the sensor nodes below it, and eventually, a downward route is constructed [39].

For instance, let us consider an example. Let us assume we have a representative topology (Fig.2.2). At first, the network administrator configures one or more nodes

as DODAG root (node 1 in Fig.2.2). The root then starts sending the multicast DIO message to advertise its presence towards the regular RPL nodes. The nearby nodes (nodes 2, 6, and 7 in Fig.2.2) will receive the DIO message and process it as it comes from a lower rank node. They will select the root as their designated parent. Then, the DIO message will be re-transmitted as a local link multicast message and eventually reach all other regular RPL nodes (nodes 3, 4, 5, and 8 in Fig.2.2). An RPL node may receive a DIO message from multiple nodes. It then selects its preferred parent based on the OF embedded in the DIO messages. The process will continue until all the regular RPL nodes join the DODAG tree. Once done, the upward route has already been constructed. For downward route construction, the regular RPL nodes start sending DAO messages towards the root through their respective parents. Once DAO messages reach the root, all the RPL regular nodes and root become aware of the nodes below them.

The RPL offers two modes of operation: *Storing* and *Non-storing*. In the storing mode, the topology is constructed using RPL control messages and the routing table is maintained at each node to facilitate the forwarding of the data packets. On the other hand, in the non-storing mode, the routing table is maintained at the root after construction and each node uses source-routing to deliver the data packets to its destination. The later mode resembles the controller based communication of SDN architecture and we use the non-storing based communication for our proposed solution.

2.1.3 Software-Defined Network (SDN)

The Software-defined Network has brought a paradigm shift in networking technologies in the last decade. It provides an alternative to the legacy network design, which is burdened with managing today's dynamic nature [7]. Legacy networking is dominated by the network components and appliances manufactured by the commercial vendors coupling with proprietary network function and control structure to configure them according to pre-determined network policies [40]. The network operators need to painfully configure individual devices of a network using the vendor-specific control instructions to reflect a certain network policy. Moreover, the legacy network has no reconfiguration and response mechanism when it is very much prone to the dynamics

of network failures and its state changes [7]. Also, the vertical integration of network devices creates a hindrance to innovation, which results in leaving the world with old technology to deal with new network demand where it is incapable of handling. So, in a nutshell, the limitations of legacy networks are [7]:

- Proprietary configuration mechanisms of vendor specifics network appliances.
- Configuration complexity to reflect specific network policies.
- Almost no response mechanisms for network changes.
- Hindrance to innovation.
- Lack of flexibility to accommodate new technology like IPv6 right away.
- Increase in operational and capital expenditure.

SDN is a concept of technology where it breaks the vertical integration of the legacy network by separating the control logic from the network devices like routers, switches, etc. By the separation of the control plane from the data plane, it makes those devices into a simple forwarding device. The control logic is placed in a logically centralized entity or controller, which simplifies the network configuration and response to network changes. This separation can only be realized by a well-defined protocol and API (e.g., Openflow [13]) between the data and control plane.

A brief overview of the SDN architecture is presented in Fig.2.3. The architecture has three layers (planes) and two APIs. They are:

1. **Forwarding devices (FD) and Data plane:** The forwarding devices are interconnected with each other by radio channel or wired connection to form the data plane. The FDs perform some basic operations (e.g., forward packets) based on the instruction set (e.g., flow rules) set by the controller. For example, OpenFlow switches.

2. **Southbound Interface (SI):** The instruction set of the FDs and the communication protocol between the data and control plane are defined by the SI. For example, Openflow [13] is a standard SI.

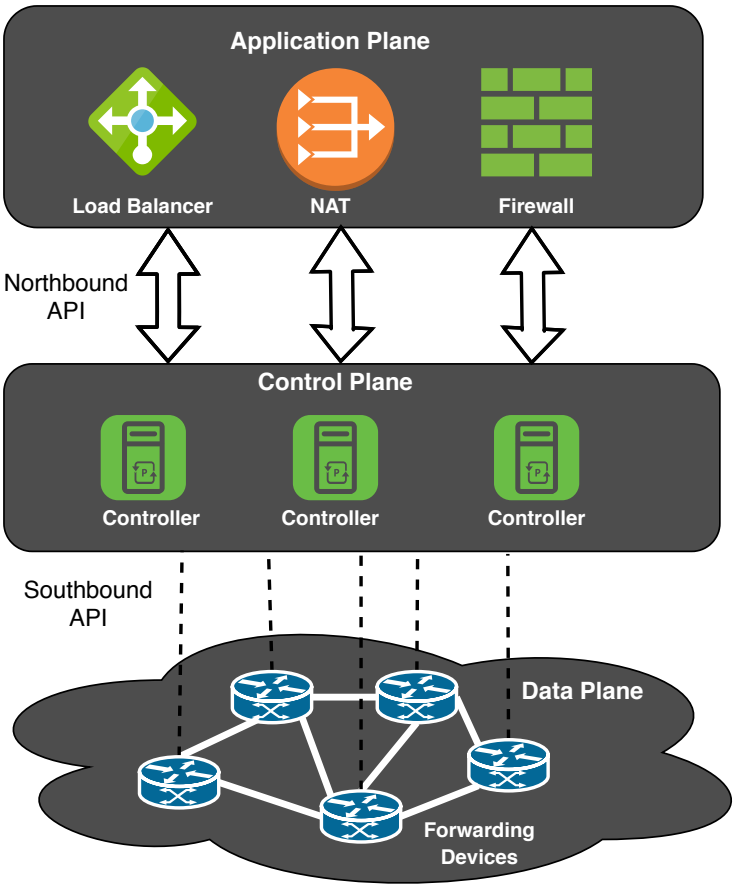


Figure 2.3: A simplified SDN architecture [2].

3. **Control Plane:** The brain of the SDN based architecture is the control plane. It consists of one or more logically centralized controllers or *Network Operating Systems (NOS)*. The controller can configure the network according to the network policy using the global knowledge of the underlying network. It creates the instruction set of the FDs that dictates their behavior in the network.

4. **Northbound Interface (NI):** The controller or NOS offers an API for the application plane to contact.

5. **Application Plane:** This plane holds applications that control the network operation and determine the network policy. The applications use the NI's functionalities to assert the necessary policy based on the demand of SLA.

SDN can essentially provide the programmability that enables the network to automatically respond to various network changes. Being a member of LLNs, the IoT

network domain can capitalize on the features provided by the SDN based architecture to fight off its unique challenges like resource constraints and communication instability.

2.1.4 Network Function Virtualization (NFV)

NFV is a concept of network architecture that decouples the network functions from their corresponding specialized hardware, making them more of a modular building block that can be deployed or chained to create a network service. The need to separate the network functions from the network devices arises when it becomes more complex to use new services into the network as network demand increases. In the legacy network, there are many middle-boxes closely associated with one or more related network services. In particular, the middleboxes are network entities or devices that are capable of transmitting, transforming, filtering, inspects, or control network traffic to manage the network better [41]. For example, NATs, firewalls, IDSs are one of the most commonly used middleboxes. Due to cutting edge technology, diverse demand of the users, and the necessity to cope with dynamic network changes, the need to deploy more of these middle-boxes has increased to many folds. For instance, almost every network needs middle-boxes like firewalls to protect their virtual environment. Also, we want some NAT middle-boxes to tackle the diminishing effects of the address spaces. The deployment of those middle-boxes not only is burdened with so many complexities but also increases the *Capital Expenditures (CapEX)* and *Operational Expenditures (OpEX)* [41].

To alleviate those issues with legacy networks, NFV coupled with *Commercial Off-The-Shelf (COTS)* presents a great solution. The COTS are providing general-purpose hardware appliances rather than specifically built hardware appliances, and building network functions as software entities and deploying them in those COTS brings flexibility, high capacity with less cost and simplified deployment strategy [6]. Technically, all the network functions offered by the legacy network devices can be virtualized and termed as *Virtualized Network Function (VNF)*. The VNFs can be instantiated and deployed in any entities of *NFV Infrastructure (NFVI)*. The VNFs can be chained together to form specialized network services to provide required and diverse user demand. The benefits of NFV are summarized below:

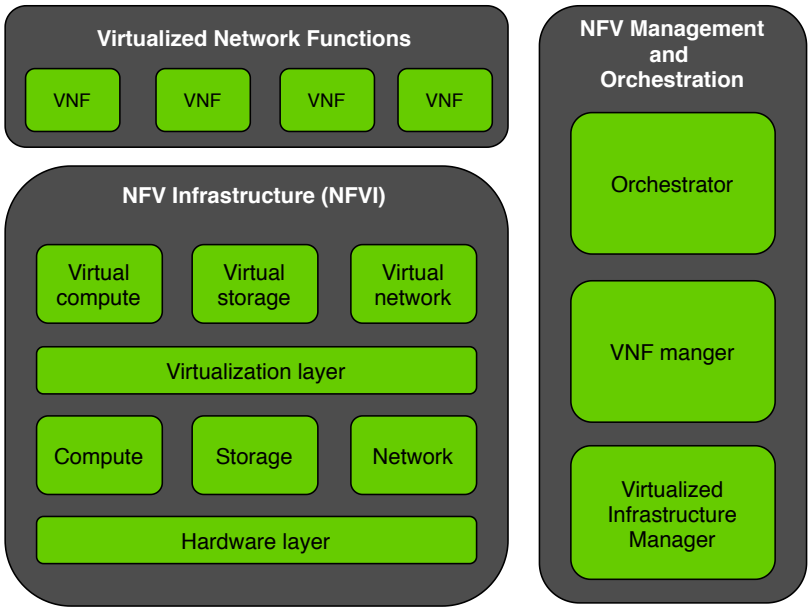


Figure 2.4: A simplified NFV architecture by ETSI [3].

- Flexibility in introducing new network services.
- Vendor independent deployment with COTS.
- High scalability.
- Reduction of time to deploy new services to the network.
- Reduction in CapEX and OpEX.
- High performance and management of the network by allocating resources on-demand.

The world's largest telecom operators like AT&T, British Telecom, ISG, Deutsche came together to form *European Telecommunication Standard Institute (ETSI)* to build a standard architecture for NFV. The proposed architecture of ETSI is presented in a simplified form in Fig.2.4. The architecture has three main components. They are:

1. **Virtual Network Function (VNF):** The legacy network function is virtualized into software entities capable of all the functionalities of their hardware counterparts. The virtualized software entities are called VNFs. This layer holds all the available VNFs in the network like a container.

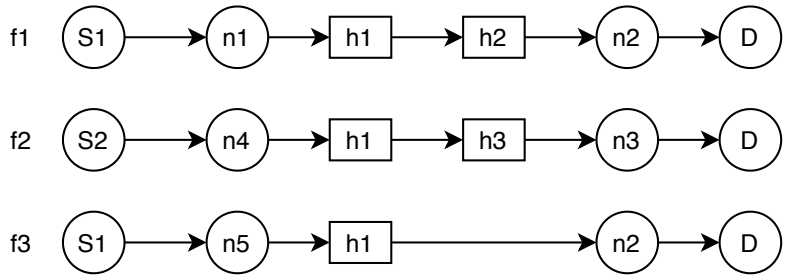


Figure 2.5: An example to illustrate service-chaining.

2. **NFV Infrastructure:** This layer provides the components to execute the operations of the VNFs. It contains the physical compute, storage, network components, and virtual counterpart through a virtualization layer.

3. **Management, Automation, and Orchestration (MANO):** This layer is mainly responsible for managing and orchestrating the deployment of the VNFs into the NFVI layer. Based on the functionality, its component can again be divided into three parts. They are:

    - **NFV Orchestration:** This part does both the service and resource orchestration. The deployment, resource allocation, service chaining, etc. are managed by this part. It acts as a liaison between the VNF layer and the NFVI layer.
    - **VNF manager:** It manages VNF instantiating and the life cycle of the VNF instances.
    - **Virtualized Infrastructure Manager:** It controls and manages the NVF infrastructures, both physical and virtual aspects of computing, storage, and network.

Given the service-oriented nature of IoT, NFV can help to better manage and deploy the services to the network to meet the user requirement.

2.1.5 NFV Service-chaining

Service-chaining [9] is a concept of creating a chain of network functions that are deployed through the network. Traditionally, an NFV service chain is an ordered list of VNFs (such as data aggregation, firewalls, network address translation (NAT),

intrusion protection, etc.) that are chained together to form a service for corresponding traffic flows as per the network policy [42]. The main advantage of the service chaining is to achieve a higher degree of automation in setting up a virtual connection between available VNFs in the network to serve corresponding traffic flows. The service-chaining also helps in on-demand resource allocation, recovery, fault detection, etc. [42]. To shed more light on the concept of service chaining, we illustrate an example in Fig. 2.5. We define a set of service chains $F = f_1, f_2, f_3, \ldots, f_n$, where each service chain consists of one or more VNFs at the network. Let us assume, we have VNFs h_1, h_2 and h_3 available at the network. The service chain f_1 consists of h_1 and h_2, while service chain f_2 consists of h_1 and h_3. Any traffic flow can request to be serviced by any chain, $f \in F$. Lets say, h_1 and h_2 represents the firewall and load-balancer network function. If any particular traffic flow requires the service of a firewall and then of a load-balancer, then it would choose the service chain f_1.

The VNFs, participating in a service chain, are placed in the network based upon the type of network and its application. They can be consolidated in a single physical node or can be distributed throughout the network based on the network policy and available resources. For instance, if we have adequate resources and want to reduce in-service latency, we might want to consolidate all the VNFs of a service chain in a single node. However, doing so might increase the traffic on physical links and nodes, creating network instability. On the other hand, if we go through the distributed deployment, then the individual nodes' resource consumption will be reduced at the cost of latency. In particular, a low-power and lossy IoT network requires low resource utilization and communication overhead, where the latter is more dominant. Thus, we consider all VNFs from a service-chain consolidated in a single IoT node. Moreover, in our proposed design, we design a hierarchical architecture that employs multiple NFV nodes to host all the VNFs in a service chain to provide services to the assigned source nodes. This design reduces the chance of clogging one single NFV node.

2.1.6 SDN/NFV Architectures

The SDN provides better control and management of the network while bringing the programmability to the network. On the other hand, Network Function Virtualization (NFV) decouples network functions (e.g., NAT, IDS, Firewall, etc.) from the

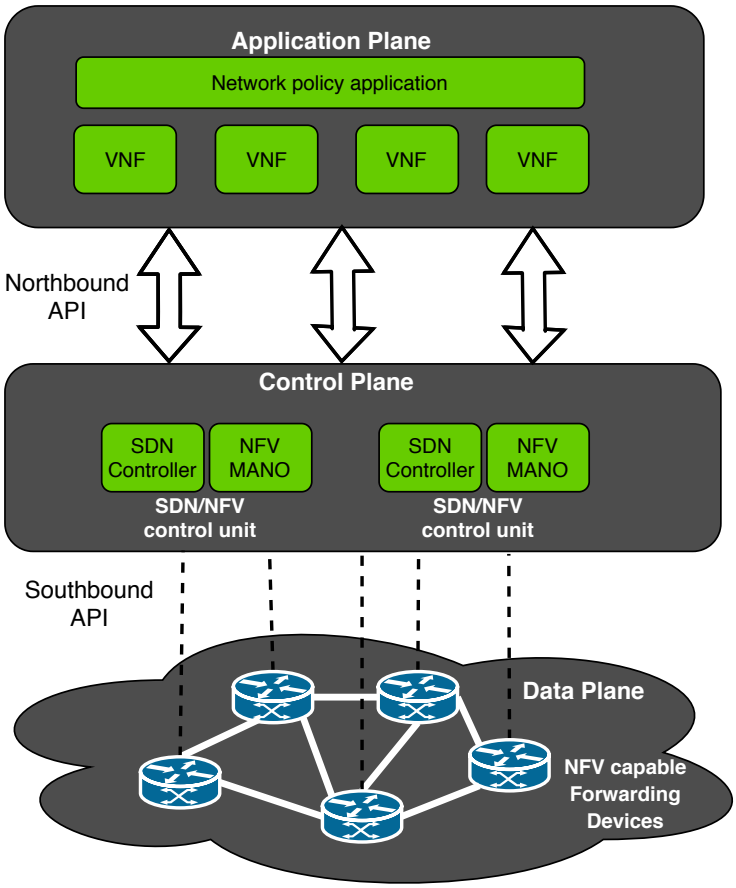


Figure 2.6: A simplified SDN/NFV architecture.

proprietary hardware. Coupled with COTS general purposed devices, the virtualized network functions can be deployed with limited difficulties. It reduces the time to deploy the new network functions as well as CapEX and OpEX. SDN and NFV are two distinct technologies that share some common elements in their implementation. The main and the most striking similarities are that both of these technologies need a control structure to manage and control their operation.

Being two complementary technologies, SDN and NFV can work together to provide one true network solution. SDN can provide dynamic management and connectivity between the network elements, NFV can provide the technology to deploy different network functions as a complete network service to the users. The controller of SDN can work as the replacement of the MANO of the NFV architecture. While SDN can constitute, the NFVI enabled forwarding devices with the capability of containing the deployed VNFs. The corresponding SDN/NFV architecture is presented

in the Fig.2.6. The architecture has three parts. The application layer holds the corresponding network policy application and VNF definition. The VNF layer of the NFV architecture (Fig.2.4) is mapped in this layer. The control layer contains the SDN controller or NOS and the NFV MANO. Together they build the control unit of the SDN/NFV architecture. The NFVI layer of NFV architecture is mapped into the data plane, where the forwarding devices that form this plane must be SDN enabled with NFV capability. The NI of SDN architecture dictates the communication between the application plane and the control plane, while SI dictates communication between control and data plane. The protocol used for these two interfaces: NI and SI, is dependent upon the application, for which the architecture would be deployed.

Being a member of LLNs, SDN/NFV architecture can help the IoT network domain to tackle its resource constraints and communication instability. SDN part can help monitor the scarce resources of the IoT network and dynamically reconfigure the network to maintain the constant operation. On the other hand, we can dynamically deploy the required services to meet user demand.

2.1.7 Data Aggregation Technique

Data aggregation is a process of collecting the data, combining or compressing it, and efficiently delivering to its final destination [43]. Billions of IoT devices are deployed as part of the smart technology, which ultimately brings a tremendous surge in data traffic. Data aggregation can reduce the high volume of traffic, which consequently reduces the number of transmissions. This also helps to reduce the bandwidth demand, energy consumption, and the chance of interference. Being a member of the family of LLNs, the IoT network is one of the application domains of various data aggregation techniques.

There are various data aggregation techniques available for wireless sensor networks. Mainly, the data aggregation techniques can be divided into two phases: *Data Collections* and *Data Compression*. The network architecture has a notable impact o the data collection. Based on the network architecture [43], data aggregation technique can be two types, *Flat model* and *Hierarchical model*. In the flat model, only the sink node is the data aggregator, and all the sensor nodes send the data to the sink node. The network delay and overhead are high in this case. On the contrary, in

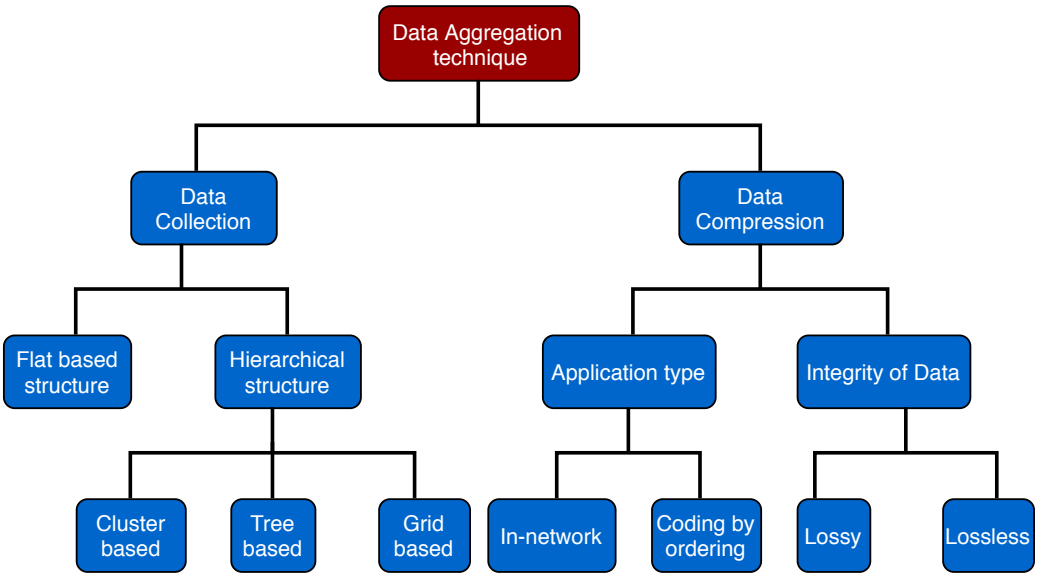


Figure 2.7: The data aggregation techniques at a glance.

the hierarchical model, the sensor nodes and the sink node are placed at the bottom and top of the hierarchy, respectively. The data collector or aggregator nodes and the relay nodes can be in the middle. Based on how to select the aggregator nodes, the hierarchical model again can be categorized into three types:

- *Cluster-based:* the cluster heads are selected as the aggregator nodes where all the data will be combined or compressed. The cluster-based approach has average overhead, low energy uniformity, average strength and flexibility, scalability, and low power consumption.

- *Tree-based:* aggregation is done through the creation of a data aggregation tree, which can be a minimum spanning tree. Tree-based methods have high overhead and energy uniformity, strength, flexibility, scalability, and energy consumption is the average.

- *Grid-based:* nodes of the network are arranged in a grid. Sensors within a particular grid send data directly to the integrator of the grid. It is a special kind of cluster-based method where the cluster-head is fixed in an array of nodes or grids. This method is useful, especially for military surveillance, weather forecast, and dynamic change in the network conditions.

There are several ways to compress the data. Based on the type of application, the type of compression [44] can be: *in-network aggregation* and *coding by ordering*. The in-network aggregation collects data in buffers and combines them by reducing redundancy. In the coding by ordering, similar data are removed and encoded using the removal information. The latter method is constrained by available resources. Also, based on the integrity of the retrieved data, there are two types of data compression: *loseless* that needs high computation and *lossy* that is computationally less heavy [45]. The techniques that are discussed here are illustrated in Fig.2.7. All of these techniques above have different utility. For a resource-constrained network like IoT network, we consider computationally light and energy-efficient in-network hierarchical aggregation, where an aggregator compresses received data as an average before sending it to the sink.

2.1.8 Interference Detection Technique

One of the main challenges of the IoT network is its instability of communication. Due to the scarcity of the radio spectrum, wireless communication faces interference very often. The network interference is a phenomenon when unwanted signals in the same channel accumulate and hamper ongoing communication [46]. Usually, due to scarce radio resources, when multiple radio devices use the same radio frequency to transmit simultaneously, interference occurs. It causes packet loss, their re-transmissions, link instability, and inconsistent protocol behavior [46]. There are various ways to detect interference at the network. Those techniques can be divided into two main categories.

- Exact modeling of Interference.
- Sampling the network data.

Sampling network data is one of the most prominent methods of detecting interference. There are various ways to sample network data like custom probing and calculation based on Signal-to-Noise Ratio [47], using a machine learning approach to study the *Received Signal Strength Indicator*(RSSI) and packet loss [48], threshold calculation through RSSI value [22, 49], etc. The RSSI threshold estimation is a widely used technique because of its availability in the off-the-shelf sensors [50]. The threshold value in this approach represents the 'noise floor', which can be estimated in

advance by measuring the RSSI value of an idle channel. The experimental value [22] and the published threshold value by Chipcon (Texas Instruments, 2006) both correspond to -45 dbm. We use this threshold in our proposed solution to estimate the interference.

2.2 Related Work

The main challenges of the IoT network domain are two folds. Resource constraints (energy, memory, CPU) and the instability of its communication medium due to interference. To tackle these issues, we propose an energy-efficient and interference-aware SDN/NFV based framework in this work. Moreover, we consider the data aggregation as the deployed VNF at the NFV nodes. We also make provisions for service chaining in our design. In this section, we present existing works related to our proposed framework. We thoroughly review the literature related to our work and point out the gaps in the present state-of-the-art solutions.

2.2.1 SDN/NFV Based Designs

μSDN [10] is an SDN framework based on Contiki OS and has its protocol stack built on IEEE 802.15.4. It has added interoperability with underlying protocol and IPv6. The proposed work brings several optimizations to minimize the overhead of SDN based architecture. The main optimizations are in four core areas: SDN *protocol*, *architecture*, *flow tables*, and *controller*. In protocol optimization, it eliminates the fragmentation by reducing the packet size and reduces the packet transmission frequency. Among other architecture level optimization, the use of source-routing is the most effective one. It optimizes memory by re-using the flow table match-actions. For instance, it reduces the repeated entries for the same forwarding actions. In the controller level, it uses an embedded custom controller, which eliminated controller communication overhead between the sink node and external controller.

The node architecture of μSDN contains four main modules. They are μSDN protocol, controller adapter, SDN engine, and SDN driver. μSDN uses its lightweight protocol to communicate with the controller. It is based on UDP, which enables it to use DTLS for secure communication. The controller adapter presents an abstraction to the controller communication. It allows μSDN protocol to be replaced with any

other protocol if necessary. SDN engine handles all the incoming and outgoing messages. The main protocol logic is implemented in this stage. SDN driver provides the necessary API to handle flow-table operations.

μSDN also provides the three basic operations: *controller discovery and configuration*, *node status update*, and *routing*. The controller discovery phase depends on RPL control messages. After receiving DAO messages from IoT nodes, the controller sends CONF messages to configure those nodes for the corresponding operations. Each node periodically updates the controller with its state information such as energy-reading, link quality, list of active neighbors, etc. using NSU control message. After initialization, the source nodes transmit data packets towards the corresponding destination following flow-table entries in a multi-hop scenario. However, the intermediate relay nodes use source-routing to forward the packets towards their destination. With heavy optimization and RPL compatibility, μSDN offers a promising solution to the SDN based architecture in the IoT network domain. But the work lacks any NFV provisioning, routing optimization related to energy consumption or interference.

Galluccio *et al.* [12] propose SDN-WISE, the first stateful SDN based solution for wireless sensor networks. In the SDN-WISE node architecture, there are three data structures embedded. The *WISE states array*, *accepted IDs array*, and *WISE flow table*. The WISE states array is a data structure that contains the state information of the WISE node. Given the broadcast nature of WSN, the accepted IDs array allows each sensor node to select only the packets, which it must further process. In particular, the accepted ID is specified in the header of the packet. If the carried ID is matched with the accepted ID array, the packet is to be processed further; otherwise, it is to be dropped. The WISE flow table contains the matching rule and corresponding action. The matching rule can be any part of the packet header or any state of the nodes. If no rules are matched against the packet, a controller request is sent for further instruction.

The SDN-WISE operation has 4 components: *forwarding, INPP, topology discovery*, and *adaptation* layer. The forwarding layer handles arriving packets as specified in a WISE flow table. The tables are continuously updated by this layer, as dictated by the configurations sent by the controller. The INPP layer runs above the forwarding layer and handles all in-network packet processing, like aggregating small

packets. The topology discovery layer does all the functionality of topology discovery protocols. It periodically collects local network state information and sends them towards the controller. The adaptation layer works as an adapter between the sink node and WISE-Visor, which ultimately works as a platform for controllers. Being a first stateful solution for WSN, the proposed solution provides a complete design though it does not provide any resource optimization, NFV compatibility or compatibility with IPv6 or RPL.

Anadiotis *et al.* [14] propose *SD-WISE*, which is the extended work of [12]. They introduce the NFV capability in the SDN-WISE architecture and show its effectiveness by deploying geographic routing as network service. SDSense [26] is also an SDN based architecture for WSN that focuses on decomposing the network function based on its agility and scope. In particular, it focuses on separating static or slow-changing phenomena (e.g., topology control) from dynamic or fast-changing phenomena (e.g., congestion control). It also provides a resource allocation scheme as well as reliability through the construction of edge-disjoint topology.

Tomovic *et al.* [51] propose an SDN based architecture in the IoT network domain that uses network virtualization to manage heterogeneous IoT network domains. Their primary focus lies in delivering an intent-based network solution, where the network configuration rules of the underlying network are inferred based on users' intent. However, the provisioning of NFV is used for management purposes. Ojo *et al.* [52] propose an SDN based architecture for IoT network domain with NFV provisioning. They focus on business values and market drives. They provide no implementation and comparison of the evaluation.

Wang *et al.* [53] propose an SDN based data-centric routing protocol, MINIFLOW. The routing mechanism considers three factors: hop distance, residual energy, and RSSI value of the links. They optimize the routing and make the routes energy and interference-aware. Farhan *et al.* [21] propose a routing approach for a traditional wireless network that selects a routing node with lesser neighbors and thus less interference. It also provides a new clustering algorithm for selecting cluster heads with shorter transmission distance. Ding *et al.* [20] design an interference and energy-aware routing algorithm for software-defined WSNs. In [54], the authors, upon investigating the challenges of future IoT domain, put forward an SDN/NFV based

Approach	Domain	IPv6 & RPL Compatibility	Energy & Interference Optimization	NFV
μSDN [10]	IoT	✓	×	×
SDN-WISE [12]	WSN	×	×	×
SD-WISE [14]	WSN	×	×	✓
SDSense [26]	WSN	×	×	×
QoS aware [51]	IoT	×	×	✓
Architecture based [52]	IoT	×	×	✓
Heuristic Routing [53]	WSN	×	✓	×
Routing [21]	WSN	×	×	×
Routing [20]	WSN	×	×	×
Design aspects [54]	IoT	×	×	✓
Architecture based [55]	IoT	×	×	✓

Table 2.1: A comparison of features for SDN based works.

framework design. But they lack in any implementation or comparison with state-of-the-art solutions. Sinh *et al.* [55] propose an SDN/NFV architecture to deploy IoT devices. Their main focus is on building applications to slice end-to-end multiple network segments for deploying IoT services.

None of the works described above has NFV capability except [14, 51, 52, 54, 55]. Also, only [53] considers designing energy and interference-aware routing without formulating any optimization problem. Finally, none of these works present an SDN/NFV based solutions for the IoT domain with IPv6 and RPL compatibility. The summarized comparison of the above solutions for wireless networks is presented in Table.2.1.

2.2.2 Energy Optimization

The authors in [15–19, 23, 25, 56] propose different approaches to optimize energy consumption both in wired and wireless environment. The authors in [15] optimize the energy consumption by limiting the activation of switch and links to carry traffic. They also formulate a MILP model along with a greedy heuristic to solve the scalability issue. The authors in [16] focus on improving memory usage while optimizing energy consumption. Their model considers capacity constraints of link and rule space constraints on routers. In [17], the authors propose a traffic-engineering approach to attain energy-efficiency. In summary, the approaches in [15–17] propose SDN based

optimization model to minimize the energy consumption of the network, but they all focus on traditional wired networks. Whereas the authors in [18] propose EDAL, an Energy-efficient Delay-aware Lifetime-balancing data collection protocol to optimize energy consumption in a wireless environment, but there is no SDN/NFV support. The approach in [19] focuses on VNF deployment in a multi-domain SDN environment while optimizing the resources. Though the work has NFV provisioning and energy optimization, they focus on the traditional wired network domain. The authors in [23,56] optimize energy consumption in the wireless environment, but they do not possess any SDN/NFV support or RPL and IPv6 compatibility. In [24,27,28,57], the authors provide solutions regarding different non-SDN based energy-aware routing approaches based on the placement or location of each node in the mobile ad-hoc network. The authors in [58] proposes solutions for the expected path length using distance-based or angle-based based greedy routing schemes in ad-hoc wireless networks. The authors in [59] propose ROEE RPL, a resource-oriented and energy-efficient RPL protocol for IoT domain with IPv6 compatibility. But they do not provide any SDN/NFV support. Also, they do not consider the network interference.

All of the above works consider energy optimization in different domains. Except for [28], none offers any optimization to mitigate the interference. Also, they do not provide a complete solution that offers an SDN/NFV based design with energy and interference optimization, provision for service chaining, and compatibility with RPL and IPv6 for the IoT networks. A summarized comparison of energy-optimization based work in different domains is illustrated in Table 2.2.

In summary, none of the existing works has considered providing a complete SDN based framework for IoT networks with the compatibility with IPv6 or RPL, energy and interference awareness, and the provision for dynamic deployment of network services through NFV. In this work, to address this gap, we consider all of the above-mentioned constraints simultaneously and propose an ILP problem, a corresponding heuristic, and an SDN/NFV framework with the provision for service chaining to implement the heuristic.

Approach	Domain	SDN	NFV	Interference	IPv6 & RPL
Dynamic Traffic [15]	Wired	✓	×	×	×
Rule Placement [16]	Wired	✓	×	×	×
Traffic Engineering [17]	Wired	✓	×	×	×
EDAL [18]	WSN	×	×	×	×
Resource optimization [19]	Wired	✓	✓	×	×
OLEAR [23]	WSN	×	×	×	×
Reliable topology [56]	WSN	×	×	×	×
Energy-aware routing [24]	Mobile adhoc	×	×	×	×
Localized routing [27]	Mobile adhoc	×	×	×	×
Localized routing [28]	Mobile adhoc	×	×	✓	×
ROEE RPL [59]	IoT	×	×	×	✓

Table 2.2: A comparison of features for energy optimization based literature.

Chapter 3

Design and Methodology

3.1 Research Methodology

In this section, at first, we present the definition of our problem and corresponding research hypothesis. Then, we move to decompose the hypothesis to identify its key components and analyze their relationship. At last, we provide a design outline of the solution of the defined problem.

3.1.1 Problem Definition

SDN and NFV are two emerging technologies in the present world. On the other hand, IoT has become the driving technology to realize smart technology. With billions of new devices coming online, the IoT network infrastructure needs restructuring to cope with the explosion of data traffic. SDN/NFV based architecture can bring those much-needed changes to the IoT domain. However, the adoption must be aware of the two main challenges; resource constraints (energy, CPU, and memory) and interference due to the nature of the communication medium. The question is if the energy and interference-aware SDN/NFV architecture (EA-SDN/NFV) can perform better than any SDN/NFV architecture without optimization (SDN/NFV) or other existing architectures like μSDN, SDN-WISE, etc. Thus, the testable hypothesis becomes Hypothesis 1.

Hypothesis 1. *Energy and interference-aware protocol in SDN/NFV architecture for IoT network can perform (network lifetime, energy consumption, and packet delivery ratio) better than other existing protocols like an SDN/NFV architecture without any optimization, μSDN, and SDN-WISE.*

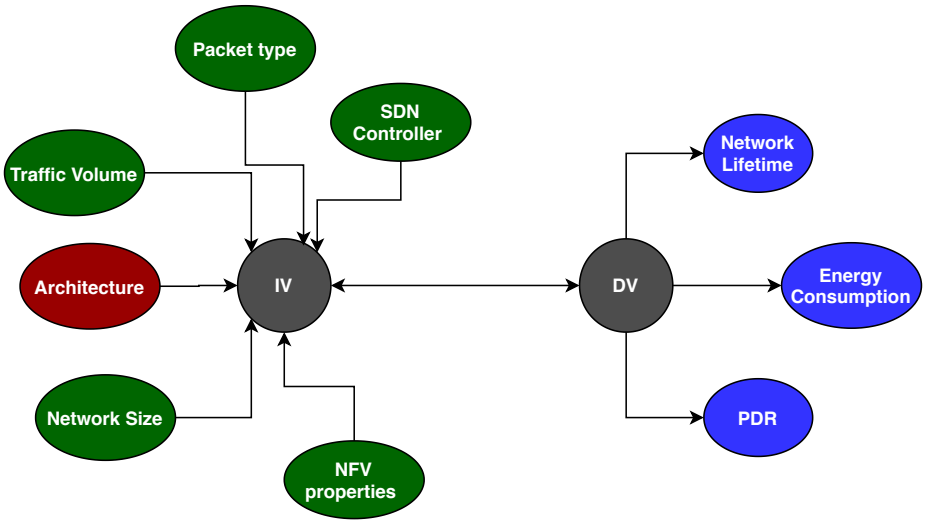


Figure 3.1: A decomposition of dependent and independent variables.

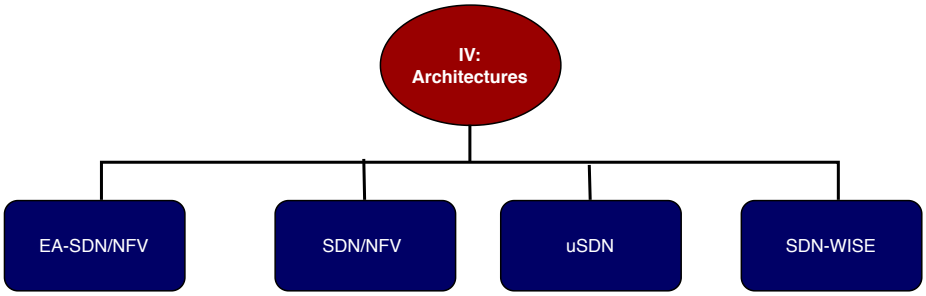


Figure 3.2: Levels of Independent variable.

3.1.2 Decomposition of Variables for Hypothesis 1

In research design, the ultimate goal is to achieve a true experiment where one can observe the result of the manipulation of one variable on another. In particular, the goal is the observance of a cause-effect relationship. The variables presented in the Hypothesis 1 have two main categories: *Independent Variable (IV)* and *Dependent Variables (DV)*. The decomposition of different variables are illustrated in 3.1.

3.1.2.1 Independent Variable (IV)

It is the variable that can be changed, manipulated, or modified to observe, monitor, or measure the response of its subjects on other variables [60]. In the Hypothesis 1, the main IV is the architecture. Because our main goal is to observe the response of the IoT network for different architecture in terms of its performance. The architecture

variable has 4 levels (Fig. 3.2) such as EA-SDN/NFV, SDN/NFV, μSDN, and SDN-WISE.

Ideally, only the choice of the independent variable, i.e., the architecture, influences the network's performance. However, pragmatically the network performance gets influenced by other variables like the size of network topology, type of traffic, volume of traffic, etc. Also, the placement of the SDN controller and the choice of NFV functionalities available at the network can affect the performance. These variables are not directly related to the hypothesis but can influence the relationship between the independent and dependent variables. Based on the characteristics of the variables that directly or indirectly influence the relationship between IV and DV, we can categorize the other variables into two categories: control and measurement variable.

Control Variable: the variables that influence the IV-DV relationship but remain constant, where we observe only one of its instances is called the control variable [60]. Here, from Hypothesis. 1, it is clear that our main IV is the architectures. But to maintain *internal validity*, i.e., to keep the outcome (performance) of the research solely dependent on the main IV, we need to promote the type of traffic, choice of SDN controller placement constant. Also, the available NFV functionalities at the network must be kept constant for those architectures with NFV capabilities.

Measurement: To maintain *external validity*, i.e., the extent to which the outcome of the research can be applied to a more generalized audience or situations [60], we need to consider how the architectures would behave under different network conditions. Based on our application domain (IoT domain), the variation in scalability (size of topology) and traffic load can most affect performance. We plan to measure the performance under different states (topology size and traffic load) of the network.

3.1.2.2 Dependent Variable (DV)

The variables that are observed or measured to inspect the influence of the IV are called dependent variables. In our proposed testable hypothesis, the performance metrics are the dependent variables. The performance is to be measured in terms of network lifetime, energy consumption, and packet delivery ratio (PDR).

Operational Definition: The dependent variable has three levels. They are:

- *Network lifetime* of the IoT domain can be measured by the residual energy after a certain time, where each IoT device starts with a fixed amount of initial energy.

- *Energy consumption* of the IoT network can be measured using the total communication (transmit and receive) energy spent by the IoT devices within a certain time where each IoT device has the same transmission and reception power.

- *Packet Delivery Ratio (PDR)* is the ratio of the total packet received at the destination and the total packet transmitted from the source.

Quantification of DV: To measure the DV, we need to design a systemic, reliable, and replicable way. In research methodology, the scales are tools to distinguish the individual data on the variables as to how they differ from one another. there are four scales of measurement: *nominal, ordinal, interval,* and *ratio* [60]. The degree of sophistication increases as we move from nominal to a ratio scale. With a higher degree, we can perform more complex analysis and get more detailed information. The nominal scale is used to distinguish data into categories or groups. The ordinal scale is used to distinguish data not only in categories but also it can differentiate data within a category by ranking them. The interval scale helps us to determine the distance between two data points on a scale. It only concerns the magnitude of difference, while it can have random origins. The ratio scale overcomes this obstacle and has a fixed start point. It is the most powerful scale of all. All three levels of DV present data at this ratio scale. They all have a meaningful interval between two measuring points and the absolute zero point for the origins. For example, the energy consumption unit is millijoule (mj), which has a fixed interval between two data points and a meaningful absolute zero origin. The recommended statistical measurement for a central tendency for ratio type measurement is arithmetic or geometric mean with standard deviation or variance is the recommended measures of dispersion of data.

3.1.2.3 Research Design

From the above discussion, it is certain that we want to evaluate the influence of the EA-SDN/NFV architecture and compare the results with that of other existing

architecture like SDN/NFV architecture without any optimization, μSDN, and SDN-WISE. In designing the EA-SDN/NFV, there are few obstacles. First, our goal is to design a system that can tackle the main challenges of the IoT network domain, i.e., the resource constraints and the interference due to the communication medium. Moreover, it is to introduce the SDN/NFV architecture to the IoT network domain in order to bring programmability. One of the key decisions in implementing an SDN/NFV solution is how many NFV nodes need to be activated, which is dictated by the purpose of reducing energy consumption and interference of the communication medium. Also, how the source nodes are going to get serviced by the NFV nodes is another important factor. So, designing an SDN/NFV solution for IoT network poses some interesting questions.

- *RQ.1* How many NFV nodes should we activate to meet the traffic demand?
- *RQ.2* How should we manage the traffic to reduce the scarce resources of the IoT network domain?
- *RQ.3* How can we increase the lifetime of the network?

Moreover, both SDN and NFV are designed for the wired environment. To adapt SDN/NFV architecture for the IoT environment, we need to optimize the protocol, memory, controller communication, and architecture level. The solution to those questions may bring trade-offs. For example, if we activate a small number of NFV nodes to reduce resource consumption, the total traffic demand may not be fulfilled. Again, we can not use the same routes to relay the traffic as it would depreciate the battery levels of the more used nodes much faster, which in turn reduces the network lifetime. We need to come up with a way to distribute the load while keeping the resource consumption minimum. Moreover, to reduce the communication energy consumption, we need to construct the routes as the shortest route, but that may not be interference aware. So, the solutions available to those questions are full of choices and trade-offs. That is why we need to define an optimization problem considering all of these constraints. Then, we need to determine its feasibility to implement it in a real-life scenario. If not, then we would design a heuristic. To realize the optimal or near-optimal solution, we need to come up with a complete SDN/NFV architecture,

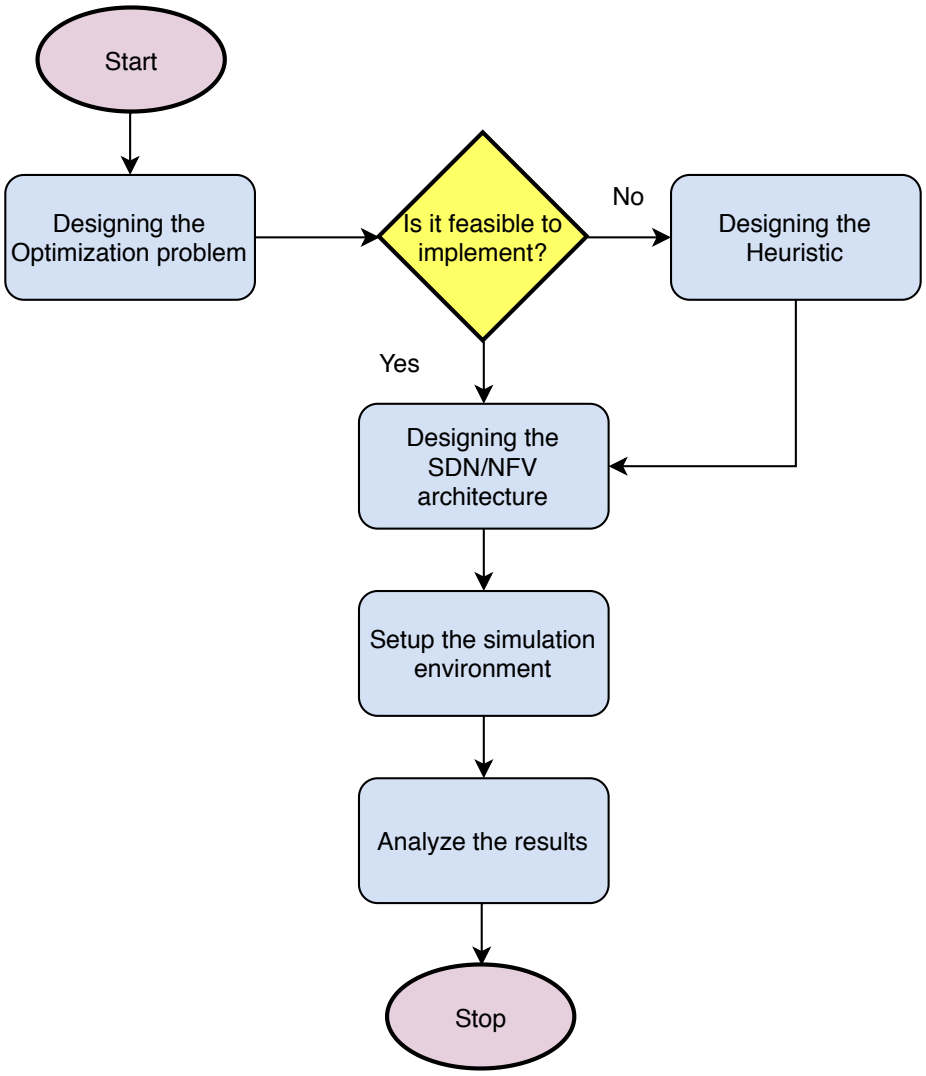


Figure 3.3: Steps of research design.

through which we can simulate solutions in an emulated environment. In Fig. 3.3, the total research designs are illustrated.

3.2 Problem Formulation

Our goal is to find an energy-efficient and interference-aware protocol and architecture for SDN/NFV based IoT networks. In this section, we try to find answers to the questions that have presented in Subsection 3.1.2.3. It is evident from our earlier discussion that we need to offer a comprehensive solution to answer those research questions. First, we study the requirements to define an optimization problem and

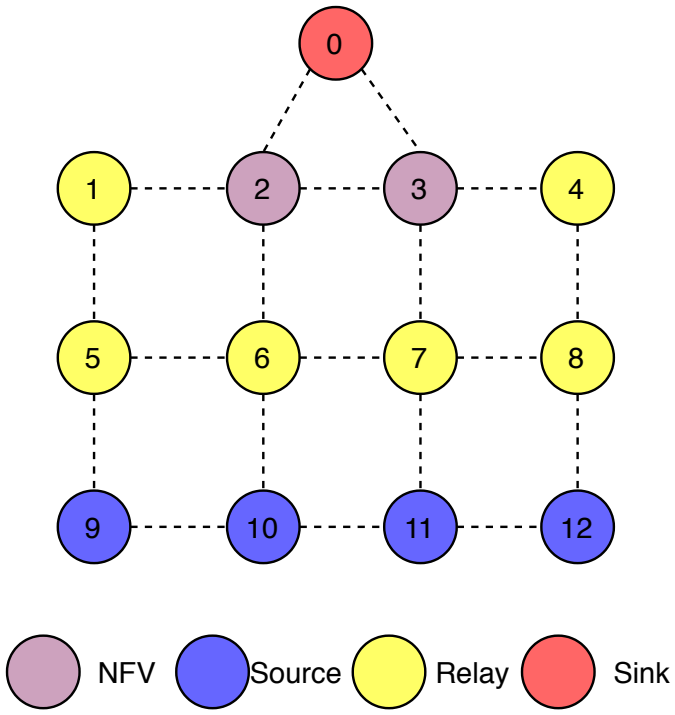


Figure 3.4: An example for ILP problem formulation.

then define the problem as an Integer Linear Programming (ILP) formulation. Furthermore, we investigate the feasibility of implementing the formulation and prove that the proposed ILP formulation is NP-complete. Please note that the formulation, corresponding proof, heuristic, and the architecture design is presented in [4, 29].

Requirement analysis: Let us consider the representative network illustrated in Fig. 3.4. We have 4 types of nodes here: NFV, source, relay, and sink node. Each of them has different capabilities in terms of both resource availability and functionality. Our goal is to activate a minimum number of NFV nodes from the set of 2 NFV nodes, i.e., 2 and 3. Also, we need to assign all source nodes to the activated NFV node-set in such a way that the total communication energy consumption of the network, as well as the probability of interference occurrence, are minimized. Let us assume that both NFV nodes (node 2,3) are activated to meet the demand of all the sources. So, each of the source nodes (nodes 9, 10, 11, and 12) must be assigned to either of the activated NFV nodes. The selection of assignment for a source node is based on the most energy-efficient and interference-aware path from the source node to the sink node via the assigned activated NFV node. Also, the assigned NFV node to a source node must also maintain its capacity based on the resources (CPU, memory and

energy) available in it and meet the demand of the source nodes, i.e., if a source node needs to be serviced by certain network functions, that functions must be present at the assigned NFV node. All of the choices are dependent on each other and need to be considered at the same time.

So, while realizing our goal we need to define an ILP formulation. Moreover, in our problem definition, we consider service chaining where a source node can demand services from multiple Virtualized Network Functions (VNF) on its way towards the destination. In order to come with the ILP formulation, we need to place all the consideration in the form of constraints such as, *assignment constraints*, *capacity constraints*, *threshold constraints*, *supply-demand constraints*, *flow conservation constraints*, etc. The objective of those constraints are discussed below:

- **Assignment constraints:** These constraints need to be placed in the formulation so that all the source nodes are assigned to the activated NFV nodes. Also, no source nodes are permitted to be assigned to multiple NFV nodes.

- **Capacity Constraints:** This type of constraint are there to ensure that no NFV nodes are to be assigned if there are not enough resources (CPU, memory, energy) to provide services to the source nodes.

- **Threshold constraints:** To keep the solution to the formulation energy-efficient and interference-aware, we need to place a threshold to both energy and link quality, respectively. We are not to select any node for the assigned path if it has energy below the energy threshold. Similarly, no link is selected for the path because the link quality is below the threshold. Because with low link quality, the probability of interference occurrence increases.

- **Supply-demand Constraints:** The NFV nodes must host the services that are to be requested by the assigned source nodes to those NFV nodes.

- **Flow-conservation Constraints:** To build the path, we need to define these constraints. It directly builds the path connecting source nodes to the sink node via an assigned NFV node.

Network Modeling: For the definition, we can translate the representative network topology into a graph $G = (V, L)$ where nodes $V = \{1, 2, 3, \ldots, n\}$ represents

the IoT devices in the network and L the links that connect them. It is clear that to illustrate the problem we need four types of network nodes: source nodes $S \subset V$, relay nodes ($R \subset V$), NFV nodes ($N \subset V$) and finally, the sink node ($S_0 \in V$), where $(S \cap N \cap R) = \emptyset$ and $(S \cup N \cup R) = V - S_0$. All the source nodes S generate the traffic where the relay nodes R relay them towards the sink node S_0 via the NFV nodes N.

Definition of parameters: For each NFV node $n \in N$, the cost of activation depends on the how much resource it is going to use and also the amount of energy associated with the processing of hosted VNFs and communication (transmission and reception) overhead. So, we define the cost of activating an NFV node, c_n, as the summation of energy associated with resource utilization (CPU and Memory), $E_{utility}$, and the processing and communication energy, E_{pc}. The c_n is expressed in (3.1).

$$c_n = E_{utility} + E_{pc} \tag{3.1}$$

We define the capacity, CP_n, of each NFV node $n \in N$ as the maximum number of source nodes it can serve based on the available resource and energy at the NFV node n. We also define the budget, B, of the whole network, which refers to the maximum number of NFV nodes that can be activated. The budget is determined by the resources of all the NFV nodes available at the network and the total demand (number) of the source nodes. In (3.2), we define the budget B of the entire IoT network in terms of average capacity of the NFV nodes, $\sum_{n \in N} CP_n \,/\, |N|$ and the number of source nodes $|S|$.

$$\lceil \frac{|S|}{\sum_{n \in N} CP_n \,/\, |N|} \rceil \leq B \tag{3.2}$$

The residual energy, e_j of each node $j \in V$ refers to the amount of energy each node j has left. We also denote the minimum residual energy threshold as E_j for each node $j \in V$. For the interference mitigation, we consider the RSSI value to be a good indicator of link quality. Based upon the RSSI value we can design an estimator to estimate the probability of interference on a link [22, 49]. We denote r_{ij} as the RSSI value of any link $(i, j) \in L$, while the minimum RSSI threshold is denoted by R_{ij}.

We also consider a set of service-chains $F = f_1, f_2, f_3, \ldots, f_n$ where each service-chain $f \in F$ consists of one or more VNFs available at the network. To reduce

Table 3.1: The list of notations used in the formulation.

Notation	**Description**
a_n	Decision variable to activate an NFV node n.
B	Budget for activating the NFV node
c_n	Cost of activating a NFV node n.
CP_n	Capacity of an NFV node n.
D_{fs}	Any demand of node s for service f
E_j	Energy threshold for a node j.
e_j	Energy level of a node j
F	Set of service chains available
f	Any functions in the service chain set F
(i, j)	A link $(i, j) \in L$.
L	Set of all links.
N	Set of NFV nodes.
R	Set of relay nodes.
R_{ij}	RSSI threshold value of a link (i, j)
r_{ij}	RSSI value of a link (i, j)
S	Set of source nodes.
S_0	Sink node.
S_{fn}	Availability of service f at node n
V	Set of all nodes.
$w1, w2$	Normalization or scaling factor.
x_{sn}	Decision variable for the assignment of a node s to node n.
y_{ijsn}	Decision variable for a link (i, j) selection for the assignment of a node s to node n.
z_{ijsn}	Combined decision variable for assignment of a node s to node n and link selection (i, j).

resource utilization and the communication overhead, we consider the VNFs of the corresponding service-chain are consolidated at one NFV node. However, based upon the resource available at the NFV node, it can host multiple instances of service-chains. For instance, in our proposed solution we consider the service-chains of data aggregation VNF, and one NFV node can host multiple instances of this service chain

based upon the availability of its resources. For each source node, $s \in S$ the demand for the services of a service chain $f \in F$ is denoted by D_{fs}. On the other hand, the availability of a service chain $f \in F$ at an NFV node $n \in N$ is denoted by S_{fn}. A summary of all the notations used in the problem formulation is illustrated in Table.3.1.

Objective Function: Our objective is to activate a minimum number of NFV nodes and assign them to the source nodes over energy-efficient and interference-aware routes. In order to translate the objective into an ILP formulation, we define three decision variables. The first one is for the activated NFV nodes, second is for the assignment of source nodes to the NFV nodes, and the third is for the selection of routes for the assignment. The decision variables are presented below.

$$a_n = \begin{cases} 1, & \text{if } n \text{ is activated} \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

$$x_{sn} = \begin{cases} 1, & \text{if } s \text{ is assigned to } n \\ 0, & \text{otherwise} \end{cases} \tag{3.4}$$

$$y_{ijsn} = \begin{cases} 1, & \text{if } (i,j) \text{ is selected for assignment of } s \text{ to } n \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

The objective function is presented in (3.6), which has two parts. The first part takes care of the minimization of activation of NFV nodes and the second part concerns about assigning the source nodes to the activated NFV nodes while constructing the energy-efficient routes towards the sink node. The construction of the routes again depends upon the shortest hop distance while preserving maximum residual energy. We use two normalization factors: w_1 and w_2 to scale down the corresponding values and make them comparable. w_1 is the normalization factor related to the activation cost and w_2 is related to energy cost, which makes all the values scaled so that they can be comparable to each other. In this instance, they work as simple tuning parameters.

$$\min \quad \sum_{n \in N} w_1 \, c_n \, a_n + \sum_{n \in N} \sum_{s \in S} \sum_{i,j \in L} x_{sn} \, y_{ijsn} \, (1 - w_2 \, e_j) \tag{3.6}$$

List of Constraints: The objective function in (3.6) is subject to some constraints. The constraints are described one by one. The constraints from (3.7) to (3.9) take care of the activation and assignment of NFV nodes. In particular, (3.7) ensures that at least one NFV node is to be activated. As to provide services to the source nodes, we need to activate an NFV node to the minimum. The constraints in question ensure that we have that. On the other hand, constraints in (3.8) make sure that each source node is to be assigned to exactly one NFV node. This is because if one source node is assigned to multiple NFV nodes, then it promotes to activate more NFV nodes, which contradicts our goal. It is to be noted that the VNFs participating in a service chain are consolidated in a single NFV node, and only if the requirement of the services needed by the source node matches the availability of the services at the NFV node, only then a successful assignment is done. So, assigning multiple NFV nodes to a source node is redundant. Constraints in (3.9) confirms that no source nodes should be assigned to non-activated NFV nodes. If an NFV node is not selected for activation, then that NFV node is not a suitable candidate for the assignment. To be assigned to a source node, an NFV node must be activated. These sets of constraints set the foundation of activation and assignment of NFV nodes to the source nodes.

$$\text{s.t.} \qquad \sum_{n \in N} a_n \geq 1 \tag{3.7}$$

$$\sum_{n \in N} x_{sn} = 1, \qquad \forall s \in S \tag{3.8}$$

$$x_{sn} \leq a_n, \quad \forall s \in S, \forall n \in N \tag{3.9}$$

The next set of constraints is related to the capacity and available residual energy of the NFV nodes. Constraints in (3.10) ensures that the total assigned source nodes to an NFV node must not exceed the capacity of that NFV node. Recall that the

capacity of the NFV node is associated with the available resources (CPU, memory, energy). If the NFV node does not have the necessary resources or capacity to serve, no source node can be served at that NFV node if assigned. The constraints related to the budget of the network are described in (3.11). Here, it is ensured that the total number of activated NFV nodes must not exceed the budget. This is to make sure that we do not activate NFV nodes more than what we need. The constraint in (3.12) makes sure that no NFV nodes are to be activated if the total activation cost exceeds its residual energy. If there is not enough energy to accommodate the total activation cost, an NFV node can not be activated and assigned to the source nodes. Because then it will not be able to process any incoming packets from the source nodes due to the lack of energy.

$$\sum_{s \in S} x_{sn} \ \leq \ CP_n, \quad \forall n \in N \tag{3.10}$$

$$\sum_{n \in N} a_n \ \leq \ B \tag{3.11}$$

$$c_n a_n \ \leq \ e_n, \quad \forall n \in N \tag{3.12}$$

Service-chain based constraints are expressed in equations from (3.13) to (3.15). In particular, constraints in (3.13) expresses that each source node must demand the service of at least one service chain, while constraints in (3.14) expresses that each NFV node must have the availability of at least one service chain. These two sets of constraints maintain the supply-demand of the service chains. To complete the assignment and activation procedure, each source node must have demand, while the NFV nodes must supply the service chains. The constraint (3.15) ensures that a source node must be assigned to an NFV node, which can meet the demand of the source nodes to be serviced by a specific service-chain.

$$\sum_{f \in F} D_{fs} \geq 1, \quad \forall s \in S \tag{3.13}$$

$$\sum_{f \in F} S_{fn} \geq 1, \quad \forall n \in N \tag{3.14}$$

$$x_{sn} D_{fs} \leq S_{fn}, \quad \forall f \in F, \forall s \in S, \forall n \in N \tag{3.15}$$

The next two constraints offer the restriction concerning energy and the RSSI threshold. In particular, (3.16) provides the constraints so that no links are to be selected if the destination node does not have enough residual energy. It means that the solution will always choose links so that the destination nodes have enough energy to operate on the incoming packet. Similarly, constraints in (3.17) make sure that no link is to be selected with lower RSSI value than the RSSI threshold. This makes sure the constructed routes are interference-aware.

$$y_{ijsn}\, e_j \geq E_j, \quad \forall (i,j) \in L, \forall s \in S, \forall n \in N \tag{3.16}$$

$$y_{ijsn}\, r_{ij} \geq R_j, \quad \forall (i,j) \in L, \forall s \in S, \forall n \in N \tag{3.17}$$

The next set of constraints from (3.18) to (3.23) are directly related to the construction of the routes. Together, they are called flow conservation constraints. The route from the source to the sink node must be via an NFV node. That is why the route construction is done in two parts. The first part takes care of the construction of the route from the source to the assigned NFV node, and the second part takes care of the routes from the assigned NFV node to the sink node. In particular, (3.18) to (3.19) are directly related to the route construction of the first part whereas (3.20) to (3.21) takes care of the route construction of the second part. Constraints in (3.18) ensure that at the beginning of the route construction, the links going out from the source nodes. It makes sure that there is only one outgoing flow from the source node towards the NFV node it is assigned to. The constraints in (3.19) make sure that the incoming flow to an NFV node must be equal to the number of source nodes assigned to it. Together, these two constraints help to build the beginning and end

of the routes in the first part of the route construction. Similarly, (3.20) ensures the links outgoing from NFV nodes towards the sink node must be equal to the assigned source nodes to the NFV node. It constructs the beginning of the second part of the route construction. On the other hand, (3.21) ensures the links coming towards the sink node. It makes sure that the total incoming links from the NFV nodes must equal to the number of source nodes. These two constraints take care of the last part of the route construction. Finally, (3.22) takes care of all the incoming and outgoing links of the relay nodes. This constraint connects the source-NFV and NFV-sink pairs. All the relay nodes fall under these constraints. All incoming and outgoing links in a relay node must be the same. To complete the set, (3.23) ensures that the constructed path has no cycle. The (3.6) along with constraints from (3.7) to (3.23) completes our formulation.

$$\sum_{\substack{j\in(S\cup R)\\(i,j)\in L\\i=s}} y_{ijsn} = x_{sn}, \quad \forall s \in S, \forall n \in N \tag{3.18}$$

$$\sum_{\substack{i\in(S\cup R)\\(i,j)\in L\\j=n}} y_{ijsn} = x_{sn}, \quad \forall s \in S, \forall n \in N \tag{3.19}$$

$$\sum_{s\in S} \sum_{\substack{(i,j)\in L\\j\in(S_o\cup R)\\i=n}} y_{ijsn} = \sum_{s\in S} x_{sn}, \quad \forall n \in N \tag{3.20}$$

$$\sum_{s\in S} \sum_{\substack{(i,j)\in L\\i\in(N\cup R)\\j=S_o}} y_{ijsn} = x_{sn}, \quad \forall s \in S \forall n \in N \tag{3.21}$$

$$\sum_{(j,i)\in L} y_{jisn} = \sum_{(i,j)\in L} y_{ijsn}, \quad \forall s \in S, \forall n \in N, \forall j \in R \tag{3.22}$$

$$\sum_{\substack{j\in V\\(i,j)\in L}} y_{ijsn} \leq 1, \quad \forall i \in V, \forall s \in S, n \in N \tag{3.23}$$

Summary of formulation: The objective function presented in (3.6) has a term, which is the product of two decision variables (x_{sn} and y_{ijsn}), which makes the

objective function a non-linear one. The product term ensures the selection of a link associated with a particular assignment. The solution to a non-linear formulation is a very complex and time consuming process. Also, our goal is to produce an ILP formulation. To realize that we take steps to replace that product term with a new decision variable $z_{ijsn} \in [1, 0]$ where it is equal to 1 when the product of x_{sn} and y_{ijsn} is equal to 1 and otherwise 0. The new objective function then looks like as in (3.24).

$$\min \quad \sum_{n \in N} w_1 c_n \, a_n + \sum_{n \in N} \sum_{s \in S} \sum_{i,j \in L} z_{ijsn} \, (1 - w_2 e_j) \tag{3.24}$$

To satisfy the relationship $z_{ijsn} = x_{sn} \times y_{ijsn}$ and make it stable we introduce new constraints from (3.25) to (3.27). In particular, the constraints in (3.25) and (3.26) ensures that the new decision variable must be lower than both of x_{sn} and y_{ijsn} as both of them are binary decision variables. So the product can only be 1 or 0. To satisfy the relation and provide the stability we propose these two constraints. Lastly, (3.27) makes sure the integrity of the replacement holds.

$$\text{s.t.} \quad \textit{Constraints } (3.7) \textit{ to } (3.23)$$

$$z_{ijsn} \leq x_{sn} \quad \forall (i,j) \in L, \forall s \in S, \forall n \in N \tag{3.25}$$

$$z_{ijsn} \leq y_{ijsn} \quad \forall (i,j) \in L, \forall s \in S, \forall n \in N \tag{3.26}$$

$$z_{ijsn} \geq x_{sn} + y_{ijsn} - 1 \; \forall (i,j) \in L, \forall s \in S, \forall n \in N \tag{3.27}$$

Feasibility analysis: The proposed ILP model provides us the optimal result. However, the model is an NP-complete problem, which makes it very complex to implement in a larger scale. To proof the NP-completeness we consider two well-known class of problems: Generalized Assignment Problem (GAP) [61] and energy-aware routing [62]. We systematically prove that the two sub-problems of the proposed ILP model are instances of GAP and energy-aware routing and thus proving the NP-completeness of our proposed problem. In the following, we provide the definitions of the two sub-problems of our proposed ILP to construct our proof.

Definition 1 (NFV Assignment Problem)**.** *Let us consider a network with V nodes consists of a set of NFV nodes, $N = 1, 2, 3, \ldots, i$, where $N \subset V$ and a set of sources,*

$S = 1, 2, 3, \ldots, j$, where $S \subset V$. The NFV assignment problem assigns each $s \in S$ to exactly one $n \in N$ in such a way that the capacity of an NFV node CP_n does not exceed and the total assignment cost is minimized.

Definition 2 (Energy-Aware Route Construction)**.** *Let us consider a network with $G(V, L)$ nodes, which consists of a set of sources, $S = 1, 2, 3, \ldots, j$, where $S \subset V$ and a set of destinations, $D = 1, 2, 3, \ldots, j$, where $D \subset V$. The energy-aware route construction problem finds a route from a node $s \in S$ to a node $d \in D$ such that the energy threshold E_j of a node does not exceed and the total communication energy consumption is minimized.*

Theorem 1. *The assignment of sources to NFV nodes and the construction of corresponding energy-aware routes is an NP-complete problem.*

Proof. At first, we facilitate the definition and formulation of GAP and then we map it to the NFV assignment problem. Let $A = 1, 2, 3, \ldots, n$ and $J = 1, 2, 3, \ldots, m$ be a set of agents and jobs, respectively. c_{ij} is the cost of assigning a job $j \in J$ to an agent $a \in A$. Let r_{aj} be the resource requirement of agent a to perform job j and b_a be the available resources at the agent a. Then, the GAP problem can be defined in (3.28), where $x_{aj} \in 1, 0$ is a decision variable such that if an agent a performs a job j it is 1, otherwise 0.

$$\min \qquad \sum_{a \in A} \sum_{j \in J} c_{aj} x_{aj} \tag{3.28}$$

The above GAP problem is subject to the constraint in (3.29), which ensures that the required resource is available at the assigned agent to perform a job.

$$\sum_{j \in J} r_{aj} x_{aj} \leq b_a, \quad \forall a \in A \tag{3.29}$$

Now, let us consider an instance of GAP, where the capacity, CP_a, of an agent a is defined by the maximum number of jobs it can serve. Then we can express CP_a in terms of resource availability b_a and resource requirement r_{aj}, i.e., $CP_a = \frac{b_a}{r_{aj}}$. Thus, the problem becomes assigning jobs to agents such that the capacity CP_a of an agent does not exceed, and the total assignment cost c_{ij} is minimized. It is evident that NFV assignment problem is a special instance of GAP.

Similarly, an energy-aware routing problem is an NP-complete problem [63, 64], where routes are selected, while the total communication energy consumption is minimized and the energy threshold of nodes does not exceed. Also, the energy-aware route selection problem is a special case of a well-known network optimization problem: minimum cost routing [65], minimum concave flow problem [66], and minimum-edge routing [62]. These problems are part of the class of constrained-path optimization problems, which are also NP-complete [32, 63]. Thus, the assignment of sources to NFV nodes and the construction of corresponding energy-aware routes is an NP-complete problem. □

3.3 Heuristic Design

As our proposed ILP formulation is an NP-complete problem, it becomes a very complex and resource-intensive implementation on a larger scale. To solve this issue, we move to develop a greedy heuristic [67] that can achieve our goal and be solvable in a large network setup.

3.3.1 Design Steps

The main objective of the proposed ILP model is to minimize the NFV activation while assigning the source nodes to them over energy-efficient and interference-aware routes. The steps of our proposed heuristic are as follows:

- **Step 1:** For a source-NFV node pair, we calculate the best shortest routes based on the energy-cost and link quality (RSSI value).
- **Step 2:** We calculate the total cost of the routes from the source node to the NFV node by adding the route's energy-cost and activation cost of that NFV node.
- **Step 3:** We repeat the process at steps 1 and 2 for the same source to all NFV node pairs.
- **Step 4:** We select the best NFV nodes based on the energy-cost, availability of the service chain, and the capacity of that NFV node.

- **Step 5:** The whole process is repeated for all source nodes.

At the design steps, we decide to choose the assignment based on the paths with least assignment cost, which makes our heuristic a greedy heuristic in nature [67].

3.3.2 Heuristic

We develop the heuristic *Assignment, and Path Selector (APS)* and present it in Algorithm 1. In the APS algorithm, we assign source nodes to the NFV node over energy-efficient and interference-aware routes, which minimize the overall energy consumption and activation cost of all the NFV nodes.

Algorithm 1 Assignment and Path Selector Algorithm (APS)

Input
NFV nodes: N
Source nodes: S
Output
NFV nodes map: X_{SN}
All routes map: Y_{SN}

1: **for** $s \in S$ **do**
2: **for** $n \in N$ **do**
3: $Routes_{short} \leftarrow$ get 3-shortest routes from r to n
4: $Routes_{energy} \leftarrow$ get 2 energy-cost routes from $Routes_{short}$
5: $primary_s[n], secondary_s[n] \leftarrow$ sort($Routes_{energy}$, link-strength)
6: $EnergyCost_s \leftarrow$ get total Energy ($primary_s[n]$, $secondary_s[n]$)
7: $c_n \leftarrow$ get the activation cost of n
8: $Cost_s[n] \leftarrow$ totalCost($EnergyCost_s, c_n$)
9: **end for**
10: $N_{sorted} \leftarrow$ sort the set of NFV($Cost_s, N$)
11: **for all** $nfv_s \in N_{sorted}$ **do**
12: **if** $nfv_s.capacity \leq CP_n$ & $D_{fs} \leq S_{fn}$ **then**
13: $nfv_s.capacity \leftarrow nfv_s.capacity + 1$
14: $S_{fn} \leftarrow S_{fn} - 1$
15: $D_{fs} \leftarrow D_{fs} - 1$
16: break
17: **end if**
18: **end for**
19: $X_{SN}.append(s, nfv_s)$
20: $Y_{SN}.append(s, primary_s[nfv_s], secondary_s[nfv_s])$
21: **end forreturn**X_{SN}, Y_{SN}

First, we calculate $k-$disjoint shortest routes where $k = 3$ from a source node to an NFV node. We deploy Dijkstra's shortest path algorithm to achieve that. In this calculation, we first call the Dijkstra's shortest path algorithm on the network graph. After getting the first shortest path, we remove it from the network graph and call the shortest path algorithm to get the second shortest path. Similarly, we get the third shortest path. Upon finding the three disjoint paths, we select 2 best routes from them based on the total energy-cost of the routes (line 4). The nodes participating in the routes with higher residual energy incur lower cost, while the nodes with lower residual energy incur a higher cost. After that, we sort the two routes into primary and secondary routes based on their overall link quality (RSSI value) (line 5). Later, these two routes are used to distribute the traffic by the source nodes—the higher the RSSI value, the better the link quality.

Next, we move to calculate the total cost of assigning the source node to the NFV node by adding the routes' total energy-cost and corresponding activation cost (CPU, memory, and energy) of the NFV node (line 6 to line 8). This process is repeated for all the NFV nodes for the same source node. We then sort the list of NFV nodes based on minimizing the total assignment cost. For the sorting operation, we employ the quick sort algorithm for better efficiency. Finally, we choose the minimum activation cost NFV node and check if the selected NFV node has the availability of the service-chain requested by the source node.

If the NFV node has the available capacity (memory and CPU) to serve the source node, we assign the source node to it. If the chosen NFV node does not satisfy the above-mentioned conditions, we chose the next minimum activation cost NFV node and continue to check the conditions. We repeat the process until we find the appropriate NFV node, or we reach the end of the NFV node list. In the latter case, we do not have any optimal assignment for that specific source node. Thus, we choose the best NFV node for the source node (line 11 to line 18). We repeat the entire process for all source nodes (line 2 to line 20).

For a better understanding of the proposed heuristic, we present an example in Fig. 3.5. The example considers a small grid topology of 10 nodes, where node 0 is the sink node, node 10 and 11 are two source nodes, node 2 and 3 are the NFV nodes, and others are the relay nodes. Our goal is to assign both of the source nodes to any

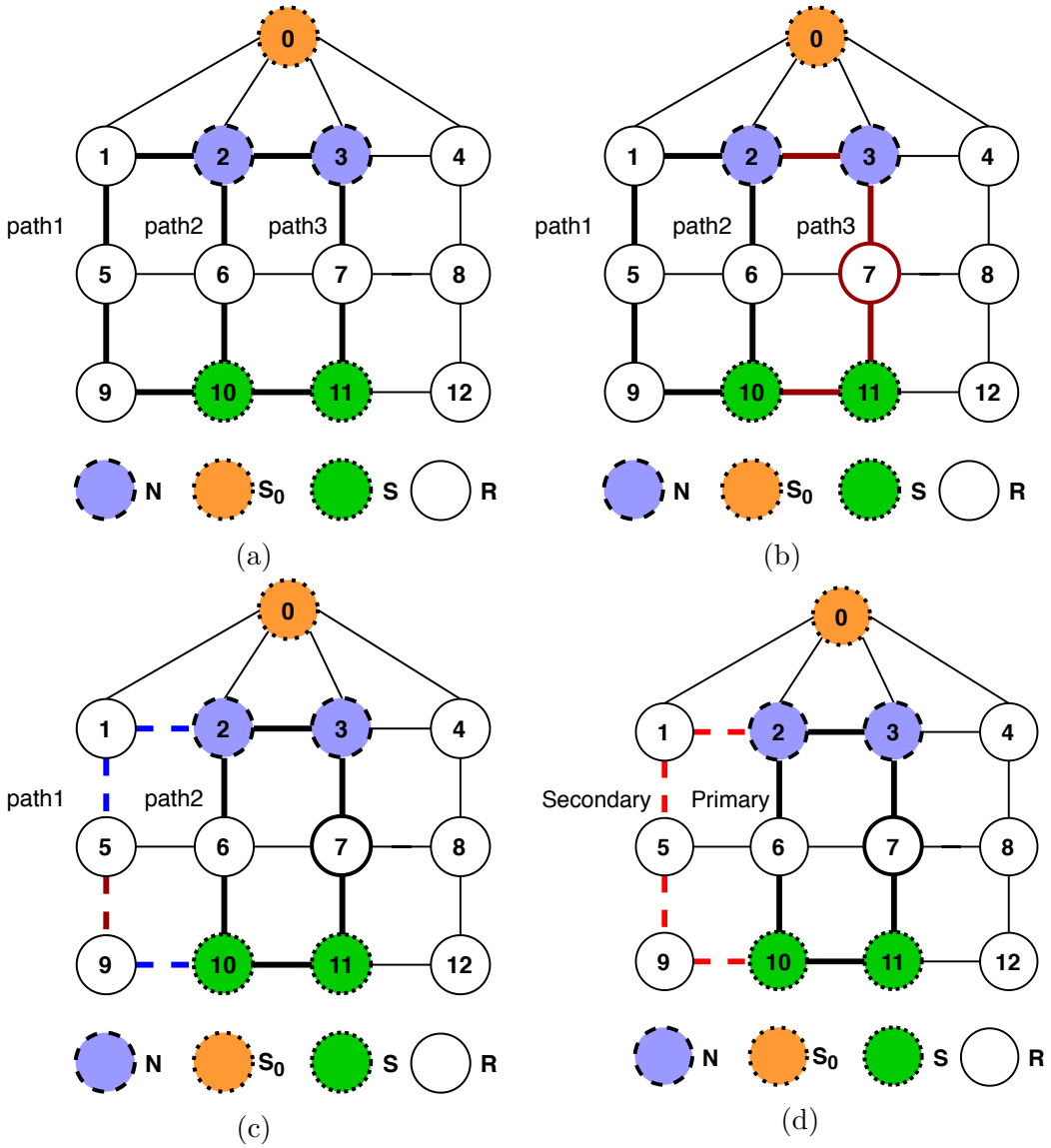


Figure 3.5: The illustration of heuristic steps.

of the NFV nodes. At first, the heuristic selects one source node and assign an NFV node to it. Then it moves on to the next source node and does the same. Let us assume the heuristic will pick node 10 at first and try to assign one of the NFV nodes 2 or 3 to it. We first calculate the three disjoint shortest paths from node 10 to node 2. Let's say the paths are path1-$10, 9, 5, 1, 2$, path2-$10, 6, 2$ and path3-$10, 11, 7, 3, 2$ (see Fig. 3.5a). We have to choose two best paths among the three paths - path1, path2, and path3 based on their energy-cost. The higher the residual energy of the nodes in the path, the lower its energy cost. Let us assume, all the nodes in the

path1, path2, and path3 have similar residual energy except node 7. Node 7 has lower residual energy, which eventually incurs higher energy-cost.

In particular, path3 has higher energy-cost among the three disjoint routes from node 10 to node 2 (see Fig. 3.5b). Thus, we select path1 and path2 for the next stage. We need to sort the paths into primary and secondary based on the link strength of each path (see Fig. 3.5c). The higher the RSSI value of a link, the higher its quality and lesser is the probability of having interference. In this instance, suppose the link $(9, 5)$ has lower strength, which makes the path2 as the secondary path (see Fig. 3.5d). If any two value is similar, we take the path with the shortest length. We then calculate the total assignment cost of the NFV node 2 for the source node 10 by adding the activation cost and both routes' total energy cost. This process repeats for source 10 and NFV node 3 pair. However, there are only two disjoint routes between them, so after the route calculation, we move to the part where we sort them based on the link strength into primary and secondary routes. Based upon minimizing the total-cost of assignment (activation and routes' energy-cost), we sort the NFV nodes. Then, we select the best NFV node with available capacity and available service-chain at the NFV node for the source node 10. The same process is repeated for the source node 11, and either NFV node 2 or 3 are assigned to it.

3.4 SDN/NFV Architecture

Being a part of the family of LLNs, the IoT network domain requires to operate with interference-prone and resource constraint environment. Due to its unreliable communication medium and ever-changing network state, it is essential to adapt new architecture to promote the programmability in the IoT network domain. We envision that SDN based architecture can enable dynamic reconfigurability, which in turn can improve the network performances. We also introduce the NFV capability in our design to provide the feature of dynamic deployment of services in the network by capitalizing on the SDN based architecture as a platform. So, we move to design an SDN based node architecture with NFV capability for IoT network. We define four types of nodes: sink, source, relay, and NFV. Among them, the source only has the SDN capability, while others can demand both the SDN and NFV capabilities. The relay nodes are there to complete the connectivity among the source, NFV,

and the Sink node, whereas both the relay and the NFV node has the ability to transport the traffic generated by the source nodes. Only the sink node hosts the embedded controller. In this section, we first describe the proposed SDN/NFV based node architecture and then we move forward to describe different functional module, which we implement in the SDN controller to monitor and manage the SDN/NFV based IoT network.

3.4.1 SDN/NFV Node Architecture

We use μSDN architecture as our foundation to build our SDN/NFV node architecture. Because, μSDN has already offered optimization in architecture, protocol, memory, and controller level to adapt the SDN with the IoT network domain. It also supports the compatibility with IPv6 and RPL, which has become almost a prerequisite for any new IoT solutions. The proposed node architecture is presented in Fig. 3.6. In our design, we introduced two new modules on top of the μSDN node architecture. They are the NFV Management module (NMM) and Route Management Module (RMM). The NMM helps the controller to maintain and dynamically deploy necessary VNFs to the NFV capable nodes. On the other hand, the RMM stores and maintains the routes towards the sink node via activated NFV nodes set by the controller. In the following, we describe the operations of NMM and RMM modules.

NFV Management Module (NMM): The NMM is the module from where all the NFV related configurations and functionalities are controlled. All the information necessary to deploy a VNF or form a service chain, are held at the NMM. The NMM contains mainly two components: *VNF container* and *VNF manager*. The VNF container holds and maintains all the available VNFs such as data aggregation, NAT, Firewall, etc. On the other hand, the VNF manager manages the main deployment of VNFs. It basically provides an API to the NFV enabled node, through which the controller will be able to deploy necessary VNFs on-demand. The VNF managers also maintain the available service-chains.

Route Management Module (RMM): The RMM module maintains all the

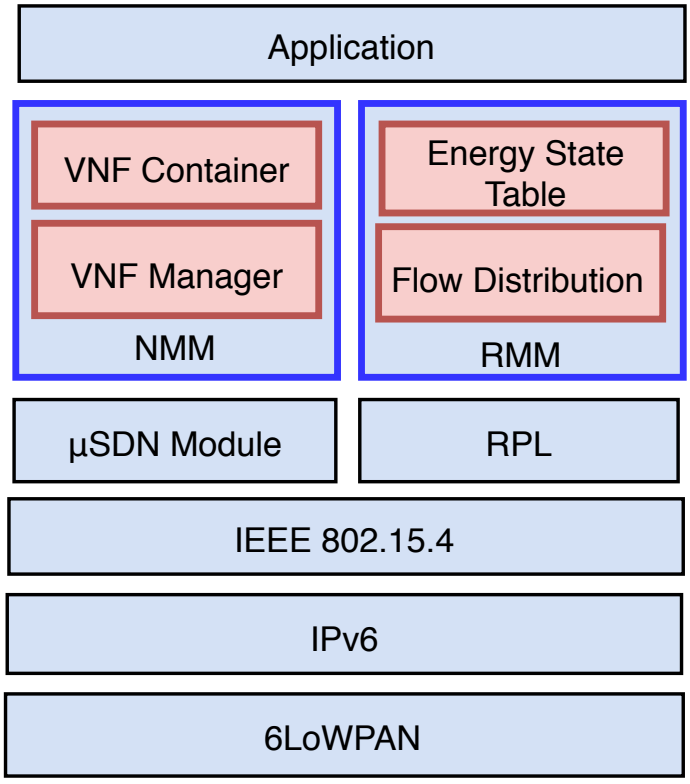


Figure 3.6: The SDN/NFV enabled node architecture.

information related to the route configuration of the network. It maintains an energy-state table to hold the energy-state and link strength (RSSI) information of its neighbors. Also, it periodically shares this information with the sink node that holds the SDN controller in it. The controller also can poll this information whenever this is necessary. The controller uses this information for the activation of an optimal number of NFV nodes as well as for the construction of the routes from source nodes to the sink node via the assigned NFV nodes. The RMM module also holds the configured routes in the flow distribution table at both source nodes and NFV nodes.

3.4.2 Functional Modules of SDN Controller

In the proposed design, the controller is placed at the sink node. In our design, we use an embedded controller provided by the μSDN architecture. For the smooth operation of different network operations, we use five types of control packets. They

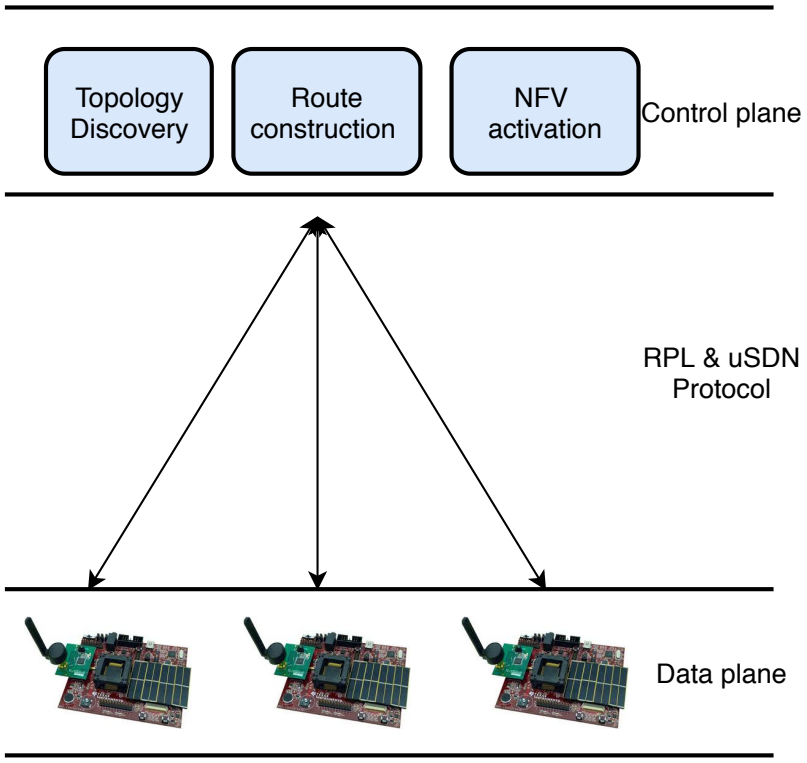


Figure 3.7: Functional modules of the proposed SDN/NFV architecture.

are: *Node-state update (NSU)*, *Flow table query (FTQ)*, *Flow table set (FTS)*, *Configuration (CONF)*, and *NFV configuration (NFV-CONF)*. It also uses RPL control messages. Using these control messages, the controller at the sink node can configure the SDN/NFV enabled nodes as per the network policy. Fig. 3.7 presents all the available functional modules of the controller. In the following, we describe the operations of different functional modules of the controller.

Topology Discovery: In this phase, the controller mainly uses RPL control messages. After the DODAG is formed, the network nodes send RPL DAO messages to the controller. After receiving this message, in response, the controller sends the CONF messages to all the requesting network nodes. The CONF message contains configuration metrics (e.g., NSU timer, frequency). The reception of the CONF messages works as the acknowledgment of a successful controller join operation. It is to be noted that the state table of the controller gets updated during the DAO message exchange. For the other nodes, it is updated during the initial DIS/DIO message exchange. The nodes periodically send requests to the neighboring nodes to update

their state table. The controller periodically collects the state table information, updates its own table, and maintains the routing topology. For the topology discovery and fallback measures in the case of any failure to contact the controller, the proposed architecture uses RPL protocol. The control messages used in this module are illustrated in Fig. 3.8.

Route construction and NFV node activation: This module is responsible for running the heuristic that we proposed in Algorithm. 1. The heuristic runs on the constructed routing topology to get minimum activated NFV nodes and their corresponding assignment as well as energy and interference-aware routes. The calculated routes and the corresponding assigned NFV nodes' information are stored in RMM and NMM, respectively. After that, the controller generates the NFV-CONF message with configuration information and sends them towards the NFV nodes. Upon receiving the NFV-CONF message, the NMM of the chosen NFV nodes activates the corresponding VNF. The controller also sends the assignment information of the activated NFV nodes to the respective source nodes. Upon receiving the information, the source nodes initiate the FTQ towards the controller. In response, the controller generates the FTS messages with two flow table entries (primary and secondary).

The source nodes use these two flow table entries to alternate between two routes. In particular, the source nodes alternate after a pre-determined number of transmissions. This alternating of routes enable the protocol to distribute the load among the relay nodes, which in turn increases the longevity of the IoT network devices. The sources forward packet as per the configured flow table entries, while the relay nodes use *Source-routing Header* (SRH) to transport the traffic towards the destination. In source-routing, the next hop information is embedded in the packet header. The relay nodes parse that information and act accordingly. In contrast, the NFV node upon receiving the traffic store them in a buffer. When the buffer gets full, it processes the packets as per the VNFs or service chain dictates. Upon processing, the NFV nodes initiate the FTQ-FTS message exchange if there is no matching rule for the packets to deliver. Thus, the route configuration control message complexity is proportional to the number of sources and NFV nodes. This route configuration and NFV activation occur at the network initialization or if there is a significant change in the network state. The Fig. 3.8 contains the control message exchanges used in this module.

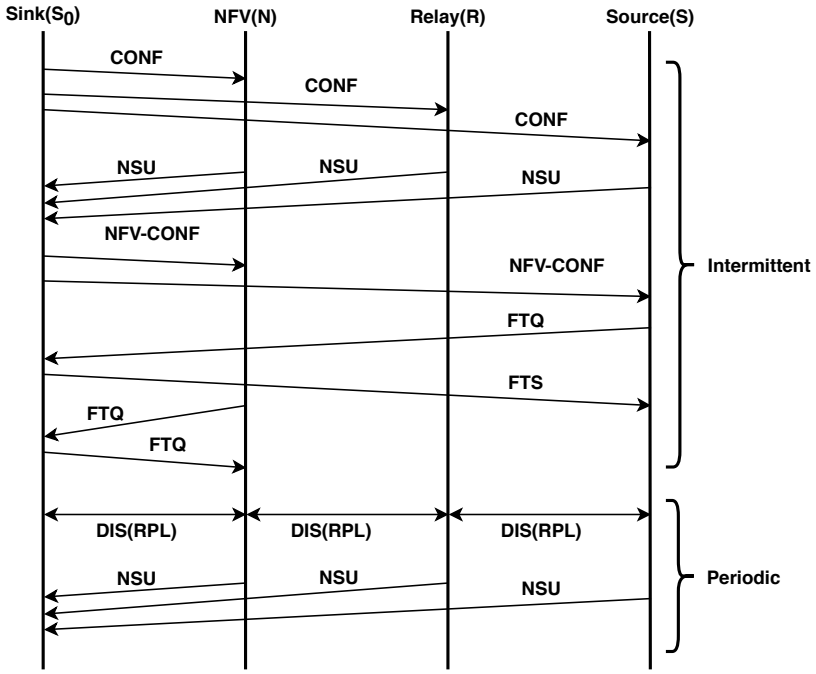


Figure 3.8: The flow of the control message exchange.

Network state collection: The controller collects the network state information periodically from all IoT nodes using the NSU control message. The NSU message contains state information such as energy levels, link quality, etc. Also, a node sends the same control message if its energy-level goes below the operational threshold, or there is a significant change in the network. The controller uses this information to maintain the routing topology and to configure the network accordingly. The summary of all the control message exchange is presented in Fig. 3.8.

Network reconfiguration: The controller maintains the network state information and routing topology in order to reconfigure the network fully or partially when it becomes necessary. The reconfiguration occurs only in the case when there is a significant change (e.g., network gets disconnected due to insufficient node energy or interference at the communication medium) [33]. The controller runs the Dynamic Route Selection (DRS) algorithm, which is illustrated in 2 to reconfigure the network. In particular, the DRS calls the APS algorithm based on the network state (line 3 to line 10) in order to get new sets of NFV nodes to be activated or new routes to the previously assigned NFV nodes from the sources. After initialization, the DRS can invoke the APS algorithm if any or some of the nodes go below the energy threshold

Algorithm 2 Dynamic Route Selection (DRS)

Input
State of network: *state*
Output
NFV nodes map: X_{SN}
All routes map: Y_{SN}

1: $N \leftarrow$ get list of NFV nodes.
2: $S \leftarrow$ get list of source nodes.
3: **if** *state* is initialization **then**
4: $X_{SN}, Y_{SN} \leftarrow \text{APS}(N, S)$
5: *state* $\leftarrow$ update the network state.
6: **else**
7: **if** any $e_j \leq E_j \| \ r_{ij} \leq R_{ij}$ **then**
8: $X_{SN}, Y_{SN} \leftarrow \text{APS}(N, S)$
9: **end if**
10: **end if return** X_{SN}, Y_{SN}

or any link quality falls below the RSSI threshold (line 7 to line 9). In our present implementation, we let DRS invoke APS whenever any NFV nodes or a certain number of relay nodes goes below the energy threshold so that a significant portion of the source node gets disconnected for their assigned NFV nodes.

Chapter 4

Evaluation and Discussion of Result

4.1 Evaluation Setup

In this section, we present the environment we use to evaluate both our ILP model and heuristic (Algorithm. 1) that are proposed in the previous chapter (Chapter 3). At first, we describe the model evaluation setup then move to the heuristic evaluation setup.

4.1.1 Model Evaluation

Our proposed ILP formulation tries to optimize the minimum activation of NFV nodes as well as assign the source nodes to the activated NFV nodes over energy and interference aware routes. To solve the model, we use CPLEX solver, where we use random topology to emulate the network structure.

Environment: We use IBM ILOG CPLEX Optimization Studio (often informally referred to simply as CPLEX), which is an optimization software package developed by IBM. The CPLEX optimizer can mathematically model real life issues and solve them using powerful tools. It increases solving efficiency, accuracy, and reduction of cost by providing precise and logical decisions for an optimization problem [68] [69]. The CPLEX solver can solve a wide range of problem types. Such as [68],

- *Integer Linear Programming (ILP)*
- *Mixed Integer Linear Programming (MILP)*
- *Quadratic Programming*
- *Quadratically Constrained Programming*

As our proposed model is an ILP problem, we choose to apply a CPLEX solver to solve it. For this purpose, we use python API 3.7.4v for the CPLEX library, referred

to as *DOCplex*. The CPLEX library offers two ways to solve a model. With a cloud solver (needs license and specialized access) and other with a local solver based on the host machine (limited capacity). In this work, we opt to use the local solver.

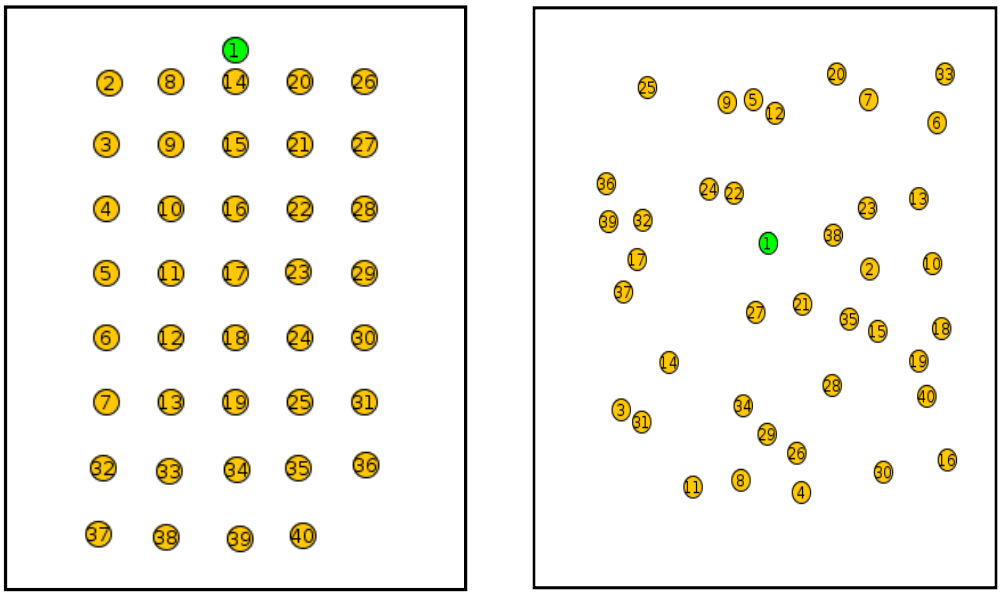
Topology and evaluation criteria: For this evaluation, we use 20 nodes random topology where we consider 3 NFV nodes and 6 source nodes. We also consider 100 transmissions for each source nodes to get the average results. In our evaluation, we try to show the correctness of our proposed model, i.e., if it does what it supposed to do. First, we calculate the number of activated NFV nodes by varying the number of sources and the NFV node's capacity. We intend to see if the number of activated NFV node varies with the variation of the number of source nodes and NFV node capacities. We also calculate the average hop distance from the sources to the assigned NFV nodes by changing the activation cost. Because with increasing activation cost, the less number of NFV nodes are to be activated with the source node assigned to them over longer routes on an average. We furthermore, measure the distribution of the residual energy of the nodes to prove the point that the model indeed helps to reduce the energy distribution and consumption.

4.1.2 Heuristic Evaluation

As our proposed ILP model is not feasible under large scale setup, we develop a heuristic presented in Algorithm. 1. To accommodate and implement the heuristic, we design an SDN/NFV architecture based on μSDN architecture.

Environment: To evaluate our proposed heuristic, we deploy our proposed SDN/NFV architecture in Contiki OS [34]. We simulate the IoT network and evaluate our heuristic on the Cooja simulator based on this Contiki OS. Contiki is a wireless sensor network operating system that consists of its own set of kernels, processing instructions, libraries, etc. [38]. It is specialized in designing and programming a smart objects applications such as IoT application. It is written in C programming language and comes with a rich library to develop products in C language, making it very portable to different architecture like Texas Instruments (TI)s *MSP430*.

Contiki OS provides a great simulator named Cooja, a Java-based simulator specifically designed for running wireless sensor-based network simulations. The great thing



(a) (b)

Figure 4.1: Example of grid and random topology.

about this simulator that it has built-in C compiler and wrapper libraries that enable it to run C language-based applications. Through Cooja, we can run programs natively on the host CPU or run on MSP430 CPU.

For the controller of the SDN/NFV architecture, we move to deploy an embedded controller, Atom [10]. The controller is the brain of an SDN based architecture. In our implementation, we choose to use the embedded controller to minimize the controller communication overhead. Moreover, the controller is part of the μSDN architecture, which is compatible with μSDN protocol optimized to meet the challenges of the IoT network domain.

Topology and IoT device configuration: For the heuristic evaluation we use both Grid (Fig. 4.1a) and Random (Fig. 4.1b) topology of 40 nodes. In each case of topology, we consider 10 source nodes, 5 NFV nodes, one sink/controller node, and others as relay nodes. Each topology is generated in a $400sq.m$ area where each node has transmission range 50m and interference range as 50m. For the generation of random topology, the nodes are generated following uniform distribution and different seeds within same 400sq.m area.

For each IoT device we consider TI's MSP430F5438CPU (Fig. 4.2) and CC2420 radio [10]. For the power supply, we consider a coin-type lithium-ion battery with

Figure 4.2: An IoT device with MSP430F5438 CPU and C2240 radio.

3V and 150 mA-h power rating [70]. The energy consumption rate for the mentioned radio is for transmission, which is 17.7 mA, and for the reception, it is 20.01 mA. In our initial investigation, we have discovered that each successful transmission of a data packet consumes 40mj of energy. Any node becomes inoperable if the available energy of that particular node falls below the 40mj. We refer to this as our energy threshold. We also have measured the standard link strength of a link in terms of RSSI value. The value varies between -41dbm to -43dbm. After considering the RSSI offset, it is determined that our RSSI threshold is -45dbm, which is the same as the widely used and standard link quality threshold [48].

Performance metric: For the evaluation of the heuristic, we use the following performance metric.

- *Communication energy consumption (mj):* is the total energy consumed due to the data traffic transmission and reception between a source and destination pair.

- *Packet Delivery Ratio (PDR):* is the ratio of the total number of packets that are successfully received at the destination and the total number of packets that are generated at the source for a specific source-destination pair.

- *Control Overhead:* is the total number of control packets that need to be exchanged in the entire operation among the nodes.

Table 4.1: The list of parameters used in the evaluation.

Parameter	**Settings**
Duration	10 min
MAC Layer	ContikiMAC
Transmission Range	100m
Interference Range	100m
Max bitrate	10 bit/sec
Link Quality	90%
Radio Medium	UDGM(distance loss)
RPL Mode	Non-Storing
RPL Route Lifetime	5min
Flowtable Lifetime	10min

Evaluation setup: We evaluated the proposed heuristic based on energy consumption, PDR, network lifetime, and control overhead on both grid and random topology of 40 nodes. We vary the hop distances from the source nodes and the sink node to investigate the effect of network size on the above-mentioned performance metrics. We also investigate the effect of various traffic loads on the performance metric. For each set of evaluations, we run the simulation 50 times. In each run, we change the positions of NFV nodes to achieve the effect of different topologies. We take the average of the simulations with 95% confidence interval.

For the implementation of our proposed solution of SDN/NFV architecture for IoT network, we use 10 as the NFV-buffer size, i.e., after receiving ten packets from the source, NFV node starts aggregating the packets. After aggregating the NFV node sends them towards the sink node. The source uses the same NFV-buffer size as the moment to alternate the transmission between primary and secondary routes to distribute the load and consumption of energy. We also consider the service-chain that has one VNF, i.e., data aggregation through a VNF is part of the service-chains. We use 3 source nodes as the default capacity for each NFV nodes.

The controller uses the DRS algorithm (Algorithm. 2) to call the heuristic APS algorithm (Algorithm. 1) in the case of initialization or any local change in the network state (any node goes out of energy or link strength falls below the RSSI threshold). In the current implementation, if any node goes below the 1% of the initial energy,

it becomes inoperable, and that is why we choose this to be our energy-threshold. For the RSSI threshold, we use -45 dbm, i.e. if any link's RSSI value goes below this threshold it becomes highly prone to interference. Right now, this invocation decision is entirely based on a learning parameter, which we intend to revisit in our future work. All the parameters used in the evaluation for the heuristic are summarized in Table. 4.1.

Packet Generation: For the data generation, we have used both synthetic and real sensor data. For the transmission, in both cases, we have used the UDP type of packet with 16 Byte size where the payload is 2 bytes. In the case of synthetic data, we have generated some random integer data. On the other hand, in the case of real data, we have used a dataset that contains the temperature sensor data, which was collected from WSU CASAS smart home project [71]. The data is from a volunteer adult home with temperature and motion sensors collected over two years through IoT devices. The temperature sensor, which was used to collect the data, is a temperature sensor compatible with TI's MSP430 CPU [72].

4.2 Discussion of Results

In this section, we discuss the results of the evaluation of our proposed model first. Then we move on to the results related to the implementation of our proposed energy-efficient and interference-aware SDN/NFV framework, which consists of the implementation of heuristic, corresponding node architecture and protocols. We refer the framework as EA-SDN/NFV and the framework without any heuristic or optimization as SDN/NFV in this section. We discuss and compare the performance of the EA-SDN/NFV with SDN/NFV and some of the contemporary protocol such as μSDN and SDN-WISE. We also discuss the effects of control overhead that an SDN based architecture brings. After that, we present a scenario where the reconfigurability feature of SDN based architecture comes into play.

4.2.1 Model Evaluation

In this part, we discuss the evaluation result of the ILP model we proposed in Chapter.3. We solve the ILP formulation using CPLEX solver.

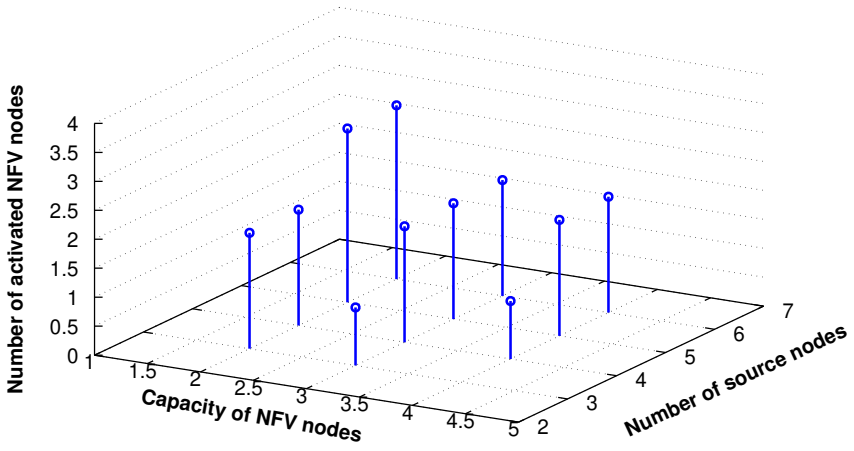


Figure 4.3: The number of activated NFV nodes.

Effect of capacity and number of sources on activating NFV nodes: In Fig. 4.3, we show the effect of the capacity of NFV nodes and the number of the source nodes on the number of NFV nodes that need to be activated. If the capacity of an NFV node increases, then the number of activated NFV nodes decreases, given the number of source nodes remains the same. Because with increasing capacity, the NFV node has more resources to serve the requesting source nodes. So, with a fixed number of the source node and with increasing serving capacity, the number of activating NFV node decreases in order to maintain the budget constraint as well as minimizing the network cost. On the other hand, when the number of the source nodes increases, the activated NFV nodes' number also increases with a constant capacity of NFV nodes. This is due to the fact that with the increasing demand of the source nodes, the number of activating NFV nodes must increase to meet those demands. In summary, the effect depends on meeting the existing demand of the source nodes with the available capacity of the NFV nodes, which clearly illustrates the budget constraints of the model.

The effect of activation cost on hop distance: In Fig. 4.4, we present the effect of increasing activation cost on the average hop distances of the source nodes from the sink node. In particular, with the increasing activation cost, the number of activated NFV nodes decreases, and consequently, the source nodes are assigned

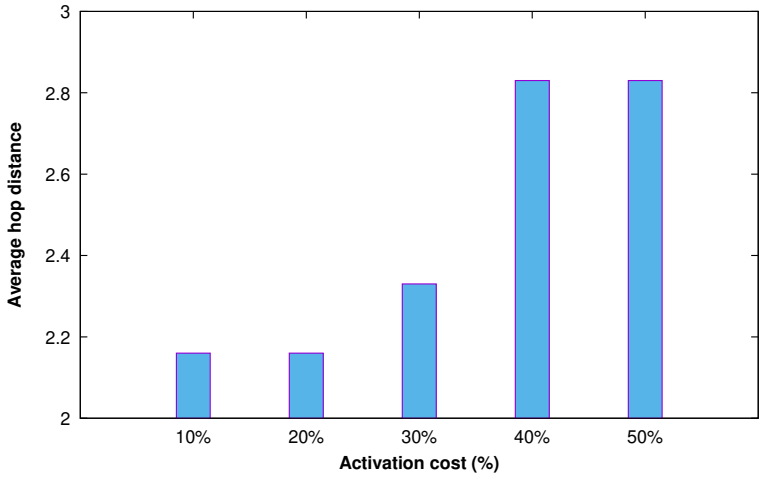


Figure 4.4: The change in hop distances.

over larger routes to the NFV nodes as the most optimal NFV nodes might not be activated in order to minimize the energy consumption. In fact, the activation cost is proportional to the total amount of energy available at an NFV node, which has a power rating of 150mA-h and 3V. If we increase the activation cost of NFV node from 10% to 50% of the initial energy, then the total number of activated NFV nodes decreases, which results in a competition among the source nodes to get their optimal NFV node. As a result, some of the source nodes are assigned to the second-best NFV node over a longer route to minimize the total network energy cost and activation cost. Eventually, this causes increased average hop distances from the sources to the sink node. Thus, the network operator can estimate their budget and keep themselves from over or under-provisioning.

The effect on the distribution of residual energy: The distribution of residual energy over all the nodes in the representative network is illustrated in Fig. 4.5. We observe that the residual energy of different nodes is well distributed over 100 transmissions. The distribution of traffic through primary and secondary paths has really paid off. The nodes retain more energy over the course of transmission. The NFV nodes have the most retention of energy as the number of transmission of NFV nodes towards the sink node is reduced by the factor of aggregation rate, i.e., the NFV nodes only transmit the aggregated packet. These results also show that the proposed model optimizes the total energy distribution of the network and increases the network lifetime.

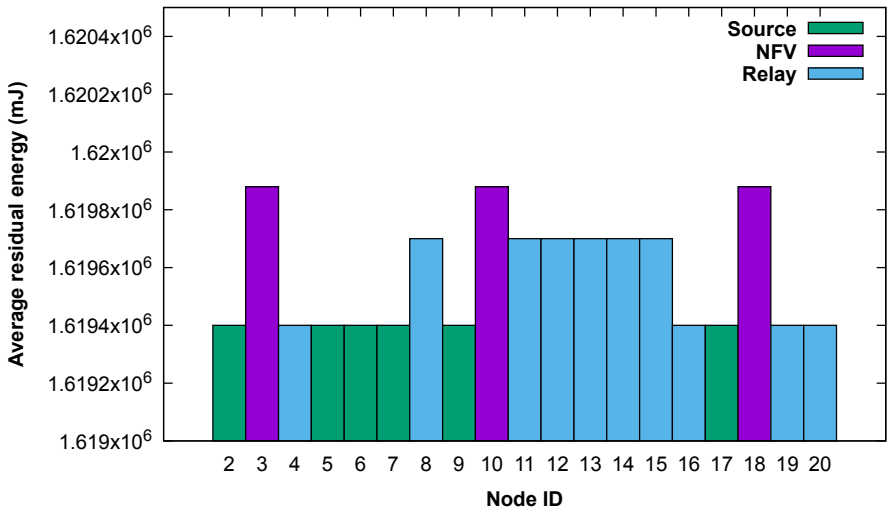
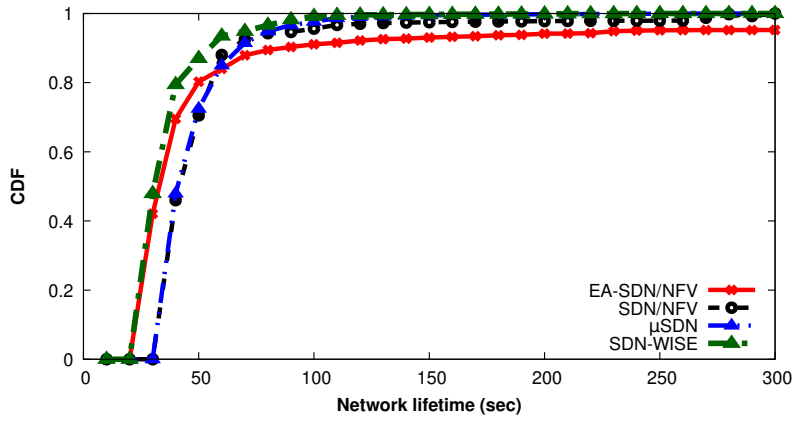


Figure 4.5: The residual energy of nodes.

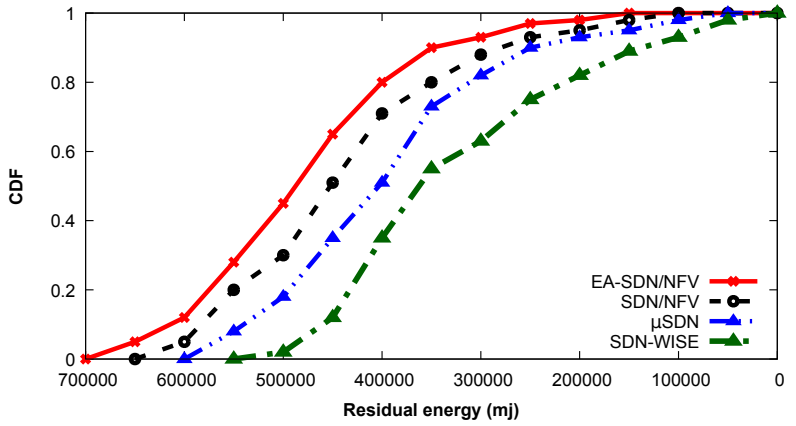
4.2.2 Heuristic Evaluation

We evaluate our proposed energy-efficient and interference-aware SDN/NFV framework (EA-SDN/NFV) and compare it with the SDN/NFV implementation with no optimization (SDN/NFV), μSDN and SDN-WISE. We run our evaluation on both the grid and random topology, while varying the hop distance and traffic load.

Effect on network lifetime: Fig. 4.6 presents the distribution of communication energy consumption(Fig. 4.6a) and the distribution of residual energy (Fig. 4.6b) on the grid topology. In particular, Fig. 4.6a shows that the EA-SDN/NFV slowly moves to the point of saturation than the other three comparing schemes. The consumption rate of EA-SDN/NFV is slower in this case. The distribution of traffic load over relay nodes and data-aggregation as the NFV function helps achieve this result. The SDN-WISE is the worst in this case where the SDN/NFV and μSDN takes the middle positions, respectively. The latter two do not have any NFV capabilities or any energy-interference optimization. To further investigate the effect on the network lifetime, we evaluate the residual energy of all four schemes after the end of the simulation, which is illustrated in Fig. 4.6b. The results follow the same pattern where the nodes in EA-SDN/NFV retain more energy than the other three schemes. In this case, also, the SDN-WISE comes up at the bottom, while SDN/NFV and



(a)

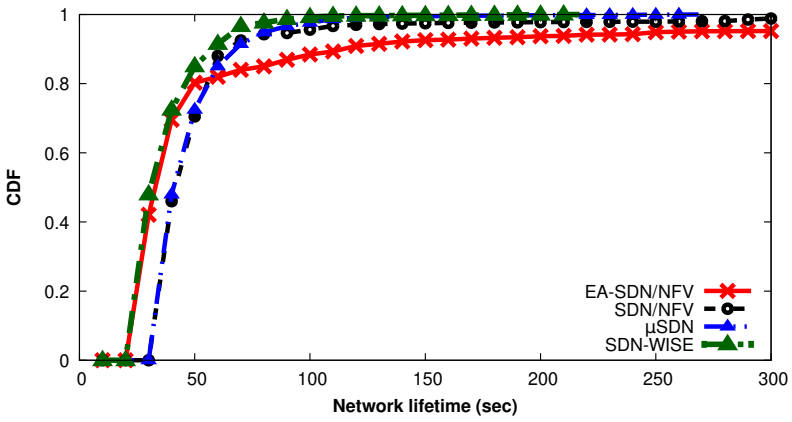


(b)

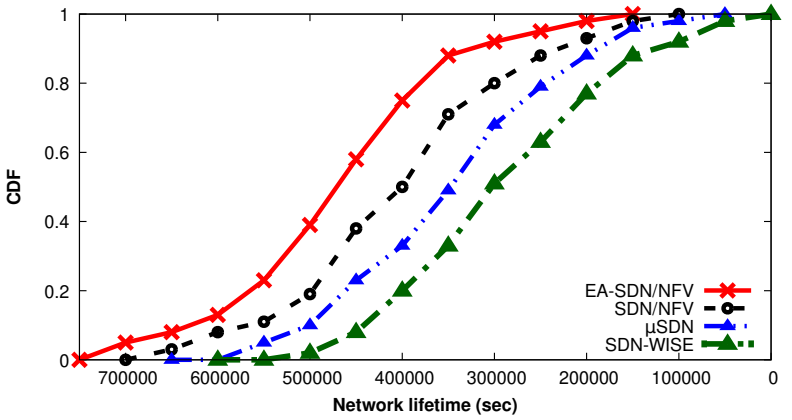
Figure 4.6: The distribution of communication energy consumption (a) and residual energy (b) in grid topology .

μSDN takes second and third place, respectively. Thus, our proposed solution, EA-SDN/NFV provides a better network lifetime than all other comparing schemes.

To investigate the effect of topology on the network lifetime, we evaluate the same set of experiments and observe the results on random topology. The evaluation results are illustrated in Fig. 4.7. The results show a similar trend as to Fig. 4.6. The EA-SDN/NFV shows better distribution in both communication energy consumption (Fig. 4.7a) and the residual energy (Fig. 4.7b). Due to the lacking of any optimization and NFV capabilities, the μSDN and SDN-WISE performs the worst. SDN/NFV



(a)

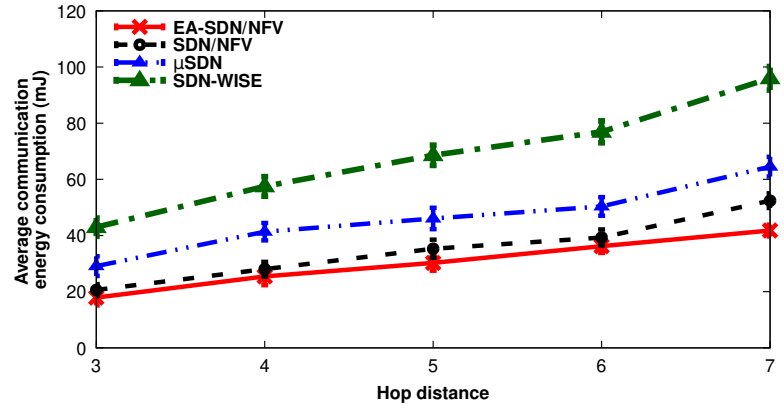


(b)

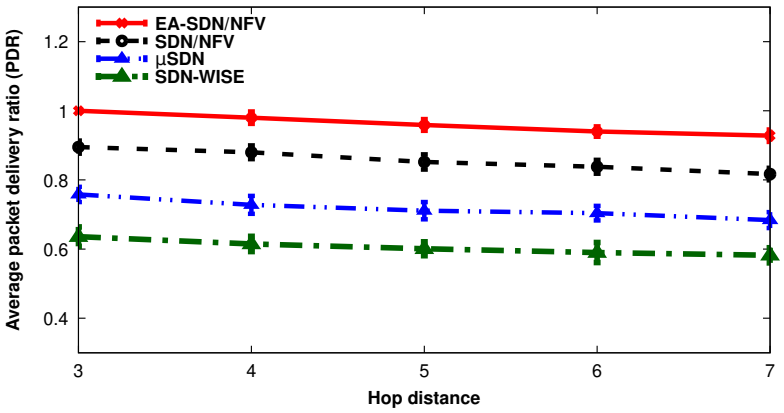
Figure 4.7: The distribution of communication energy consumption (a) and residual energy (b) in random topology.

comes as the second-best having no energy or interference optimization. In summary, both in the grid and random topology, our proposed solution (EA-SDN/NFV) promises a better network lifetime.

Effect of hop distances: To illustrate the effect of varying hop distances on the communication energy consumption and PDR, we present Fig. 4.8. In particular, we illustrate the effect on communication energy consumption for varying the hop distance in Fig. 4.8a where it shows that with increasing hop distance between the source nodes and the sink node, the communication energy consumption of the network increases. This is due to the fact that with increasing hop distance, the number of



(a)

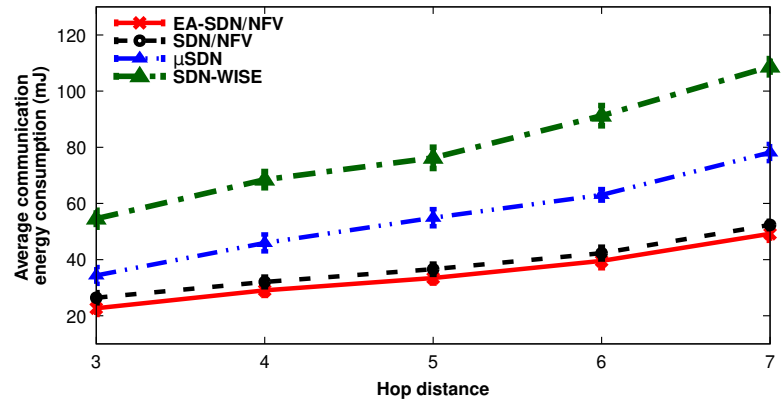


(b)

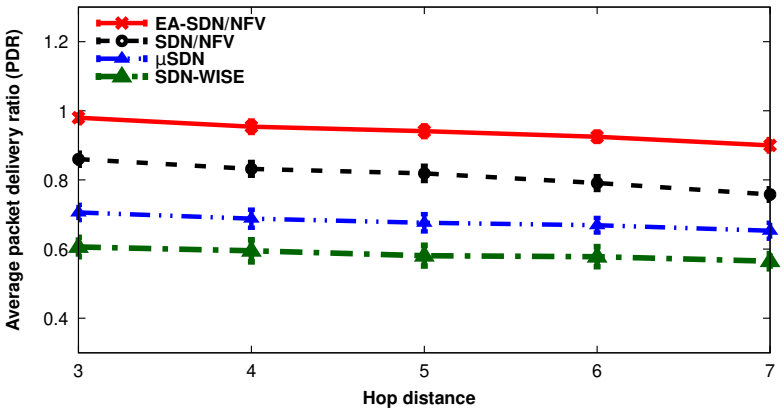
Figure 4.8: The average communication energy consumption (a) and PDR (b) in grid topology.

transmission increases to reach the final destination. This is true for all four schemes though, among them, the EA-SDN/NFV comes on top. We suspect that with load distribution and integration of data aggregation as NFV function contributes to the performance improvement of EA-SDN/NFV.

On the contrary, both SDN-WISE and μSDN performs worse, while the former comes at the bottom. Both of them create the case of interference near the sink node for not having an aggregator node. However, μSDN has some optimization and compatibility with RPL and IPv6, while SDN-WISE has no such optimization in the protocol level, which is why SDN-WISE performs the worst of the two. Though



(a)



(b)

Figure 4.9: The average communication energy consumption (a) and PDR (b) in random topology.

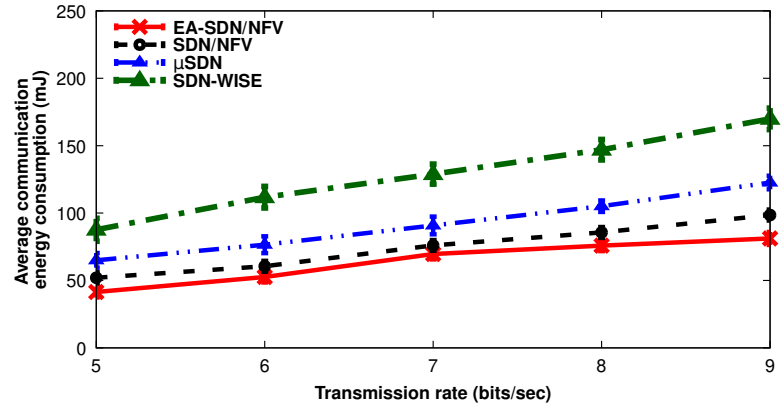
SDN/NFV performs better than these two due to having data aggregator as VNF deployed at the NFV node, it falls short to EA-SDN/NFV due to not having any communication energy or interference optimization. In fact, EA-SDN/NFV improves the communication energy consumption by 1.39x and 1.6x compared to μSDN and SDN-WISE, respectively.

Fig. 4.8b illustrates the PDR with varying hop distance. Following the same pattern as to Fig. 4.8a, the EA-SDN/NFV performs the best among all four comparing schemes. In particular, EA-SDN/NFV offers 1.36x and 1.58x better performance

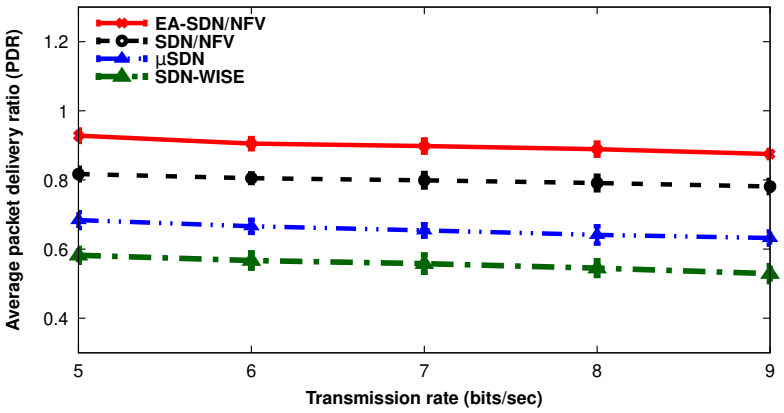
compared to μSDN and SDN-WISE, respectively. In the case of μSDN and SDN-WISE, the probability of interference is high near to the sink node, which is why we observe a diminishing performance in PDR. The main reason is the alternate path near to the sink node becomes very few. The high incoming traffic and less way to reach the destination makes a favorable condition for the interference to occur. Without any data aggregation, the μSDN and SDN-WISE fall victim to this effect. Also, the lower energy consumption rate of EA-SDN/NFV makes it suitable for long operation hours, which results in better PDR. Moreover, with data aggregation as VNF deployed at the NFV node, it reduces the traffic near the sink node, which in turn reduces the probability of interference.

We also investigated the effect of hop distance variation in random topology, and the evaluation is presented in Fig. 4.9. The results show similar trends as to the results presented in Fig. 4.8 where EA-SDN/NFV shows the best performance. In particular, in Fig. 4.9a, the effect on communication energy consumption is illustrated. EA-SDN/NFV saves 1.4x and 1.62x more energy compared to μSDN and SDN-WISE, respectively. The only difference here in random topology is that the nodes are located very near to each other, which presents the fact that the alternate path to the primary one is shorter on an average than in grid topology. This provides better communication energy consumption than that of grid topology. However, due to very closely placed nodes, the effect of interference is higher here, which shows in the slightly lower PDR than grid topology (Fig. 4.9b). Overall, EA-SDN/NFV performs best in the random topology also than the other three schemes.

The effect of traffic load: In these sets of evaluations, we consider the variation of transmission rate over the highest possible hop distance, i.e., we select the set of sources that are apart from the sink node with highest possible hop distance. We let the traffic flow at variable speed and observe the corresponding outcomes in the form of communication energy consumption and PDR. We illustrate the evaluation results for grid topology in Fig. 4.10. The effect on different traffic load on communication energy consumption is presented in Fig. 4.10a, which shows the same trends as in the other evaluation, here also EA-SDN/NFV shows better performance. With a higher traffic load, the average communication energy consumption increases in all four schemes. We suspect that the higher traffic volume increases the number of



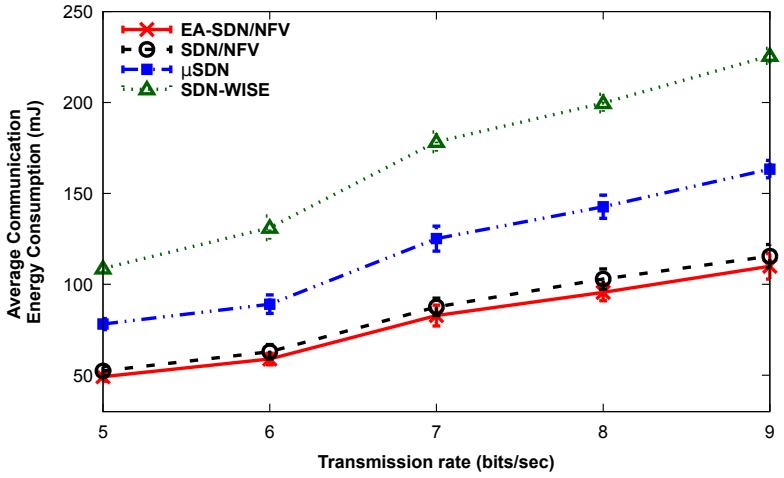
(a)



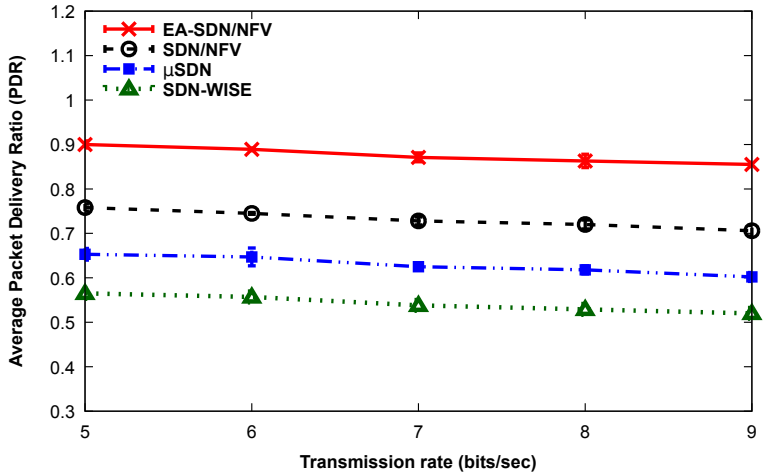
(b)

Figure 4.10: The impact of traffic loads on the average communication energy consumption (a) and PDR (b) in grid topology.

transmissions, which results in higher communication energy consumption. With a higher load, the relay nodes need to carry more traffic. As relay nodes are used again and again, they may lose their energy faster and goes into node failure. With the distribution of load into place, the EA-SDN/NFV handles this high traffic situation better than all other schemes, which is translated into the fact that it is 1.38x and 1.59x better than μSDN and SDN-WISE, respectively. It is to be noted that this improvement is more prominent with higher loads. With higher loads, other schemes suffer most due to not having the extra feature of load distribution and data aggregation of EA-SDN/NFV. This makes our solution better suited for a high volume traffic



(a)



(b)

Figure 4.11: The impact of traffic loads on the average communication energy consumption (a) and PDR (b) in random topology.

scenario.

On the other hand, the high traffic volume has impact on the PDR. With more traffic, the probability to become victim to the interference becomes higher, which is translated into the performance of all the schemes (Fig. 4.10b). We observe that the packets get dropped with the increasing load. Also, we suspect that due to high volume of traffic packets may face congestion and get dropped due to the overflowing of queues, which affects the average PDR of the schemes.

We also investigate the effect of traffic load for random topology(Fig. 4.11). In

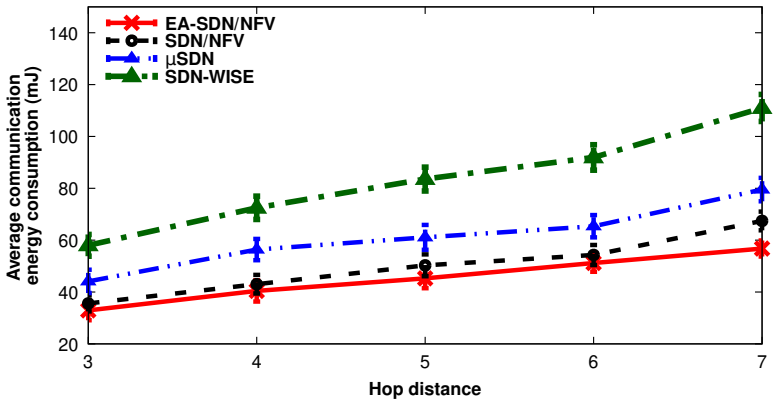


Figure 4.12: The average communication energy consumption with real data.

Fig. 4.11a, we show the average communication energy consumption in random topology with a varied load that follows the same pattern as of grid topology with EA-SDN/NFV provides the best performance. With increased load, the relay nodes need to carry more traffic. Though in random topology, the effect of interference is much higher, which is shown the performance trends for average PDR. Fig. 4.11b shows the average PDR in random topology with varied load, which also follows the same trends. With the interference mitigation mechanism in place, EA-SDN/NFV has a much better chance to handle this interference effect than other schemes. Overall, here also, EA-SDN/NFV performs better than all other three comparing schemes.

The effect of real sensor data: So far, in all evaluation, we have used dummy payload for the UDP packet. We also evaluate all four schemes under real sensor data. We use previously collected sensor data for the temperature to simulate the scenario where the schemes have to handle real sensor data (Fig 4.12). We run this evaluation in 40 node grid topology with the same setup as before. Following the same trends, the EA-SDN/NFV performs the best among four comparing schemes. For the data aggregation purpose, we have used data compression by arithmetic mean, which is less computationally heavy. This shows, even with real sensor data, our proposed solution performs better.

4.2.3 Control Overhead Analysis

By decoupling the data and control plane, any SDN based architectural protocol brings additional control overhead. Being an SDN based architecture, our proposed EA-SDN/NFV introduces some other control traffic for network monitoring and flow rule installation, which is not present in non-SDN based protocol like RPL. In order to investigate the effects of the control overhead over the performance, we evaluate the control overhead of 4 SDN based protocol: EA-SDN/NFV, SDN/NFV, μSDN, SDN-WISE. For this set of evaluation, we consider the 40 node grid topology with 5 source nodes and 3 NFV nodes. We measure the control overhead from four phases: *Initialization, Route Configuration, Topology Maintenance, and Update*. The initialization phase starts when the network comes into being. All the control messages related to topology discovery, configuration setup, etc. fall under this phase, i.e., all the control messages required to set up the network. All the control messages related to the transmission of data traffic such as flow rule query, flow rule setup, etc. fall in the route configuration phase, whereas maintenance of the routing topology falls under the topology maintenance phase. The update phase contains all the control messages needed to re-configure the network if its state changes (node or link failure). The detailed analysis of the control overhead is presented in Table. 4.2. In the table, we have summarized the control packets for all 4 SDN-based protocols of all phases except the topology maintenance phase. We observe that for topology maintenance phase, all the comparing protocol needs the same amount of control traffic.

Table 4.2: The comparison of control overhead.

Protocol	**Initialization** (packets)	**Route Configuration** (packets)	**Update** (packets)
SDN-WISE	80	40	40
μSDN	120	10	10
SDN/NFV	128	16	24
EA-SDN/NFV	128	16	24

At the initialization phase, SDN-WISE performs the best with the least number of control packets, whereas the EA-SDN/NFV has the highest number of the control packet. In this phase, SDN-WISE only uses 2 sets of TD packets: one is from

the controller and another from all the nodes towards the controller. On the contrary, all NFV-based protocols (SDN/NFV and EA-SDN/NFV) uses sets of CONF message, initial NSU messages, and NFV-CONF messages along with RPL control messages for the NFV functionalities configuration. It is evident that with the extra feature, these two NFV based protocols need more control packets to set up the network whereas SDN-WISE and μSDN do not need those control packets. However, we can see the trade-offs here with extra control packet we have protocols, which have better functionalities, features, compatibility, and performance. In the route configuration phase, except SDN-WISE, all other three schemes use source routing for data traffic forwarding, which contributes to their less number of control packets than SDN-WISE. However, having NFV capabilities, SDN/NFV, and EA-SDN/NFV have slightly higher control overhead than μSDN. In the case of SDN-WISE, it needs to install flow rules at all the sources and corresponding relay nodes, which contributes significantly to the higher number of control traffic. During the topology maintenance phase, as all of the comparing schemes use the same number of control traffics due to the same rate of message exchanges, we have not included that in the Table. 4.2.

In the update phase, SDN-WISE again uses the highest number of control packets as with network changes, it needs to update all the flow-rules of the source nodes and corresponding new relay nodes. On the other hand, the other three schemes use source-routing for data forwarding, which causes less control overhead. In the case of SDN/NFV and EA-SDN/NFV, we have observed the highest control overhead as they have NFV capabilities. We also show the corresponding performance of communication energy and PDR of all the comparing schemes compare to that of EA-SDN/NFV, which reflects the evaluation results presented before (Table. 4.3). Here, we observe that in terms of the control overhead EA-SDN/NFV has the worst performance, while SDN-WISE has the best. However, the performance is far better in EA-SDN/NFV than SDN-WISE. In summary, though our proposed EA-SDN/NFV protocol introduces more control overhead, the additional control overhead is comparable to other schemes. Furthermore, it offers better performance in terms of communication energy consumption, PDR, and network lifetime.

We suspect that the control overhead of our proposed protocol heavily depends on the topology configuration and its sizes. Let N be the number of NFV nodes,

Table 4.3: The summary performance comparison of EA-SDN/NFV (1x).

Protocol	**Control Overhead**	**Energy Consumption**	**PDR**
SDN-WISE	0.998x	1.6x	0.63x
μSDN	0.999x	1.39x	0.73x
SDN/NFV	.9992x	1.15x	0.88x

S be the number of source nodes, and R be the number of relay nodes. At each phase, the number of control messages directly depends on the number of N, S, and R as the number of sources and NFV nodes are minimal compared to the relay nodes the control overheads at all phases can be expressed as the growth function of R or $O(R)$. So, it is evident that the control overhead grows with the size of the network, in particular with the number of the relay nodes.

4.2.4 Network Reconfigurability

One of the main reasons to introduce the SDN/NFV framework is to introduce the reprogrammability to the network. The ever-changing dynamic nature of the IoT network domain needs the reconfiguration capabilities to maintain the network running. In our implementation, we introduce the dynamic reconfiguration concept using the DRS algorithm (Algorithm. 2). Though this algorithm, the controller can call the heuristic, APS algorithm (Algorithm. 1) based on the current state of the network. The configuration of the network nodes, i.e., to configure a network node based on its functionalities such as, as a source node or NFV node, occurs at two states: either at the initialization (network setup at the beginning) or at the local changes to the routing topology (node or link failure), which is called reconfiguration. To show this reconfiguration feature, we consider two scenarios: one, when an NFV node goes out of energy (a critical case), and the other, is when a relay node goes out of energy. For both of the scenarios, we consider 40 node grid topology with 10 source nodes and 5 NFV nodes (Fig. 4.1a).

Scenario 1 (NFV node failure): In Fig. 4.13, we show the effect of an NFV node failure upon the average PDR with or without invoking reconfiguration. The evaluation shows that the PDR drops at the time of the NFV node failure in both

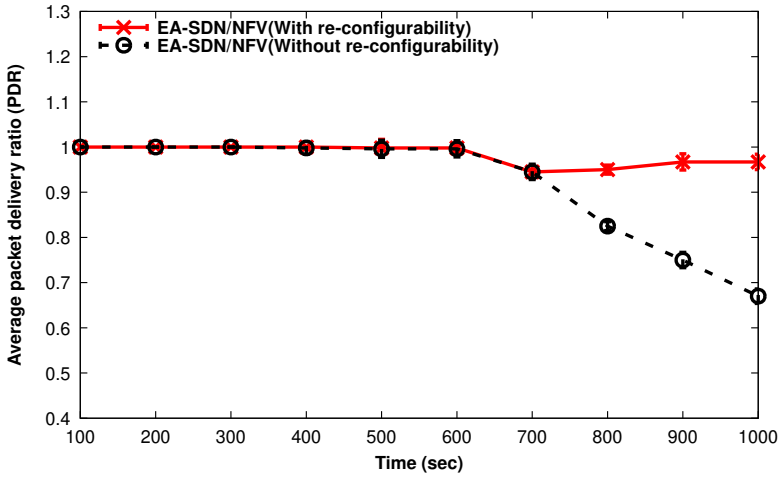
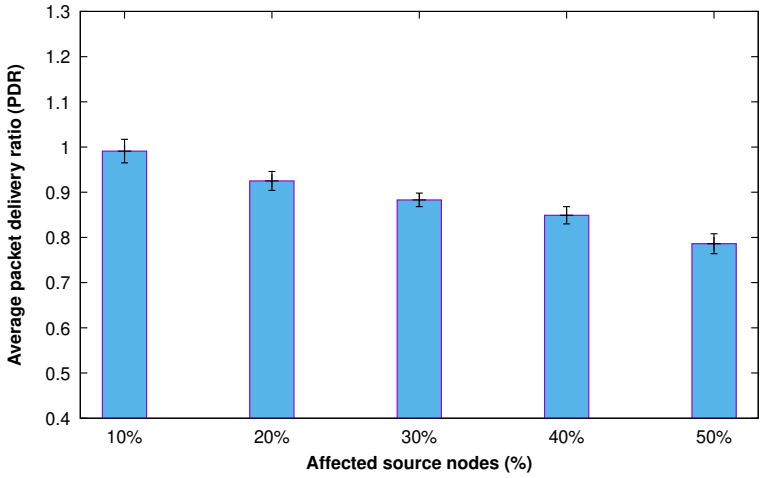


Figure 4.13: The average PDR with an NFV node failure.

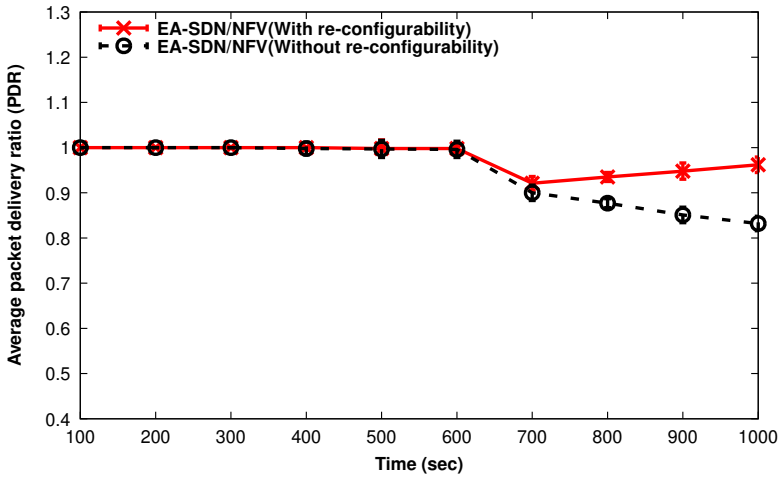
cases; however, in the case of re-configuration, it rises again after some time. In fact, after an NFV node fails, within thirteen seconds, the EA-SDN/NFV re-configures and assigns the affected source nodes to an alternative NFV node. The transmitted packets from the source node again start to be processed at the newly assigned NFV node and continue towards the sink node. However, lacking the re-configurable feature, the other case can not recover from the packet loss and thus have a diminishing PDR.

Scenario 2 (Relay node failure): In this set of evaluations, we investigate the effect of one or more relay node failure. This failure may affect one or more source nodes. Because one relay node can be a part of multiple routes that concerns over multiple source nodes. If a relay node fails and it turns out that it has been a part of multiple routes, then multiple source nodes can be without a way to reach its destination. At first, we investigate the effect on average PDR (Fig. 4.14a) of the network with a different number of affected source nodes due to one or more relay node failure. We observe that with an increasing number of affected source nodes, the average PDR decreases. So, in order to reduce the effect on PDR, we need to invoke the re-configuration as early and frequently as possible.

However, there is a relationship between the re-configuration and control overhead, which is presented in Table. 4.2. With frequent re-configuration, we will incur heavy control overhead. We need to come to the point of trade-off where we can balance the effect on PDR and the control overhead. We find the reconfiguration



(a)



(b)

Figure 4.14: The average PDR after relay nodes failure with affected sources (a) and with and without reconfiguration (b).

point at 20% affected sources as the balance. With that reconfiguration point, we further investigate the effect on PDR with or without invoking reconfiguration and illustrate the result in Fig. 4.14b. In this case, also, we find similar trends as to the results presented in Fig. 4.13. With the reconfiguration option, the PDR drops when relay node failure happens, but it catches up after being assigned to the NFV nodes with new relay nodes. However, without the reconfiguration option, the PDR keeps dropping. In summary, it is evident that the reconfiguration option not only can handle the unexpected, but it also improves the performance.

Chapter 5

All Relays with NFV Capability

In the previous chapters, we have proceeded with the assumption that we have a set of NFV enabled IoT nodes other than the source, sink, and relay nodes, which do not have NFV capabilities. We move to select the optimal NFV nodes from that NFV set to be activated. In this chapter, we go beyond that assumption and adjust our proposed model by introducing the concept. We eliminate the necessity of a specialized set of NFV nodes and consider all of the relay nodes with NFV capabilities, i.e., all of the relay nodes become possible candidates for NFV activation. Then we move to present the modified heuristic. Lastly, we evaluate the modified model and the corresponding heuristic. We compare the results with the EA-SDN/NFV with a given NFV set and discuss the results.

5.1 Problem Formulation

In our proposed ILP model ((3.24), subject to the constraint (3.7) to (3.23) and (3.25) to (3.27)), we consider that the only NFV nodes have the capability to host VNFs. We have to choose the right set of NFV nodes among those. We need to look at the problem from the assumption that all relay nodes have NFV capabilities. That means any relay node can be configured as an NFV node based on the demand of the source nodes. However, our main objective remains the same. We need to activate the minimum number of relay nodes as NFV nodes and assign the source nodes to them over energy-efficient and interference-aware routes.

Requirement analysis: For example, let us consider the representative network illustrated in Fig. 5.1. We have 4 source nodes numbered 9, 10, 11, and 12. The sink node is numbered 0., and the rest is NFV capable relay nodes. The relay nodes can work both as a relay of traffic and an NFV node (with NFV functionalities). Based on the demand of the network, any relay node can be configured as an NFV node. So, our goal is to find out, which relay nodes are best suited to serve the source

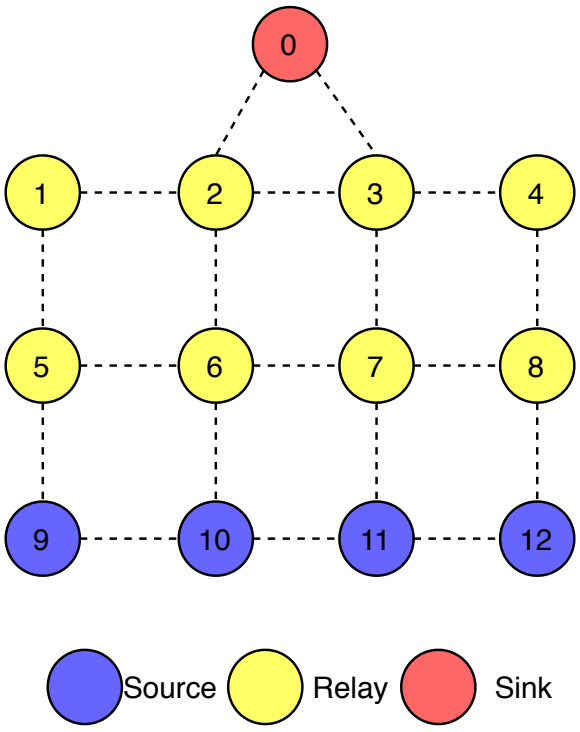


Figure 5.1: An example for without set of NFV nodes problem formulation.

nodes (node 9, 10, 11, and 12) as NFV nodes while minimizing the total number of NFV nodes and assign the source nodes to the selected NFV nodes over energy and interference optimized routes. The selection of NFV nodes from the set of relay nodes is based on the total cost of the route from the source node to the sink node via the NFV nodes. Also, the NFV nodes must have the capacity (CPU, memory, and energy) and available service-chaining. That means the selected NFV node for a source node must have the necessary resources to serve the source node and have the service-chain functions that the source demands.

So, like our previous ILP modeling, we need following types of constraints to enable our objective: *assignment constraints*, *capacity constraints*, *threshold constraints*, *supply-demand constraints*, and *flow conservation constraints*. The functionalities of all of these constraints remain the same as our previous model; only the operating range is different. In the previous model, we have a fixed small set of NFV nodes to search to get the solution. However, we need to search over all of the relay nodes to get the set of optimal NFV nodes, which provides the optimal solution.

Network modeling: For the formulation of our problem at hand, we can assume

that the IoT network can be represented by a graph $G = (V, L)$ where V represents all nodes in the network, and L gives us the links that connect those nodes. For our requirement analysis, it is evident that we need three types of nodes: source nodes $S \subset V$, relay nodes ($R \subset V$), and finally, the sink node ($S_0 \in V$), where $(S \cap R) = \emptyset$ and $(S \cup R) = V - S_0$. Let us assume; all relay nodes are capable of holding VNFs of a service-chain. Once configured, a relay node can act as a proper NFV node. All the source nodes S generate the traffic where the regular relay nodes R relay them towards the sink node S_0 via the activated relay nodes.

Definition of parameters: Definitions of all parameters remains the same as before. Only the range over which the parameter acts changes. For each relay node $r \in R$, the cost of activation depends on how much resource it is going to use and the amount of energy associated with the processing of hosted VNFs and communication (transmission and reception) overhead. So, we define the cost of activating a relay node as NFV node, c_r, as the summation of energy associated with resource utilization (CPU and Memory), $E_{utility}$, and the processing and communication energy, E_{pc}. The c_r is expressed in (5.1).

$$c_r = E_{utility} + E_{pc} \tag{5.1}$$

We define the capacity of a relay node as CP_r, which is the number of source nodes it can serve. This capacity is based on the available resources (CPU and Memory) and energy at the relay node. We also define the budget, B, of the total network. The budget depends upon the average capacity of the available relay nodes and the number of the source nodes. In (5.2), we define the budget B of the entire IoT network in terms of average capacity of the relay nodes, $\sum_{r \in R} CP_r \,/\, |R|$ and the number of source nodes $|S|$.

$$\lceil \frac{|S|}{\sum_{r \in R} CP_r \,/\, |R|} \rceil \leq B \tag{5.2}$$

The residual energy, e_j of each node $j \in V$ refers to the amount of energy each node j has left. We also denote the minimum residual energy threshold as E_j for each node $j \in V$. Same as before, we use RSSI value as an indicator for link quality. We denote r_{ij} as the RSSI value of any link $(i, j) \in L$, while the minimum RSSI threshold is denoted by R_{ij}.

Table 5.1: The list of notations used in the formulation.

Notation	**Description**
a_r	Decision variable to activate an NFV node r.
B	Budget for activating the relay node
c_r	Cost of activating a relay node r.
CP_r	Capacity of a relay node r.
D_{fs}	Any demand of node s for service f
E_j	Energy threshold for a node j.
e_j	Energy level of a node j
F	Set of service chains available
f	Any functions in the service chain set F
(i, j)	A link $(i, j) \in L$.
L	Set of all links.
R	Set of relay nodes.
R_{ij}	RSSI threshold value of a link (i, j)
r_{ij}	RSSI value of a link (i, j)
S	Set of source nodes.
S_0	Sink node.
S_{fr}	Availability of service f at node r
V	Set of all nodes.
$w1, w2$	Normalization or scaling factor.
x_{sr}	Decision variable for the assignment of a node s to node r.
y_{ijsr}	Decision variable for a link (i, j) selection for the assignment of a node s to node r.
z_{ijsr}	Combined decision variable for assignment of a node s to node r and link selection (i, j).

We also consider a set of service-chains $F = f_1, f_2, f_3, \ldots, f_n$ where each service-chain $f \in F$ consists of one or more VNFs available at the network. Same as our previous model, we assumed that all of the VNFs participating in a service-chain are consolidated at a single NFV activated relay node. For each source node, $s \in S$ the demand for the services of a service chain $f \in F$ is denoted by D_{fs}. On the other hand, the availability of a service chain $f \in F$ at a relay node $n \in N$ is denoted by

S_{fr}. A summary of all the notations used in the problem formulation is illustrated in Table.5.1.

Formulation: Our objective is to select the minimum number of relay nodes activated as NFV node and assign the source node to them over energy-efficient and interference-aware routes. To realize the objective, we define three decision variables: first is for the activated NFV nodes, second is for the assignment of source nodes to the NFV nodes, and the third is for the selection of routes for the assignment. The decision variables are presented below.

$$a_r = \begin{cases} 1, & \text{if } r \text{ is activated} \\ 0, & \text{otherwise} \end{cases} \tag{5.3}$$

$$x_{sr} = \begin{cases} 1, & \text{if } s \text{ is assigned to } r \\ 0, & \text{otherwise} \end{cases} \tag{5.4}$$

$$y_{ijsr} = \begin{cases} 1, & \text{if } (i,j) \text{ is selected for assignment of } s \text{ to } r \\ 0, & \text{otherwise} \end{cases} \tag{5.5}$$

In the objective function presented in (5.6), there are two parts: one is related to determining the activation of the minimum number of relay nodes as NFV nodes, and the other part is related to the assignment of source nodes to them over energy and interference optimized routes. The construction of routes from the source nodes to the sink node must be via the assigned NFV activated relay nodes. While constructing the routes, we consider the same factor as before. The routes must be of the shortest path while maximizing residual energy. In the objective function we use two normalization factor: w_1 and w_2. Both are used to scale the corresponding terms to the same standard as we use them in our previous model formulation.

$$\min \quad \sum_{r \in R} w_1 \, c_r \, a_r + \sum_{r \in R} \sum_{s \in S} \sum_{i,j \in L} x_{sr} \, y_{ijsr} \, (1 - w_2 \, e_j) \tag{5.6}$$

As we described in our requirement analysis, our objective is subject to some constraints. All the constraints has the similar functionality as the previous formulation, but they act over different reange. The constraints from (5.7) to (5.9) takes care of the activation and assignment of NFV activated relay nodes. In particular, (5.7) ensures that at least one relay node is to be activated. On the other hand, constraints in (5.8) make sure that each source node is to be assigned to exactly one NFV activated relay node. Constraints in (5.9) confirms that no source nodes should be assigned to non-activated relay nodes.

$$\text{s.t.} \qquad \sum_{r \in R} a_r \geq 1 \tag{5.7}$$

$$\sum_{r \in R} x_{sr} = 1, \qquad \forall s \in S \tag{5.8}$$

$$x_{sr} \leq a_r, \quad \forall s \in S, \forall r \in R \tag{5.9}$$

Constraints from (5.10) to (5.12) concerns over capacity and available energy at the activated relay nodes. Constraints in (5.10) ensures that the total number of assigned source nodes at an activated relay node must not exceed the capacity of that relay node. The constraints related to the budget of the network are described in (5.11). The constraint expressed in (5.12) makes sure that no relay nodes are to be activated if the total activation cost exceeds its residual energy. If there is not enough energy to accommodate the total activation cost, a relay node can not be activated and assigned to the source nodes.

$$\sum_{s \in S} x_{sr} \ \leq \ CP_r, \quad \forall r \in R \tag{5.10}$$

$$\sum_{r \in R} a_r \ \leq \ B \tag{5.11}$$

$$c_r a_r \ \leq \ e_r, \quad \forall r \in R \tag{5.12}$$

The constraints related to the availability of the service chains and its corresponding demand-supply at the source and relay nodes are described from (5.13) to (5.15).

$$\sum_{f \in F} D_{fs} \geq 1, \quad \forall s \in S \tag{5.13}$$

$$\sum_{f \in F} S_{fr} \geq 1, \quad \forall r \in R \tag{5.14}$$

$$x_{sr} D_{fs} \leq S_{fr}, \quad \forall f \in F, \forall s \in S, \forall r \in R \tag{5.15}$$

The constraints related to two thresholds: residual energy and RSSI value are described in (5.16) and (5.17).

$$y_{ijsr}\, e_j \geq E_j, \quad \forall (i,j) \in L, \forall s \in S, \forall r \in R \tag{5.16}$$

$$y_{ijsr}\, r_{ij} \geq R_j, \quad \forall (i,j) \in L, \forall s \in S, \forall r \in R \tag{5.17}$$

The corresponding flow conservation rules related to the construction of routes are described in the next set of constraints. The routes construction is done in two parts as in our previous formulation. One is from the source to the activated relay nodes, and the other is from there to the sink. The (5.18) and (5.19) take care of the first part, while (5.20) and (5.21) takes care of the second part with the help of (5.22), which takes care of all the relay nodes connecting the routes. Finally, (5.23) takes care of any cycles in the constructed routes.

$$\sum_{\substack{j\in(S\cup R)\\(i,j)\in L\\i=s}} y_{ijsr} = x_{sr}, \quad \forall s \in S, \forall r \in R \tag{5.18}$$

$$\sum_{\substack{i\in(S\cup R)\\(i,j)\in L\\j=r}} y_{ijsr} = x_{sr}, \quad \forall s \in S, \forall r \in R \tag{5.19}$$

$$\sum_{s\in S} \sum_{\substack{(i,j)\in L\\j\in(S_o\cup R)\\i=r}} y_{ijsr} = \sum_{s\in S} x_{sr}, \quad \forall r \in R \tag{5.20}$$

$$\sum_{s\in S} \sum_{\substack{(i,j)\in L\\i\in(N\cup R)\\j=S_o}} y_{ijsr} = x_{sr}, \quad \forall s \in S \forall r \in R \tag{5.21}$$

$$\sum_{(j,i)\in L} y_{jisr} = \sum_{(i,j)\in L} y_{ijsr}, \quad \forall s \in S, \forall r \in R, \forall j \in R \tag{5.22}$$

$$\sum_{\substack{j\in V\\(i,j)\in L}} y_{ijsr} \;\leq\; 1, \quad \forall i \in V, \forall s \in S, r \in R \tag{5.23}$$

The objective function presented in (5.6) has the same non-linear term as in (3.6). To make it linear, we bring another decision variable z_{ijsr} as the previous model. The new decision variable can replace the multiplying term of x_{sr} and y_{ijsr} in the (5.6). Also, to make the model stable, we need to place some other constraints. Our linearized version of the model is as follows:

$$\min \quad \sum_{r\in R} w_1 c_r\, a_r + \sum_{r\in R}\sum_{s\in S}\sum_{i,j\in L} z_{ijsr}\,(1 - w_2 e_j) \tag{5.24}$$

s.t. *Constraints* (5.7) *to* (5.23)

$$z_{ijsr} \leq x_{sr} \quad \forall (i,j) \in L, \forall s \in S, \forall r \in R \tag{5.25}$$

$$z_{ijsr} \leq y_{ijsr} \quad \forall (i,j) \in L, \forall s \in S, \forall r \in R \tag{5.26}$$

$$z_{ijsr} \geq x_{sr} + y_{ijsr} - 1 \;\forall (i,j) \in L, \forall s \in S, \forall r \in R \tag{5.27}$$

The ILP formulation we presented this chapter provides us to optimize the number of activating relay nodes and the assignment of source nodes to them over energy and interference aware routes. The problem is a combination of two classical NP-complete problems: Generalized Assignment Problem (GAP) and energy-aware routing the same as our ILP model presented in Chapter 3. So, the ILP model described in this chapter is also an NP-complete problem. Thus, we move to design a heuristic to facilitate this model for a large IoT network.

5.2 Heuristic

The main objective of our proposed ILP model for without a separate set of NFV nodes is the same as our model in Chapter 3. We have to minimize the number of activating relay node and assign the source nodes to them over energy and interference optimized routes. The only difference is that we are choosing from the set of relay nodes instead of a set of NFV nodes. The steps to design a heuristic for the ILP model without a separate set of NFV nodes are exactly the same as before with a little adjustment of search parameters. The adjusted heuristic is detailed in Algorithm 3.

Here also, we at first try to calculate the total cost of activating a relay node for a source node. We sort all the relay nodes based on the cost and select the best one from the list with available capacity and service-chain functionalities. We try to repeat the process until all the source nodes are assigned. To calculate the routes energy cost, we first calculate three disjoint shortest paths from the source to the relay node (line 3). Then, we select the best two paths based on the total energy cost of those paths and sort them into primary and secondary paths based on their link quality (line 4 to line 5). After that, we calculate the total route cost with the total energy cost of those paths and the activation cost of the relay node (line 6 to line 8). We repeat the process for all relay nodes. Then we select the best relay node for the source nodes based on its cost, capacity, and availability of the service-chain functionality (line 12 to line 16). We repeat the above procedure for all source nodes that need to be assigned to an activated relay node. To facilitate the heuristic, we use the same SDN/NFV architecture and communication protocol we described in Section 3.4.

Algorithm 3 Assignment and Path Selector Algorithm (APS) Without set of NFV nodes

Input
Relay nodes: R
Source nodes: S
Output
Activated relay nodes map: X_{SR}
All routes map: Y_{SR}

for $s \in S$ **do**
  for $r \in R$ **do**
    $Routes_{short} \leftarrow$ get 3-shortest routes from s to r
    $Routes_{energy} \leftarrow$ get 2 energy-cost routes from $Routes_{short}$
    $primary_s[r], secondary_s[r] \leftarrow$ sort($Routes_{energy}$, link-strength)
    $EnergyCost_s \leftarrow$ get total Energy ($primary_s[r]$, $secondary_s[r]$)
    $c_r \leftarrow$ get the activation cost of r
    $Cost_s[r] \leftarrow$ totalCost($EnergyCost_s, c_r$)
  end for
  $R_{sorted} \leftarrow$ sort the set of NFV($Cost_s, R$)
  for all $nfv_s \in R_{sorted}$ **do**
    if $nfv_s.capacity \leq CP_r$ & $D_{fs} \leq S_{fr}$ **then**
      $nfv_s.capacity \leftarrow nfv_s.capacity + 1$
      $S_{fr} \leftarrow S_{fr} - 1$
      $D_{fs} \leftarrow D_{fs} - 1$
      break
    end if
  end for
  $X_{SR}.append(s, nfv_s)$
  $Y_{SR}.append(s, primary_s[nfv_s], secondary_s[nfv_s])$
end forreturn X_{SR}, Y_{SR}

5.3 Evaluation

In this section, we first describe the evaluation setup of the ILP model without a separate set of NFV nodes ((5.24) s.t. (5.7) to (5.23) and (5.25) to (5.27)) and the corresponding heuristic (Algorithm 3). Then we present the results of both the model and heuristic and analyze them.

Evaluation Setup: For the most part, we use the same evaluation setup described in Section 4.1. The only difference is that we use a grid topology with 13 nodes, as in Fig. 5.1 for the model evaluation with one sink node, four source nodes and the rest as relay nodes. On the other hand, for the heuristic evaluation, we use 21 nodes grid topology with 4 source nodes, one sink node, and the rest as relay nodes. In both cases, we use the same node configurations and setup similar to the one in Section 4.1.

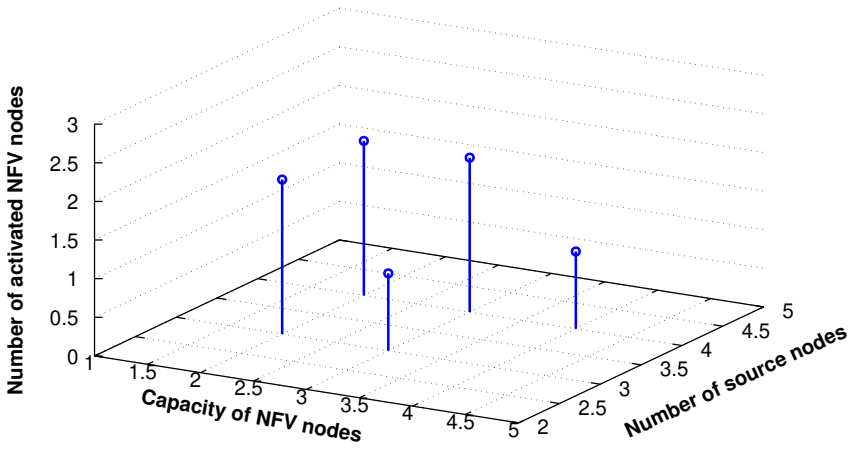


Figure 5.2: The number of activated NFV nodes.

5.3.1 Model Evaluation

For the model evaluation, we solved our ILP model without the separate set of NFV nodes using CPLEX solver like in the previous ILP model. In Fig. 5.2, we illustrate the effect of the number of source nodes and the capacity of relay nodes on the activated relay nodes. It is evident from the result that the number of activated relay nodes

increases with the increase of source nodes with constant capacity. On the other hand, with a fixed number of source nodes, it decreases with the increase in capacity to serve the source nodes. This result supports the budget of the total network, which directly proportional to the number of source nodes while being inversely proportional to the capacity of the relay nodes. The result provides the fact that the model correctly follows the budget of the network while minimizing the number of activated relay nodes.

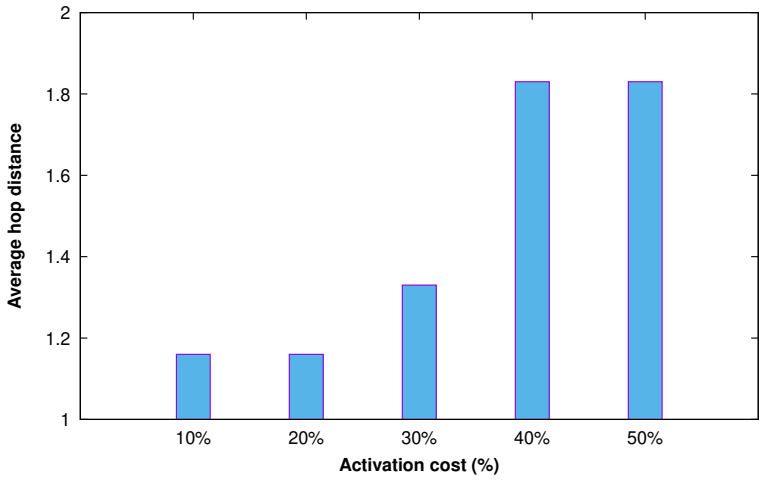


Figure 5.3: The change in hop distances.

The effect of increasing the activation cost on the average hop distance from the sources to the activated relay nodes is illustrated in Fig. 5.3. With increasing activation cost, the sources are assigned to the activated relay nodes with longer hop distances on an average. The results follow the same patterns shown in Fig. 4.4. The average hop distance is lower than our previous model evaluation as the model tries to place the activated relay nodes much closer to the source nodes. Our model primarily tries to minimize the transmission, which results in selecting the shortest path to the destination. With higher activation cost, the number of activated relay nodes decreases. With less activated relay nodes available, the sources have to be assigned to an activated relay node with longer distance. This shows the activation cost has a direct impact on our model solution.

In Fig. 5.4, we show the distribution of energy after 100 transmissions, and we use 30 transmissions as the point to switch between alternate paths to transmit. The use of two paths and the larger search parameter for optimal NFV assignment contributes

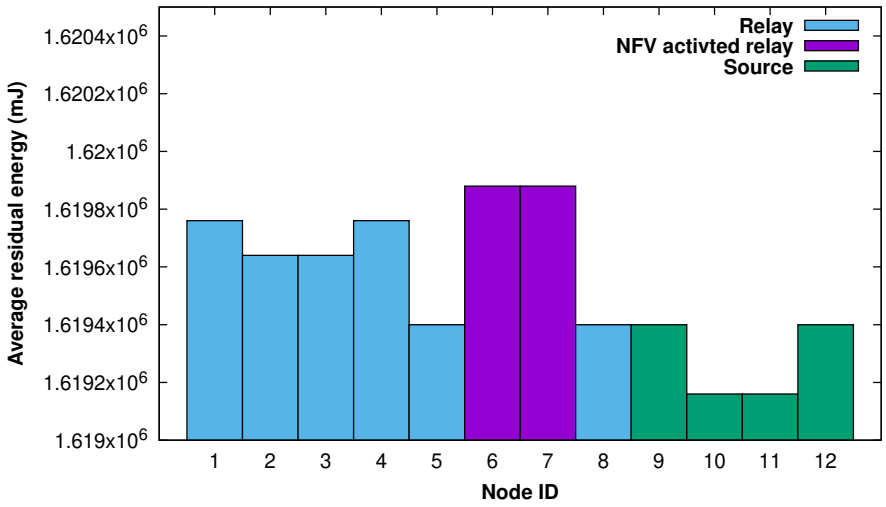


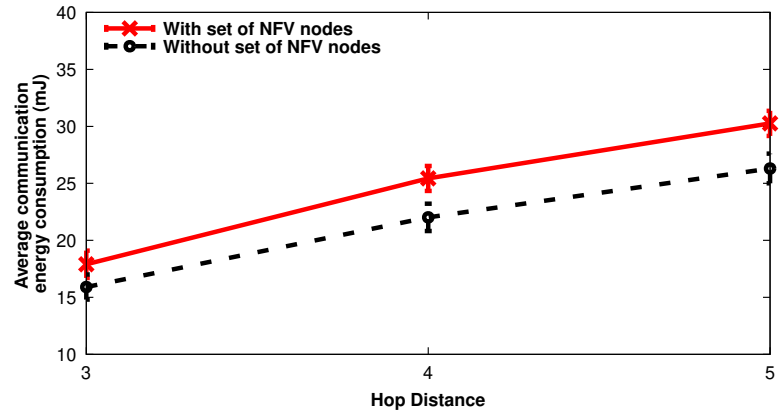
Figure 5.4: The residual energy of nodes.

to a better energy distribution than our previous model evaluation. The NFV nodes retain most energy among the participating nodes. Due to placing the activated relay nodes much closer to the source nodes, the number of transmissions by the relay nodes between the source and the assigned activated relay nodes decreases significantly. Although, the distance between the activated relay nodes and the sink node increases, due to having data aggregation as deployed VNFs, the number of transmission between them decreases, which leads to having a good energy distribution than our previous model evaluation (Fig. 4.5).

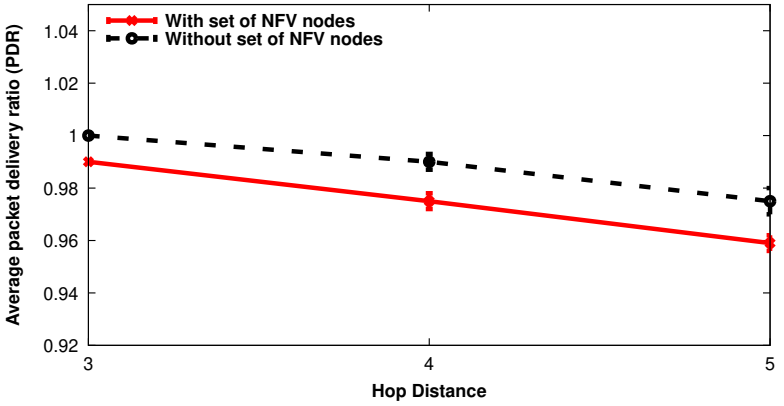
5.3.2 Heuristic Evaluation

We evaluate the proposed heuristic (Algorithm 3) and compare with the heuristic (APS Algorithm 1) described in Chapter 3.

Effect of hop distance: In this set of evaluations, we vary the hop distance between the source nodes and the sink node from 3 to 5 hop distance. In Fig. 5.5, we illustrate the effect of different hop distances on the communication energy consumption and PDR. In particular, Fig. 5.5a shows the effect of varying hop distance on the communication energy consumption. The results show similar trends to Fig. 4.8a where the energy consumption increases with the increase in hop distances. The heuristic without a set of NFV nodes shows better performance over heuristic with



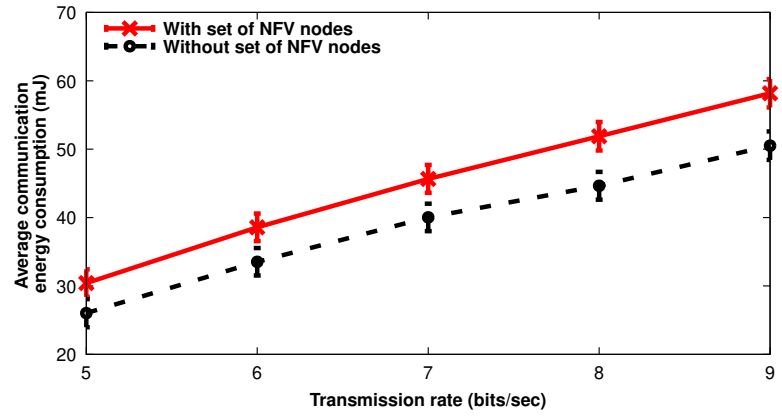
(a)



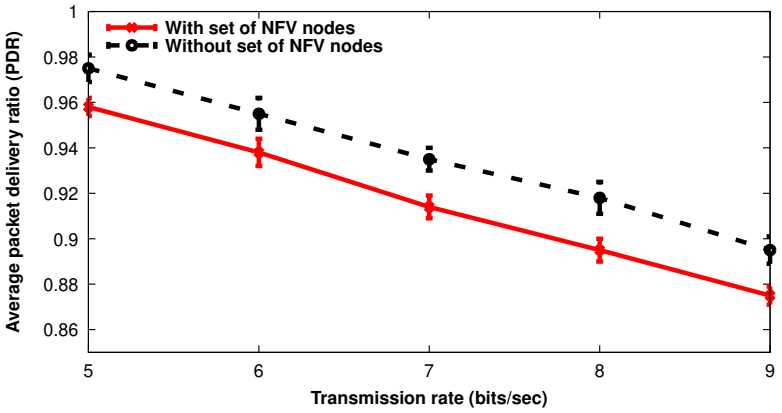
(b)

Figure 5.5: The average communication energy consumption (a) and PDR (b) in grid topology with and without set of NFV nodes.

a set of NFV nodes. The activated relay nodes are much closer to the source nodes in this case. This decreases the number of transmission between the source and the assigned activated relay nodes. Though the transmissions over the longer routes between activated relay nodes and the sink node are increased, with data aggregation as the deployed VNF, the overall number of transmission is minimized. This trend is also continued in the case of PDR, which is illustrated in Fig. 5.5b. It shows the heuristic without a set of NFV nodes has better performance. With increasing hop distance, the PDR of both cases decreases, consistent with the results illustrated in Fig. 4.8b. However, better energy management gives the heuristic without a set of



(a)



(b)

Figure 5.6: The impact of traffic loads on the average communication energy consumption (a) and PDR (b) in grid topology with and without set of NFV nodes.

NFV nodes an upper hand.

Effect of traffic load: In this part of the evaluations, we vary the transmission rate of the traffic to emulate the effect of different traffic loads. We choose the highest hop distance between the source and the sink node. We illustrate the behavior in Fig. 5.6. In particular, the effect of varying transmission rates on communication energy consumption is shown in Fig. 5.6a. The energy consumption increases with the increase of traffic load following the same trends, as in Fig. 4.10a. The heuristic without a set of NFV nodes comes on top. With higher traffic load, this improvement is more prominent. We suspect that the placement of activated relay nodes helps to

minimize the overall number of transmissions. And with a higher load, this is more evident. The effect on PDR is illustrated in Fig. 5.6b where the results also show similar trends as in Fig. 4.10b. With increasing load, the PDR drops. Here also, the heuristic without a set of NFV nodes comes on top as it offers better energy management, which consequently translated into a better network lifetime.

Comparison Analysis: In summary, the solution without a set of NFV nodes gives slightly better performance than the solution with a set of NFV nodes. This is because, in the case of without any set of NFV nodes, we have a better search range. However, the solution with a separate set of NFV nodes is much more practical. Because to make all the relay node NFV capable is costly. Moreover, the complexity of the solution increases as the search range increases. Thus, we argue that though the energy consumption might be a little higher, the solution with a set of NFV nodes is more feasible to implement.

Chapter 6

Conclusion and Future Works

6.1 Conclusions

In this work, we have proposed an energy-efficient and interference-aware SDN/NFV framework with a provision for service-chaining for IoT networks. First, we have formulated an Integer Linear Programming (ILP), which minimizes the activation of NFV nodes and assigns the source nodes to the activated NFV nodes so that total cost (activation cost and communication energy consumption) is minimized. Then, we have provided proof of the NP-completeness of our proposed ILP model. To facilitate the solution to a larger scale, we have then designed a heuristic. We also have proposed an SDN/NFV node architecture based on μSDN node architecture to implement the proposed heuristic. We have also designed the corresponding protocol to enable the SDN and NFV features like network programmability, VNF deployment, etc. To evaluate our proposed solution, we solved our ILP model using CPLEX and simulate the heuristic using the Cooja simulator, which is based on the Contiki OS. We have found that our proposed solution performs 1.39x better than μSDN and 1.6x better than SDN-WISE in terms of communication energy consumption. It also has provided better performance in terms of PDR than that of μSDN and SDN-WISE. We have also analyzed the control overhead of our solution and found that the overhead is comparable with RPL and SDN-WISE architecture. Lastly, we have shown that our architecture can dynamically configure the underlying network in case of any node failure.

In the above work, we have assumed that we have a separate set of NFV enabled IoT nodes other than the source, relay, and sink nodes. Later, we have generalized our solution, where we have lifted our assumption and considered all of the relay nodes are NFV capable which eliminates the necessity to have a separate set of NFV nodes. We have incorporated our consideration into our ILP model and corresponding heuristic, where we choose the best set of relay nodes for the activation of NFV nodes

from all of relay nodes relay nodes and assign them to the source nodes. Here also, we have used CPLEX to solve the generalized ILP and Cooja simulator and Contiki OS to implement the heuristic with corresponding SDN/NFV architecture and protocols. We have found that the result follows similar trends as to the previous solution but slightly outperforms the solution with set of NFV nodes. However, we argue that it is not feasible both computationally and realistically to consider all of the relay nodes with NFV capability [73].

In a real-life scenario, our proposed work can be applied in smart home applications where IoT sensor devices are used to collect information like temperature, humidity, etc. The data collector sensors are deployed in different rooms where the relay nodes can be deployed in the corridor with the sink node at the end of the floor or hallway. We can consider the organization in the form of a grid topology. The choice of the deployment of NFV nodes can be made using any of our proposed approaches, i.e., we can use a subset of NFV capable relays, or consider all relays with the NFV capabilities. Our solution can optimize resource consumption in the IoT network in the case of a high traffic volume.

6.2 Future Works

In this work, we have not studied the behavior of different energy models and radio duty cycling. Radio duty cycling can reduce energy-consumption further. We plan to integrate this technique into our solution in the future. On the interference end, we used a simple method of predicting the interference based on observing the link quality. In our future work, we plan to design an interference-estimator using other interference models. Also, the simulation environment poses some threats to external validity. To control external factors like network failure, congestion, traffic volume, etc. we have used either measurement or assigning some benchmark values. In the future, we intend to implement our solution in real test-beds to observe the real effects. Also, this work is a part of designing a complete end-to-end SDN/NFV based framework with heterogeneous network domains. In works ahead, we intend to extend our solution to provide a complete SDN/NFV based framework for different IoT network domains with different capabilities and connect them together.

Appendices

Appendix A

A.1 Implementation of EA-SDN/NFV

A.1.1 Network Setup & Configuration

Initialization: To run the EA-SDN/NFV heuristic, we need to initialize the network nodes according to the node architecture and their respective roles. The controller node, the source node, and the relay nodes are configured for initializing the SDN protocol and running the topology discovery phase. As our architecture is built on the μSDN architecture, we use its defined methods. The general flow of the node initialization is described in Listing A.1.

Listing A.1: Node Initialization

```
void node_init(){
    sdn_init();
    sdn_stats_start();
    Add_accept_ICMP6_RPL();
    /* if the node is source node*/
    if(IS_TX_NODE(node_id)){
        configure_to_transmit();
    }
    configure_to_relay();
}
```

After the initialization, the source nodes are configured for both generating data packets and relaying them, where others are configured for only relaying the data packets. For the controller node, we need to initialize the embedded controller and configure it as RPL DAG root. The general flow of this process is in Listing A.2.

Listing A.2: Controller Initialization

```
void controller_init(){
    configure_sdn();
    configure_rpl();
    simple_UDP_connection_register();
}
```

Configuration and state maintenance: After the initialization, the controller pushes the CONF message containing (SDN flow table setup, Flow table lifetime, its update period, RPL period information, default entry, etc.). The data structure of the CONF is given in the Listing A.3.

Listing A.3: CONF message

```
/* SDN Configuration data structure */
typedef struct sdn_configuration {
  /* general configuration */
  uint8_t       sdn_net;
  /* virtual network id */
  uint8_t       cfg_id;
  /* sdn configuration id */
  uint8_t       hops;
  /* hops from controller */
  /* flowtable configuration */
  clock_time_t  ft_lifetime;
  /* time to live for flowtable entries */
  uint8_t       query_full;
  /* query part of or full packet */
  uint8_t       query_idx;
  /* start index to query */
  uint8_t       query_len;
  /* length to query */
  /* rpl configuration */
  uint8_t       rpl_dio_interval;
```

```
  /* RPL_DIO_INTERVAL_MIN */
  uint8_t        rpl_dfrt_lifetime;
  /* RPL_DEFAULT_LIFETIME */
} sdn_cfg_t;
```

After receiving this CONF message, all the nodes configure themselves further and start collecting energy and link-state information using RPL DIO/DIS control packet exchange. With that information at hand, the nodes start preparing the NSU packet and start sending it to the controller node periodically. A node uses the `ENERGEST LIBRARY` to calculate its own energy level. The general data structure of the NSU packet is described in Listing A.4. The NSU packet contains the energy information of the node itself as well as that of the neighbors and corresponding link-state information.

Listing A.4: NSU message

```
typedef struct usdn_nsu_link {
  sdn_node_id_t nbr_id;
  int16_t       rssi;
} usdn_nsu_link_t;

typedef struct usdn_nsu {
  /* Node Info */
  uint8_t          cfg_id;
  uint8_t          rank;
  uint8_t        energy;
  uint8_t        energy_nbr[];
  /* Link Info */
  uint8_t          num_links;
  usdn_nsu_link_t links[];
} usdn_nsu_t;
```

A.1.2 Operation

Heuristic: After the configuration phase is over, the controller moves to start the proposed heuristic described in APS Algorithm (Algorithm 1). The controller collects information of the network state using the NSU updates. Based on the state of the network (initialization or change of the network state), the controller calls the heuristic (APS). For this purpose, the controller uses the DRS algorithm presented in Algorithm. 2. The main assignment of the NFV nodes is done in the APS algorithm, which at first calculates the 3 shortest paths from source to the NFV node and then selects the best 2 paths based on the energy-cost of the corresponding nodes. After that, we sort the 2 selected paths into the primary and secondary paths. The process repeats for all source NFV pairs. Based on the assignment and path cost, the APS decides on the final assignment. In the case of any change (any node goes below the energy threshold or any link's RSSI value goes below the RSSI threshold) in the network, the DRS algorithm decides to call the APS algorithm. The invoking of APS can be controlled based on the reconfiguration parameter. For example, whenever any NFV nodes go out, DRS can call the APS algorithm or if a certain percentage of source nodes get affected. The affected list of source nodes is calculated in (Listing A.5).

Listing A.5: Affected Source Node

```
static int* getAffected(){
  int result[TOPOLOGY_SIZE] = {0};
  atom_node_t node = NULL;
  for (int i = 0; i < TOPOLOGY_SIZE; ++i){
        for (int j = 0; j < TOPOLOGY_SIZE; ++j){
        if (prilist[i][j] != 0 || seclist[i][j] != 0){
                node = atom_net_get_node_id(routelist[i][j])
                if(node.energy >= ENERGY_THRESHOLD)
                    result[i] = 1;
        }
      }
    }
    return result;
```

```
}
```

The details of the APS algorithm (Algorithm 1) and DRS (Algorithm 2) are presented in [35] in the file *apps/atom/atom-heuristic.c*.

Operation of source nodes: After the assignment, the controller generates the NFV-CONF message (Listing A.6) to send the assignment information to the corresponding nodes. The NFV nodes configure themselves further to activate the VNF functions of the corresponding service chains. On the other hand, the source node upon receiving the assignment information sends an FTQ message to the controller. In response, the controller sends both the primary and secondary routes to the source nodes.

Listing A.6: NFV-CONF message

```
typedef struct nfv_configuration {
  uint8_t        nfv-node;
  uint8_t*        source;
  uint8_t        functional-value;
  unit8_t        NFV-buffersize;
} nfv_cfg_t;
```

Upon receiving the FTS message, the source nodes start sending the data packets. The source nodes use the NFV-buffer size value to alternate between the two routes to transmit. The NFV nodes check if there is any aggregated value in the queue and send that data (listing. A.7).

Listing A.7: Send Function

```
#ifdef BUILD_WITH_MULTIFLOW
static void
send(mflow_t *a)
{
  uint8_t buf[MAX_PAYLOAD] = {0};
  int pos = 0;
#if UIP_CONF_IPV6_SDN
if(sdn_connected()) {
```

```
    if(sdn_ft_contains(&a->remote_addr,
    sizeof(uip_ipaddr_t))){
#endif
      rpl_dag_t *dag = rpl_get_any_dag();
      if(dag != NULL) {
#if UIP_CONF_IPV6_SDN
        usdn_hdr_t hdr;
        hdr.net = SDN_CONF.sdn_net;
        hdr.typ = USDN_MSG_CODE_DATA;
        hdr.flow = a->id;
        memcpy(&buf, &hdr, USDN_H_LEN);
        pos += USDN_H_LEN;
#endif /* UIP_CONF_IPV6_SDN */
        if(IS_TX_NODE(node_id))
          sprintf((char *)buf + pos,
          "a:%d id:%d", a->id, a->seq);
        if(IS_NFV_NODE(node_id)){
          for (int i = 0; i < DEFAULT_CAPACITY; ++i)
          {
            /* code */
            if (aggregate[i]!=0)
            sprintf((char *)buf + pos,
            "a:%d id:%d", aggregate[i], a->seq);
          }
        }
        LOG_STAT("TX APP s:%d d:%d length:%d %s\n",
          node_id,
          a->remote_addr.u8[15],
          strlen((char *)(buf + pos))+pos,
          (buf + pos));
          mflow_send(a, &buf, strlen((char *)(buf + pos))
          + pos);
```

```
            int id = get_id(node_id);
            if (id !=-1)
            {
              node_count[id]++;
              if (node_count[id] == 10){
                node_count[id] = 0;
                route_set(a);
              }
            }
          }
            sdn_energy_print();
#if UIP_CONF_IPV6_SDN
      } else {
        controller_query_data
        (&a->remote_addr, sizeof(uip_ipaddr_t));
        sdn_energy_print();
      }
    } else {
      LOG_WARN("app %d can't send, no controller\n", a->id);
    }
#endif
}
#endif
```

For our implementation, we used data aggregation as our VNF. We use NFV buffer-size as the rate of aggregation. When NFV nodes receive the data sent from the source nodes, it stores them in buffer. After buffer gets full or reaches at the size of NFV buffer-size, it calls the aggregate function to aggregate them (Listing A.8).

Listing A.8: NFV Buffer Callback

```
static void
buffer_callback(mflow_t *a,
const uint8_t *data, uint16_t datalen){
```

```
  char buf[15] = {0};
  int pos = 0;
#if UIP_CONF_IPV6_SDN
  usdn_hdr_t hdr;
  memcpy(&hdr, data, USDN_H_LEN);
  pos += USDN_H_LEN;
#endif
  memcpy(&buf, data + pos, datalen);
  BUF[count] = (uint8_t)buf[2];
  count++;
  if(count == NFV_BUFFER_SIZE){
    count = 0;
    aggregate();
    mflow_new_sendinterval(nfv_tx_app, 0);
    send(nfv_tx_app);
  }
```

In summary, we have tried to give as details as possible about our implementation. We provide full implementation along with detailed instruction to reproduce the results in [35].

Bibliography

[1] R. Spencer, "What is an iot gateway?" Mar 2020. [Online]. Available: https://www.lanner-america.com/blog/what-is-an-iot-gateway/

[2] D. Kreutz, F. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *IEEE*, 2015.

[3] M. Ersue, "Etsi nfv management and orchestration-an overview," *Presentation at the IETF*, vol. 88, 2013.

[4] I. Haque and D. Saha, "On the benefits of network programmability in iot networks," in *IEEE Transactions on Network and Service Management (IEEE TNSM)), Under Review*, 2020.

[5] CISCO, "Cisco annual internet report," accessed March 23,2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/executive-perspectives/annual-internet-report/index.html

[6] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, 2015.

[7] I. Haque and N. Abu-Ghazaleh, "Wireless software defined networking: a survey and taxonomy," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 4, pp. 2713–2737, May 2016.

[8] P. Neves, *et al.*, "The SELFNET approach for autonomic management in an NFV/SDN networking paradigm," *International Journal of Distributed Sensor Networks*, vol. 12, no. 2, p. 2897479, 2016.

[9] M. Boucadair, C. Jacquenet, R. Parker, D. Lopez, J. Guichard, and C. Pignataro, "Service function chaining: Framework & architecture," February 2014, accessed May 06,2020. [Online]. Available: https://tools.ietf.org/html/draft-boucadair-sfc-framework-02

[10] M. Baddeley, R. Nejabati, G. Oikonomou, M. Sooriyabandara, and D. Simeonidou, "Evolving SDN for low-power IoT networks," in *IEEE NetSoft*, 2018, pp. 71–79.

[11] T. Winter *et al.*, "RPL: IPv6 routing protocol for low-power and lossy networks," IETF, Tech. Rep., 2012.

[12] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks," in *IEEE INFOCOM*, 2015, pp. 513–521.

[13] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[14] A.-C. G. Anadiotis, L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SD-WISE: a software-defined wireless sensor network," *arXiv preprint arXiv:1710.09147*, 2017.

[15] A. Markiewicz, P. N. Tran, and A. Timm-Giel, "Energy consumption optimization for software defined networks considering dynamic traffic," in *IEEE CloudNet*, 2014, pp. 155–160.

[16] F. Giroire, J. Moulierac, and T. K. Phan, "Optimizing rule placement in software-defined networks for energy-aware routing," in *IEEE GLOBECOM*, 2014, pp. 2523–2529.

[17] A. Fernandez-Fernandez, C. Cervello-Pastor, and L. Ochoa-Aday, "Achieving energy efficiency: An energy-aware approach in SDN," in *IEEE GLOBECOM*, 2016, pp. 1–7.

[18] Y. Yao, Q. Cao, and A. V. Vasilakos, "EDAL: An energy-efficient, delay-aware, and lifetime-balancing data collection protocol for heterogeneous wireless sensor networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 23, no. 3, pp. 810–823, 2015.

[19] K. Kaur *et al.*, "An energy-driven network function virtualization for multi-domain software defined networks," *arXiv preprint arXiv:1903.09924*, 2019.

[20] Z. Ding, S. Xing, F. Yan, W. Xia, and L. Shen, "An interference-aware energy-efficient routing algorithm with quality of service requirements for software-defined wsns," *IET Communications*, vol. 13, no. 18, pp. 3105–3116, 2019.

[21] L. Farhan, O. Kaiwartya, L. Alzubaidi, W. Gheth, E. Dimla, and R. Kharel, "Toward interference aware iot framework: Energy and geo-location-based-modeling," *IEEE Access*, vol. 7, pp. 56 617–56 630, 2019.

[22] R. Musaloiu-E and A. Terzis, "Minimising the effect of wifi interference in 802.15. 4 wireless sensor networks," *International Journal of Sensor Networks*, vol. 3, no. 1, pp. 43–54, 2008.

[23] I. T. Haque and C. Assi, "OLEAR: Optimal localized energy aware routing in mobile ad hoc networks," in *Proceedings of the 2005 IEEE International Conference on Communications*, ser. ICC '05, 2005.

[24] I. Haque, C. Assi, and W. Atwood, "Randomized energy-aware routing algorithms in mobile ad hoc networks," in *Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, ser. MSWiM '05, 2005.

[25] I. Haque and C. Assi, "Localized energy efficient routing in mobile ad hoc networks," *The Willey Journal of Wireless and Mobile Computing*, vol. 7, no. 6, pp. 781–793, August 2007.

[26] I. Haque, M. Nurujjaman, J. Harms, and N. Abu-ghazaleh, "SDSense: An agile and flexible SDN-based framework for wireless sensor networks," *The IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1866 – 1876, February 2019.

[27] I. T. Haque, "Non-deterministic geographic forwarding in mobile ad hoc networks," in *2008 IEEE Asia-Pacific Services Computing Conference*. IEEE, 2008, pp. 1144–1149.

[28] I. Haque, I. Nikolaidis, and P. Gburzynski, "On the benefits of nondeterminism in location-based forwarding," in *International Conference on Communications (ICC)*, 2009.

[29] D. Saha, M. Shojaee, M. Baddeley, and I. Haque, "An Energy-Aware SDN/NFV architecture for the internet of things," in *IFIP Networking 2020 Conference (IFIP Networking 2020)*, Paris, France, Jun. 2020.

[30] M. Khan *et al.*, "Big data processing using internet of software defined things in smart cities," *International Journal of Parallel Programming*, pp. 1–14, 2018.

[31] Y.-B. Lin, S.-Y. Wang, C.-C. Huang, and C.-M. Wu, "The SDN approach for the aggregation/disaggregation of sensor data," *Sensors*, vol. 18, no. 7, p. 2025, 2018.

[32] J. Hartmanis, "Computers and intractability: a guide to the theory of np-completeness (michael r. garey and david s. johnson)," *Siam Review*, vol. 24, no. 1, p. 90, 1982.

[33] Y. B. Zikria, S. W. Kim, O. Hahm, M. K. Afzal, and M. Y. Aalsalem, "Internet of things (iot) operating systems management: opportunities, challenges, and solution," 2019.

[34] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *IEEE international conference on local computer networks*, 2004, pp. 455–462.

[35] D. Saha, M. Baddeley, and I. Haque, "EA-SDN/NFV source code," 2020. [Online]. Available: https://github.com/dipon76/EA-SDN-NFV-THESIS-final

[36] M. Burgess, "What is the internet of things? wired explains," Feb 2018. [Online]. Available: https://www.wired.co.uk/article/internet-of-things-what-is-explained-iot

[37] K. K. Patel, S. M. Patel *et al.*, "Internet of things-iot: definition, characteristics, architecture, enabling technologies, application & future challenges," *International journal of engineering science and computing*, vol. 6, no. 5, 2016.

[38] H. Ali, "A performance evaluation of rpl in contiki," 2012.

[39] O. Iova, P. Picco, T. Istomin, and C. Kiraly, "RPL: The routing standard for the internet of things... or is it?" *IEEE Communications Magazine*, vol. 54, no. 12, pp. 16–22, December 2016.

[40] A. M. Alshnta, M. F. Abdollah, and A. Al-Haiqi, "Sdn in the home: A survey of home network solutions using software defined networking," *Cogent Engineering*, vol. 5, no. 1, p. 1469949, 2018.

[41] I. Alam, K. Sharif, F. Li, Z. Latif, M. Karim, S. Biswas, B. Nour, and Y. Wang, "A survey of network virtualization techniques for internet of things using sdn and nfv," *ACM Computing Surveys (CSUR)*, vol. 53, no. 2, pp. 1–40, 2020.

[42] C. Pham, N. H. Tran, S. Ren, W. Saad, and C. S. Hong, "Traffic-aware and energy-efficient vnf placement for service chaining: Joint sampling and matching approach," *IEEE Transactions on Services Computing*, 2017.

[43] S. A. Dehkordi, K. Farajzadeh, J. Rezazadeh, R. Farahbakhsh, K. Sandrasegaran, and M. A. Dehkordi, "A survey on data aggregation techniques in iot sensor networks," *Wireless Networks*, vol. 26, no. 2, pp. 1243–1263, 2020.

[44] E. T. FUTE, A. B. BOMGNI, and H. M. KAMDJOU, "An approach to data compression and aggregation in wireless sensor networks," *International Journal of Computer Science and Telecommunications*, 2016.

[45] A. Ullah, G. Said, M. Sher, and H. Ning, "Fog-assisted secure healthcare data aggregation scheme in iot-enabled wsn," *Peer-to-Peer Networking and Applications*, vol. 13, no. 1, pp. 163–174, 2020.

[46] U. Noreen, A. Bounceur, and L. Clavier, "Modeling interference for wireless sensor network simulators," in *Proceedings of the International Conference on Future Networks and Distributed Systems*, 2017, pp. 1–6.

[47] G. Zhou, T. He, J. A. Stankovic, and T. Abdelzaher, "Rid: Radio interference detection in wireless sensor networks," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 2. IEEE, 2005, pp. 891–901.

[48] V. Iyer, F. Hermans, and T. Voigt, "Detecting and avoiding multiple sources of interference in the 2.4 ghz spectrum," in *European Conference on Wireless Sensor Networks*. Springer, 2015, pp. 35–51.

[49] C. Noda, S. Prabh, M. Alves, C. A. Boano, and T. Voigt, "Quantifying the channel quality for interference-aware wireless sensor networks," *ACM SIGBED Review*, vol. 8, no. 4, pp. 43–48, 2011.

[50] D. Konings, N. Faulkner, F. Alam, F. Noble, and E. M. Lai, "The effects of interference on the rssi values of a zigbee based indoor localization system," in *2017 24th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*. IEEE, 2017, pp. 1–5.

[51] S. Tomovic *et al.*, "An architecture for qos-aware service deployment in software-defined IoT networks," in *International Symposium on Wireless Personal Multimedia Communications (WPMC)*, 2017, pp. 561–567.

[52] M. Ojo, D. Adami, and S. Giordano, "A sdn-iot architecture with nfv implementation," in *2016 IEEE Globecom Workshops (GC Wkshps)*, 2016, pp. 1–6.

[53] S. Wang, A. Hawbani, X. Wang, O. Busaileh, and L. Ping, "Heuristic routing for software defined wireless sensor network," in *2018 World Symposium on Digital Intelligence for Systems and Machines (DISA)*. IEEE, 2018, pp. 39–44.

[54] N. Omnes, M. Bouillon, G. Fromentoux, and O. Le Grand, "A programmable and virtualized network & it infrastructure for the internet of things: How can nfv & sdn help for facing the upcoming challenges," in *2015 18th International Conference on Intelligence in Next Generation Networks*. IEEE, 2015, pp. 64–69.

[55] D. Sinh, L.-V. Le, B.-S. P. Lin, and L.-P. Tung, "Sdn/nfv—a new approach of deploying network infrastructure for iot," in *2018 27th Wireless and Optical Communication Conference (WOCC)*. IEEE, 2018, pp. 1–5.

[56] I. Haque, S. Islam, and J. Harms, "On selecting a reliable topology in wireless sensor networks," in *Proceedings of the 2015 IEEE International Conference on Communications*, ser. ICC '15, 2015.

[57] T. Fevens, I. Haque, and L. Narayanan, "A class of randomized routing algorithms in mobile ad hoc networks," *AlgorithmS for Wireless and mobile Networks (A SWAN 2004), Boston*, 2004.

[58] I. T. Haque, I. Nikolaidis, and P. Gburzynski, "Expected path length for angle and distance-based localized routing," in *2009 International Symposium on Performance Evaluation of Computer Telecommunication Systems*, vol. 41, 2009, pp. 137–141.

[59] A. Barbato, M. Barrano, A. Capone, and N. Figiani, "Resource oriented and energy efficient routing protocol for ipv6 wireless sensor networks," in *2013 IEEE Online Conference on Green Communications (OnlineGreenComm)*. IEEE, 2013, pp. 163–168.

[60] D. H. McBurney, "Research methods," *Wadsworth/Thomson Learning*, 2001.

[61] P. C. Chu and J. E. Beasley, "A genetic algorithm for the generalised assignment problem," *Computers & Operations Research*, vol. 24, no. 1, pp. 17–23, 1997.

[62] F. Giroire, D. Mazauric, J. Moulierac, and B. Onfroy, "Minimizing routing energy consumption: from theoretical to practical results," in *IEEE/ACM Int'l Conference on Green Comput. and Comm.*, 2010, pp. 252–259.

[63] A. Pourkabirian and A. T. Haghighat, "Energy-aware, delay-constrained routing in wireless sensor networks through genetic algorithm," in *2007 15th International Conference on Software, Telecommunications and Computer Networks*. IEEE, 2007, pp. 1–5.

[64] M. K. Awad, Y. Rafique, and R. A. M'Hallah, "Energy-aware routing for software-defined networks with discrete link rates: A benders decomposition-based heuristic approach," *Sustainable Computing: Informatics and Systems*, vol. 13, pp. 31–41, 2017.

[65] B. Yaged Jr, "Minimum cost routing for static network models," *Networks*, vol. 1, no. 2, pp. 139–172, 1971.

[66] G. M. Guisewite and P. M. Pardalos, "Minimum concave-cost network flow problems: Applications, complexity, and algorithms," *Annals of Operations Research*, vol. 25, no. 1, pp. 75–99, 1990.

[67] T. H. Coreman, C. E. Leiserson, R. L. Ronald, and C. Stein, *Introduction to Algorithms*. MIT Press, 1990.

[68] "Cplex optimizer," accessed July 11, 2020. [Online]. Available: https://www.ibm.com/analytics/cplex-optimizer

[69] C. Bliek1ú, P. Bonami, and A. Lodi, "Solving mixed-integer quadratic programming problems with IBM-CPLEX : a progress report," in *Proceedings of the twenty-sixth RAMP symposium*, 2014, pp. 16–17.

[70] P. Saurav, "Different types of batteries in iot devices," Apr 2018. [Online]. Available: https://www.baseapp.com/iot/batteries-iot-device-types-character/

[71] "Datasets." [Online]. Available: http://casas.wsu.edu/datasets/

[72] "Msp430g2553 internal temperature sensor." [Online]. Available: https://e2e.ti.com/support/microcontrollers/msp430/f/166/t/258911?MSP430G2553-internal-temperature-sensor

[73] M. Bouet, J. Leguay, T. Combe, and V. Conan, "Cost-based placement of vdpi functions in nfv infrastructures," *International Journal of Network Management*, vol. 25, no. 6, pp. 490–506, 2015.